2. TEORI PENUNJANG

2.1. Sistem Informasi Manajemen

2.1.1. Pengertian Sistem

Sistem didefinisikan sebagai sesuatu yang diciptakan untuk menjalankan sesuatu hal yang rutin dijalankan. Sistem merupakan kumpulan dari bagian-bagian sistem yang lebih kecil yang disebut subsistem. Dan subsistem itu sendiri merupakan unsur-unsur yang dirangkaikan menjadi struktur sistem.

Selain itu diperlukan adanya pendekatan sistem yang digunakan untuk menganalisa jalannya sistem jika terjadi adanya kesalahan dalam pembentukan sistem yang menyebabkan sistem tersebut tidak berfungsi secara optimal atau sesuai dengan tujuan sistem tersebut diciptakan. Suatu sistem harus dikembangkan sesuai dengan berkembangnya pembentukan sistem, karena itu diperlukan adanya informasi yang berguna sebagai unsur pembentuk subsistem. (Mulyadi, 1997)

Saat ini penggunaan konsep sistem sangat luas dan meliputi berbagai bidang sehingga timbul berbagai definisi tentang sistem, masing-masing menekankan pada sudut pandang dan kebutuhan sendiri.

Sistem adalah suatu kesatuan (*entity*) yang terdiri dari bagian-bagian (disebut subsistem) yang saling berkaitan dengan tujuan untuk mencapai tujuan tertentu. (Baridwan, 1993)

Sedangkan menurut Dr. Richardus Eko Indrajit (2000):

Sistem adalah kumpulan dari komponen-komponen yang memiliki unsur keterkaitan antara satu dan lainnya.

Dengan kata lain, sistem adalah suatu satuan (*entity*) yang terdiri dari dua atau lebih komponen (subsistem) atau suatu kerangka kerja terpadu yang terjalin satu sama lain untuk mencapai suatu tujuan/satu sasaran.

Ada 4 unsur utama dalam suatu sistem, yakni:

(1). Terdiri dari elemen-elemen atau bagian-bagian.

(2). Mempunyai interaksi antar elemen-elemen tersebut.

(3). Adanya suatu yang mengikat elemen-elemen tersebut menjadi suatu kesatuan.

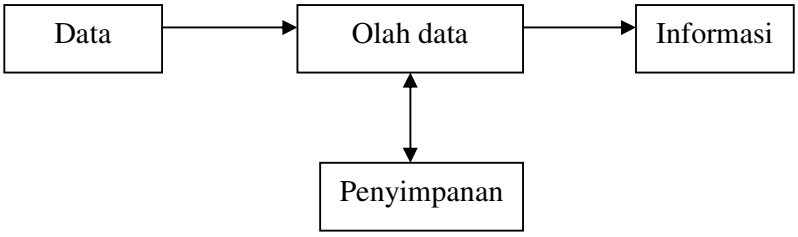
(4). Mempunyai tujuan sebagai hasil akhir.

2.1.2. Pengertian Informasi

Informasi merupakan kebutuhan yang vital di dalam tubuh suatu organisasi. Fungsi utama dari informasi adalah mengurangi ketidakpastian di dalam proses pengambilan keputusan tentang suatu keadaan.

Informasi adalah sesuatu yang dihasilkan dari pengolahan data yang dilakukan oleh suatu sistem. (Iryantoro, 2000)

Hubungan antara data dengan informasi dapat digambarkan sebagai berikut :



Gambar 2.1. Hubungan Data dengan Informasi

2.1.3. Pengertian Sistem Informasi

Informasi merupakan hal yang sangat penting dalam pengambilan keputusan. Pengolahan data menjadi informasi disajikan oleh suatu sistem, yaitu sistem informasi. Sistem Informasi merupakan suatu sistem yang melakukan fungsi untuk menyediakan semua informasi yang dibutuhkan. Mengolah data dan kemudian menyajikan informasi adalah suatu hal yang terintegrasi. Sistem informasi adalah suatu sistem yang bertujuan menyajikan informasi kepada penggunanya melalui hasil pengolahan data yang sesuai. (Iryantoro, 2000)

Informasi dalam suatu badan usaha dihasilkan oleh suatu sistem informasi, menurut Dr. Richardus Eko Indrajit (2000), definisi sistem informasi adalah:

Sistem Informasi merupakan suatu kumpulan dari komponen-komponen dalam perusahaan atau organisasi yang berhubungan dengan proses penciptaan dan pengaliran informasi.

Jadi sistem informasi merupakan suatu kerangka kerja yang mengkoordinasi sumber daya-sumber daya (manusia, komputer) melalui suatu jaringan prosedur-prosedur yang terintegrasi untuk mengolah input (data) menjadi suatu bentuk output (informasi) yang dapat bermanfaat bagi pemakainya untuk pengambilan keputusan dalam rangka mencapai tujuan badan usaha yang bersangkutan.

Dengan kata lain, suatu sistem informasi mengubah data mentah menjadi laporan manajemen atau laporan keuangan bagi pihak internal maupun eksternal perusahaan melalui proses pengolahan data. Sistem informasi suatu badan usaha dalam dunia bisnis memiliki 3 tujuan, yaitu:

a. Untuk menyediakan informasi yang mendukung pengambilan keputusan.
b. Untuk menyediakan informasi yang dapat mendukung kegiatan operasi sehari-hari.
c. Untuk menyediakan informasi yang berkenaan dengan pelayanan.

Jadi, keandalan suatu sistem informasi dalam badan usaha terletak pada keterkaitan antar komponen yang ada (proses dan prosedur, struktur organisasi, sumber daya manusia, produk, pelanggan, *supplier*, rekanan, dan lain sebagainya), sehingga dapat dihasilkan dan dialirkan suatu informasi yang berguna (akurat, terpercaya, detail, cepat, relevan, dan sebagainya) untuk badan usaha yang bersangkutan.

Ada 3 macam dimensi sistem informasi dalam mendukung sistem manajemen:

- *Data Processing (Elements)*

  Merupakan komponen fisik dari sistem, yang terdiri dari *hardware, data storage equipment, software, and operating procedure.*

- *Managerial Activities*

  Aktivitas-aktivitas manajemen sangat erat hubungannya dengan tingkatan-tingkatan yang ada dalam organisasi. Aktivitas-aktivitas ini berpengaruh pada proses informasi karena informasi yang dibutuhkan pada setiap tingkat manajerial adalah berbeda.

- *Organizational Functions*

  Pada umumnya setiap badan usaha berskala besar mempunyai struktur organisasi yang dibedakan berdasarkan fungsi-fungsi seperti pemasaran, produksi, keuangan, personalia dan akuntansi. Karena alasan inilah diperlukan pengorganisasian dan pengembangan sistem informasi.

2.1.4. Pengertian Sistem Informasi Manajemen

Salah satu sumber daya yang tersedia bagi seorang manager adalah informasi, dimana informasi ini dapat dikelola seperti sumber daya yang lain yang membentuk suatu sistem informasi sesuai dengan konsep dasar informasi. Agar suatu sistem dapat dikenal dengan baik, maka sistem tersebut harus dipelajari. Sistem didefinisikan sebagai kumpulan dari beberapa elemen yang berinteraksi untuk mencapai tujuan tertentu. Suatu sistem mempunyai susunan dasar, antara lain: *input, output*, transformasi, mekanisme pengendalian, dan tujuan. (McLeod, 2000).

Sistem Informasi Manajemen (*Management Information System/*MIS) merupakan sistem berbasis komputer yang membuat informasi tersedia bagi pengguna yang bertujuan sama. MIS menghasilkan informasi yang tersedia bagi semua bagian perusahaan. Ide utama dari MIS adalah memberikan *supply* informasi yang terus-menerus bagi manajer untuk dapat mengenali masalah, mengerti masalah dan mampu untuk memecahkan masalah dengan pengambilan keputusan.

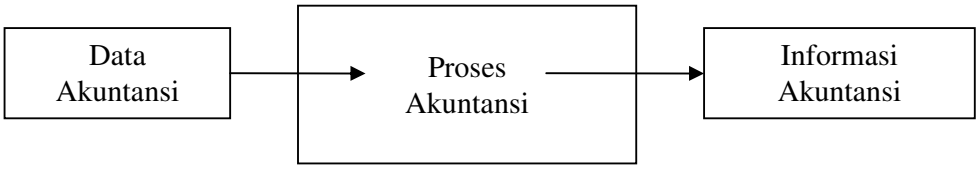
2.2. Sistem Informasi Akuntansi

2.2.1. Pengertian Sistem Informasi Akuntansi

Akuntansi berasal dari bahasa bisnis. Di dalam akuntansi, data-data yang berupa transaksi-transaksi akan diolah sedemikian rupa hingga menjadi suatu informasi yang berupa laporan yang berguna untuk pengambilan keputusan.

Sistem Informasi Akuntansi adalah pengolahan data akuntansi dengan menggunakan teknologi komputer dengan tujuan mengendalikan aktivitas input dan output yang mempunyai nilai uang. (Iryantoro, 2000)

Secara blok diagram sederhana dapat digambarkan :



Gambar 2.2. Blok Diagram Sistem Informasi Akuntansi

Definisi Sistem Informasi Akuntansi menurut George H. Bodnar Dan William S. Hopwood (1995):

An accounting information system is a collection of resources, such as people and equipment, designed to transform financial data into information.

Menurut Gelinas dan Oram (1996), definisi Sistem Informasi Akuntansi adalah:

The accounting information system is a specialized subsystem of the management information system whose purpose is to collect, process, and report information related to financial transaction.

Menurut Steven A. Moscove yang dikutip oleh Drs. Zaki Baridwan (1993):

Sistem Informasi Akuntansi adalah suatu komponen organisasi yang mengumpulkan, menggolongkan, mengolah, menganalisa, dan mengkomunikasikan informasi keuangan yang relevan untuk pengambilan keputusan kepada pihak-pihak luar (seperti inspeksi pajak, investor, dan kreditur) dan pihak-pihak dalam (terutama manajemen).

Jadi, dapat disimpulkan bahwa Sistem Informasi Akuntansi adalah kumpulan manusia dan sumber-sumber modal di dalam suatu organisasi yang bertanggung jawab untuk penyiapan informasi keuangan dan informasi yang diperoleh dari pengumpulan dan pengolahan data transaksi selanjutnya informasi ini disediakan untuk dipakai oleh semua tingkatan manajemen dalam perencanaan dan pengendalian aktivitas organisasi atau dengan kata lain Sistem Informasi Akuntansi adalah bagian dari organisasi yang mengumpulkan, mengklasifikasikan, memproses, menganalisis dan mengkomunikasikan data-data keuangan guna menghasilkan informasi, baik bagi pihak luar maupun badan

usaha, sebagai dasar dalam perencanaan, pengendalian dan pengambilan keputusan.

2.2.2. Tujuan dan Manfaat Sistem Informasi Akuntansi

Setiap Sistem Informasi dibentuk untuk mencapai satu atau lebih tujuan tertentu. Tujuan akhir dari kegiatan akuntansi adalah informasi, yaitu dengan cara membuat atau menerbitkan laporan-laporan yang diperlukan (laporan keuangan).

Laporan-laporan ini disiapkan untuk dua kelompok lain yang berbeda, dimana masing-masing mempunyai kebutuhan yang berbeda pula. Pihak-pihak yang membutuhkan laporan tersebut adalah pihak eksternal dan pihak internal perusahaan. Pihak eksternal menggunakan laporan-laporan yang dihasilkan perusahaan sebagai dasar pengambilan keputusan mengenai hubungan mereka dengan badan usaha, sedangkan pihak internal memanfaatkan laporan ini berkaitan dengan rencana badan usaha yang menyangkut penggunaan anggaran dan untuk pengambilan keputusan.

Sistem Informasi Akuntansi merupakan sumber utama dari informasi yang dibutuhkan karena dapat mengumpulkan dan mengolah data-data transaksi yang ada menjadi suatu informasi yang dapat dipertanggung jawabkan dan berguna dalam pengambilan putusan yang berkaitan dengan rencana-rencana maupun untuk mengadakan pengendalian terhadap aktivitas-aktivitas yang terjadi di badan usaha.

2.3. Perancangan Sistem

Dalam melakukan perancangan terhadap sistem, maka ada dua model yang dipergunakan dalam melakukan desain sistem yaitu *process modeling* dengan menggunakan *Data Flow Diagram* (DFD) dan data *modeling* dengan menggunakan *Entity Relationship Diagram* (ERD).

2.3.1. *Data Flow Diagram* (DFD)

2.3.1.1. Definisi *Data Flow Diagram* (DFD)

Sebelum membuat program *database* dilakukan pembuatan *Data Flow Diagram* (DFD) terlebih dahulu. Beberapa definisi mengenai DFD adalah sebagai berikut:

Data Flow Diagram (DFD) adalah suatu grafik yang menjelaskan sebuah sistem dengan menggunakan bentuk-bentuk simbol untuk menggambarkan aliran data dari proses-proses yang saling berhubungan (Raymond McLeod, 2000).

Data Flow Diagram (DFD) merupakan penggambaran dari sistem yang menggunakan bentuk simbol untuk menggambarkan aliran dari data dalam suatu proses yang saling berhubungan (McLeod, Jr., Schell, 1979).

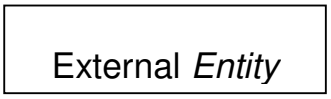
Data Flow Diagram (DFD) adalah model sistem untuk mengekspresikan aliran data antar proses. (Jalote, 1994).

Data Flow Diagram (DFD) adalah representasi dari sebuah sistem secara grafis yang digambarkan dengan sejumlah simbol tertentu untuk menunjukkan perpindahan data dalam proses-proses suatu sistem. Dalam hal ini, DFD menunjukkan perpindahan dan perubahan data dalam suatu sistem. Meskipun diberi nama *Data Flow Diagram* (DFD), namun penekanan pada DFD lebih pada prosesnya, bahkan DFD merupakan salah satu alat pemodelan proses dari sistem yang paling sering digunakan. DFD menggambarkan *input,* proses, dan *output* yang terjadi dalam suatu sistem. Diagram aliran data (*Data Flow Diagram* atau DFD) adalah teknik yang digunakan untuk menjelaskan aliran informasi/transformasi data yang bergerak dari pemasukan data hingga keluar data. DFD merupakan salah satu alat bantu yang dipergunakan dalam perancangan sistem.

2.3.1.2. Elemen dasar/simbol DFD

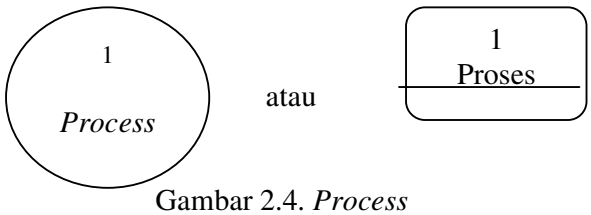
Sebelum menggambarkan DFD, sistem yang ada harus dipelajari terlebih dahulu termasuk aktivitas dan proses yang terjadi pada sistem tersebut. Dalam menganalisis *data flow* semua kejadian dievaluasi ke dalam bentuk *data flow* proses, tempat penyimpanan data (*data store*) serta asal dan tujuan data berupa arus data. DFD digambarkan secara bertingkat, mulai dari yang paling global (*Contex Diagram*) sampai ke tingkat yang lebih rinci : level-1, level-2, dst. Simbol-simbol yang digunakan dalam DFD terdiri dari 4 macam, yaitu : *process*,

data flow, *data store*, dan *external entity*. Berikut uraian singkat mengenai 4 macam simbol di atas :



Gambar 2.3. *External Entity*

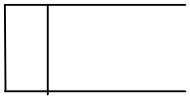
External Entity, adalah seseorang, sekelompok orang, sebuah departemen di dalam maupun di luar organisasi, atau sebuah sistem lain yang memberi masukan untuk sistem yang ada atau menerima keluaran dari suatu sistem. *External Entity* disebut juga *terminator*, karena merupakan batas dari sebuah sistem, berfungsi sebagai input dan output sistem, yang menggambarkan asal dan tujuan dari system tersebut. Dengan kata lain, *External Entity* merupakan simbol yang mewakili elemen yang berada di luar sistem, tetapi mememiliki hubungan interaksi dengan sistem. *External Entity* merupakan *entity* diluar sistem yang dapat merupakan organisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan *input* atau menerima *output* dari sistem. Suatu *External Entity* dapat disimbolkan dengan suatu notasi kotak.



Gambar 2.4. *Process*

Process, adalah simbol yang mengubah suatu data dari suatu bentuk menjadi bentuk yang lain. *Process* menerima masukan data dan mengeluarkan keluaran data lain yang telah diproses. Dengan kata lain, *process* merupakan simbol yang mewakili kegiatan untuk mengubah *input* menjadi *output*. *Process* melakukan suatu perubahan berdasarkan data yang dimasukkan dan menghasilkan data atau beberapa keterangan dari perubahan tersebut. Suatu *process* dapat

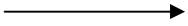
ditunjukkan dengan simbol empat persegi dengan sudut-sudut yang tumpul atau dapat juga disimbolkan dengan bentuk lingkaran.

Data store atau 

Gambar 2.5. *Data Store*

Data Store, merupakan tempat penyimpanan data dalam suatu sistem, baik secara manual maupun secara elektronik. Simpanan data digunakan jika suatu proses perlu menggunakan lagi data tersebut. *Data Store* simbol yang mewakili tempat penyimpanan dari data dan dapat dipergunakan bila dibutuhkan. *Data Store* dapat berupa :

- Suatu *file* atau *database* dari komputer.
- Suatu arsip atau catatan manual.
- Suatu tabel acuan manual.
- Suatu agenda atau buku.



Gambar 2.6. *Data Flow*

Data Flow, berfungsi untuk menggambarkan arah aliran data dari sistem. merupakan aliran yang menunjukkan perpindahan data dari satu bagian ke bagian yang lain dalam suatu sistem. *Data Flow* adalah simbol yang mewakili arah aliran data yang berasal dari satu atau beberapa proses, *data store* atau elemen lingkungan menuju ke satu atau beberapa proses, *data store* atau *enviromental element* lainnya. *Data Flow* dalam diagram diberi simbol anak panah. *Data Flow* sebaiknya diberi nama yang jelas dan mengandung arti. Nama pada *Data Flow* ditulis disamping atas.

Permodelan sistem dengan menggunakan DFD memiliki *level* yang menandai penggambaran sistem mulai dari umum hingga detail. *Level* yang menandai hubungan yang paling umum dan menggambarkan sistem secara keseluruhan disebut *context diagram*. Dari *context diagram* selanjutnya dapat diuraikan ke dalam DFD *level* 0, 1, hingga *level* dimana proses dianggap cukup detail dan jelas serta tidak perlu diuraikan lebih lanjut.

Dalam penggambaran proses, perlu diberikan penomoran, demikian juga dengan *level* uraian dari proses tersebut. Contohnya proses dengan nomor "1" pada DFD *level* "0" bila diuraikan pada DFD *level* "1", maka penomoran proses perlu didahului dengan nomor proses yang diuraikan, yaitu "1.1", "1.2", "1.3" dan seterusnya. Hal yang sama juga akan berlaku pada penomoran DFD *level* selanjutnya.

Context Diagram merupakan pokok sistem atau bahasa utama dari sistem yang akan dikembangkan. Contoh: Sistem Penggajian atau Sistem Pembayaran dengan komputerisasi. *Data Flow Diagram* ada 3 macam:

1. *Logical* DFD

   Penggambaran sistem hanya secara global (garis besarnya saja).
2. *Conceptual* DFD

   Penggambaran sistem cukup detail.
3. *Physical* DFD

   Penggambaran sistem paling detail, dimana tiap proses yang digambar dalam DFD sudah dalam bentuk proses atomik (sudah tidak dapat dipecah lagi).

2.3.2. *Entity Relationship Diagram* (ERD)

2.3.2.1. Definisi *Entity Relationship Diagram* (ERD)

Setelah membuat DFD baru membuat ERD dimana ERD ini nantinya akan dibuat relational tabelnya.

Entity Relationship Diagram (ERD) yaitu mendokumentasi data perusahaan dengan mengidentifikasi tipe-tipe data *entity* dan hubungan antar *entity-entity* tersebut (Raymond McLeod, 2000).

Entity Relationship Diagram (ERD) adalah diagram yang menggambarkan hubungan antara *entity key* yang satu dengan yang lain. (McLeod, 2000).

Entity Relationship Diagram (ERD) mendokumentasikan data perusahaan dengan mengidentifikasikan tipe dari data *entity-entity* dan *interrelationship* mereka. ERD dipersiapkan pada titik dalam sistem pengembangan proses yang merupakan gambaran dari data secara spesifik. Titik itu bisa berasal dari :

- Ketika *eksekutif* perusahaan ikut serta dalam *data modeling* untuk *enterprise*, mempertimbangkan kebutuhan data untuk seluruh perusahaan.
- Ketika *eksekutif* terlibat dalam *data modeling* untuk sebagian besar dari operasi perusahaan, seperti area bisnis.
- Ketika spesialis informasi dan pemakai terlibat dalam *data modeling* untuk area aplikasi.

Entity bisa berupa :

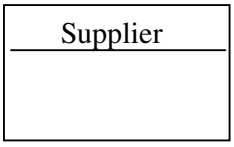
a. *Environmental element*
b. *Resource*
c. *Transaction*

Entity Relationship Diagram merupakan *data modeling* yang dipergunakan untuk mendokumentasikan data sebuah perusahaan dengan cara menentukan data apa saja yang terdapat dalam tiap *entity* dan bagaimana hubungan antara *entity* satu dengan lainnya.

Entity Relationship Diagram adalah permodelan yang menggambarkan data sebagai sekumpulan *entity, atribut*, dan *relationship*. ERD menggambarkan hubungan logical antar tabel, dan tidak ada hubungan dengan cara penyimpanan data secara fisik di dalam *database*. *Entity* adalah perwujudan dari suatu objek dalam dunia nyata yang berdiri sendiri. *Atribut* merupakan *properties* atau bagian dari suatu *entity*. *Relationship* menggambarkan hubungan antar *entity*. Untuk menghubungkan suatu *entity* dengan *entity* yang lain dalam *database* diperlukan *entity key*, yaitu satu atau beberapa atribut tertentu yang bersifat unik sehingga dapat digunakan untuk membedakan anggota *entity* yang satu dengan yang lainnya pada *entity* yang sama. Pada *diagram entity* juga diperlukan *relationship key*, yaitu setiap hubungan yang diperlukan untuk menyatakan hubungan antara *entity key* yang satu dengan yang lain.

2.3.2.2. Simbol ERD

Simbol yang dipergunakan dalam ERD adalah:



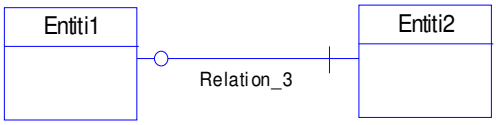
Gambar 2.7. *Entity*

Entity, yang termasuk ke dalam *entity* antara lain *environmental element*, *resource*, dan transaksi yang sangat penting bagi perusahaan. Tipe *entity* dapat berupa elemen lingkungan, sumber, atau transaksi yang penting bagi perusahaan untuk didokumentasikan dengan data. Contoh tipe *entity* adalah *Suppiler*. Tipe *entity* didokumentasikan dalam ERD dengan bentuk persegi. Setiap persegi diberi nama berdasarkan nama tipe *entity*, biasanya digunakan kata benda. Pemodelan *entity* tampak dalam Gambar 2.7.

2.3.2.3. Hubungan/relasi ERD

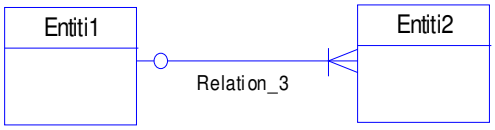
Selain istilah dan simbol di atas, terdapat juga istilah yang menggolongkan jenis relasi yang dilakukan antar *entity*, yaitu:

a. *Cardinality*, menandai jumlah *entity* yang muncul dalam relasi dengan *entity* lainnya. Nilai *cardinality* ada dua yaitu "1" atau "*many*". Bentuk relasi yang dapat dihasilkan ada tiga yaitu :
   - *One to one* (1:1), artinya semua bagian dari *entity* pada satu sisi berkoresponsi satu-satu dengan bagian dari *entity* pada sisi lain. Hubungan ini dibedakan menjadi dua macam, yaitu *non-obligatory* dan *obligatory*.
     Disebut *non-obligatory* bila semua anggota *entity* tidak harus mempunyai hubungan dengan anggota *entity* yang lain dan *obligatory* bila semua anggota dari suatu *entity* harus berpartisipasi atau mempunyai hubungan dengan *entity* yang lain. Gambar *one to one relationship* dapat dilihat pada Gambar 2.8. Pada Gambar 2.8. simbol *non-obligatory* ada di sebelah Entiti1, sedangkan simbol *obligatory* ada di sebelah Entiti2 yang mempunyai arti bahwa semua anggota Entiti1 harus berpartisipasi dengan Entiti2, sedangkan semua anggota Entiti2 tidak harus mempunyai hubungan dengan Entiti1.



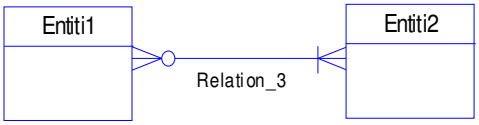
Gambar 2.8. *One to One Relationship*

- *One to many* (1:N), artinya setiap bagian dari *entity* pada derajat 1 berhubungan lebih dari satu bagian dari *entity* pada sisi N. Gambar *one to many relationship* dapat dilihat pada Gambar 2.9. Pada Gambar 2.9. simbol *non-obligatory* ada di sebelah Entiti1, sedangkan simbol *obligatory* ada di sebelah Entiti2 yang mempunyai arti bahwa semua anggota Entiti1 harus berpartisipasi dengan Entiti2, sedangkan semua anggota Entiti2 tidak harus mempunyai hubungan dengan Entiti1.



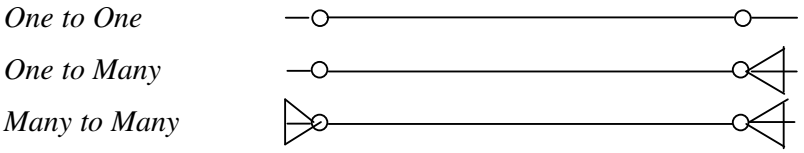
Gambar 2.9. *One to Many Relationship*

- *Many to many* (M:N), artinya setiap bagian dari *entity* dengan derajat M berhubungan dengan lebih dari satu bagian dari *entity* pada sisi N dan sebaliknya. Gambar *many to many relationship* dapat dilihat pada Gambar 2.10. Pada Gambar 2.10. simbol *non-obligatory* ada di sebelah Entiti1, sedangkan simbol *obligatory* ada di sebelah Entiti2 yang mempunyai arti bahwa semua anggota Entiti1 harus berpartisipasi dengan Entiti2, sedangkan semua anggota Entiti2 tidak harus mempunyai hubungan dengan Entiti1.



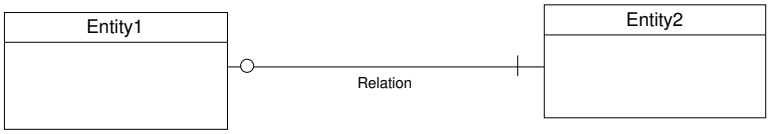
Gambar 2.10. *Many to Many Relationship*

Tipe garis dalam penggambaran untuk masing-masing jenis relasi adalah sebagai berikut:



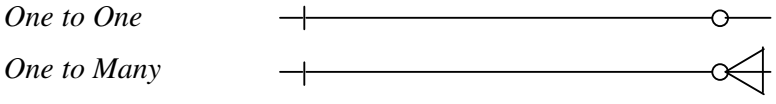
Gambar 2.11. Tipe garis berdasarkan tipe relasi

b. *Mandatory*, menandai apakah semua anggota *entity* harus berelasi dengan anggota *entity* lain atau tidak. Bila semua anggota harus berelasi maka diberi simbol "|" atau disebut juga *mandatory/obligatory* dan bila semua anggota tidak harus berelasi maka diberi simbol "o" atau disebut *non mandatory/non obligatory*. Contoh: *mandatory* Entity1 ke Entity2 dengan *one to one relationship*.



Gambar 2.12. *Mandatory*

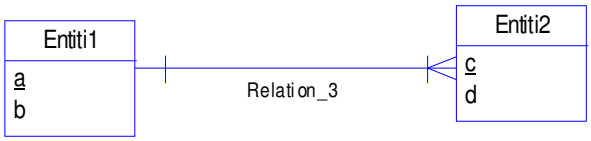
Dari Gambar 2.12. berarti semua anggota dari Entity1 harus berelasi dengan anggota dari Entity2, dengan memberi simbol "|" di depan Entity2. Sementara dari Entity2 terhadap Entity1 diberi simbol "o" di depan Entity1, berarti anggota dari Entity2 tidak harus berelasi seluruhnya dengan anggota dari Entity1. Untuk fungsi *mandatory* yang melambangkan bahwa *entity* tersebut harus mempunyai *item* terlebih dahulu baru *item* pada *entity* yang lain bisa diisi *item* dan dihubungkan. Tipe garisnya hampir sama, hanya saja tanda 'o' diganti dengan '|'.



Gambar 2.13. Tipe garis dengan *mandatory*

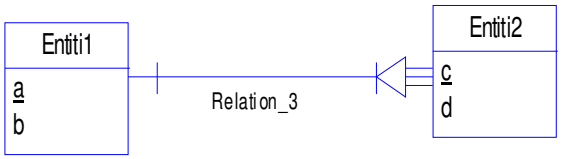
c. *Nonidentifying Relationships* dan *Identifying Relationships.*

- *Nonidentifying relationships* yaitu partisipasi antara *entity-entity* yang mempunyai *primary key* yang tidak bergantung atau dengan kata lain atribut *key* dari suatu *entity* tidak dibagi dengan *entity* lain sebagai *primary key*. Contoh gambar *conseptual* diagram-nya dapat dilihat pada Gambar 2.14.



Gambar 2.14. *Conseptual Diagram Nonidentifying Relationships*

- *Identifying relationships* yaitu *parent entity* menyumbangkan *entity primary key*-nya pada *child entity*. *Child entity* pada *identifying relationship* disebut sebagai *weak entity* karena identifikasinya bergantung pada keberadaan *parent entity*. Contoh gambar *conseptual diagram*-nya dapat dilihat pada Gambar 2.15.



Gambar 2.15. *Conseptual Diagram Identifying Relationships*

Dengan menggunakan metode hubungan *entity* dalam perencanaan *database*, dapat dilakukan beberapa langkah untuk membedakan hubungan antara *file* yang ada:

a. Menentukan *entity* yang ada dalam *database* dengan masing-masing atribut
b. Menentukan semua hubungan yang dapat terjadi antar *entity* yang ada.
c. Menentukan hubungan yang ada, termasuk *one to one*, *one to many* atau *many to many* juga menentukan adalah apakah hubungan tersebut *obligatory* atau *non obligatory.*
d. Dari jenis hubungan yang telah ditentukan tersebut, maka ditentukan jumlah relasi yang diperlukan.

Setelah tahap ini selesai dilaksanakan, maka telah tersedia suatu *database* yang telah didesain dengan baik dan siap digunakan dalam kondisi yang seefisien mungkin dan dapat memberikan informasi secara tepat dan benar.

2.3.2.4. Pemetaan ERD

Pemetaan ERD atau lebih dikenal dengan sebutan *relational mapping* adalah merubah ERD menjadi *relational database schema* sehingga menghasilkan *field-field* yang dapat diimplementasikan pada aplikasi *database*. Berikut ini adalah cara pemetaan ERD:

1. Untuk setiap *entity type* E yang ada pada ERD diciptakan sebuah relasi R yang terdiri dari semua *simple attribute* dari E. Untuk sebuah *composite attribute*, hanya memasukkan *simple component attribute*-nya. Pilih salah satu dari *key attribute* E sebagai *primary key* dari R
2. Untuk setiap *weak entity* W yang ada pada ERD dengan *owner entity type* E, buatlah sebuah relasi R yang menyertakan semua *simple attribute* W sebagai *attribute* dari R. *Primary key* dari R merupakan gabungan dari *primary key owner* dan *partial key* dari *weak entity type* W.
3. Untuk setiap 1:1 *relationship type* R, tentukan relasi S & T yang bertipe *total participation* (misalnya: S), maka *primary key* dari T ditambah ke S sebagai *foreign key*.
4. Untuk setiap 1:N *relationship type* R, identifikasikanlah hubungan S & T yang mewakili partisipasi dari *entity type* pada bagian N dari *relationship type*. Masukkan *primary key* dari T yang mewakili partisipasi *entity type* lainnya dalam R sebagai *foreign key* dalam S.
5. Untuk setiap M-N *relationship type* R, buatlah relasi S yang baru untuk mewakili R. Masukkan *primary key* dari relasi yang mewakili partisipasi *entity type* sebagai *foreign key* dalam S. Masukkan juga beberapa *simple attribute* dari M:N *relationship type* sebagai *attribute* dari S. Yang harus diperhatikan adalah tidak diperbolehkannya sebuah M:N *relationship type* dengan sebuah *foreign key* seperti pada relasi 1:1 atau 1:N.
6. Untuk setiap *multivalue attribute* A, buatlah satu relasi R yang baru dan masukkan sebuah *attribute* yang sesuai dengan A ditambah dengan *primary*

key K dari relasi yang menggambarkan *entity type* atau *relationship type,* dimana A sebagai *attribute*. *Primary key* K dari R adalah gabungan dari A & K. Jika *multivalue* adalah *composite*, masukkan *simple attribute*.

7. Untuk setiap N-any *relationship type*, dimana N>2, buat relasi S yang baru untuk menggambarkan R. Masukkan *primary key* dari relasi yang menggambarkan partisipasi *entity type* sebagai *attribute* dalam S. Masukkan juga beberapa *simple attribute* di n-any *relationship type* sebagai *attribute* di S. *Primary key* pada S biasanya merupakan gabungan di semua *foreign key* pada relasi yang menunjukkan partisipasi dari *entity type*.

2.4. Pembuatan Sistem

2.4.1. *Database*

2.4.1.1. Definisi *Database*

Database adalah jantung (inti) dari banyak informasi, yang merupakan kumpulan dari komputer data yang diintegrasikan, dan sistem informasi yang berbasis *web* juga tidak berbeda. (Stanek, 1996).

Database adalah kumpulan dari komputer data yang diintegrasikan, diorganisir, dan disimpan dengan cara yang mudah untuk diakses. Untuk ini digunakan *Direct Access Storage.* Integrasi secara logika dari *record* di beberapa *file* disebut konsep *database*.

Database yaitu digunakan untuk menyimpan data-data yang diperlukan. Setelah membuat DFD dan ERD baru membuat relational tabel untuk dipakai dalam pembuatan *database*.

Dua tujuan utama konsep *database* :

- Meminimalkan *redundancy* (data yang berlebihan).
- Memperoleh data *independen* merupakan data yang tidak bergantung pada data yang lain dan dapat berdiri sendiri.

Kemampuan untuk mengubah struktur data tanpa mengubah program untuk memproses data. Hal ini dapat dicapai dengan menempatkan data *specification* pada suatu tabel yang secara fisik terpisah dari programnya. Program

akan mereferensi tabel untuk mengakses data, perubahan struktur data hanya dilakukan sekali pada tabel.

2.4.1.2. Tipe *Database*

Tipe sistem *database* yang digunakan untuk menyimpan dan mengolah data, ada 2 macam, yaitu:

a. *Relational*

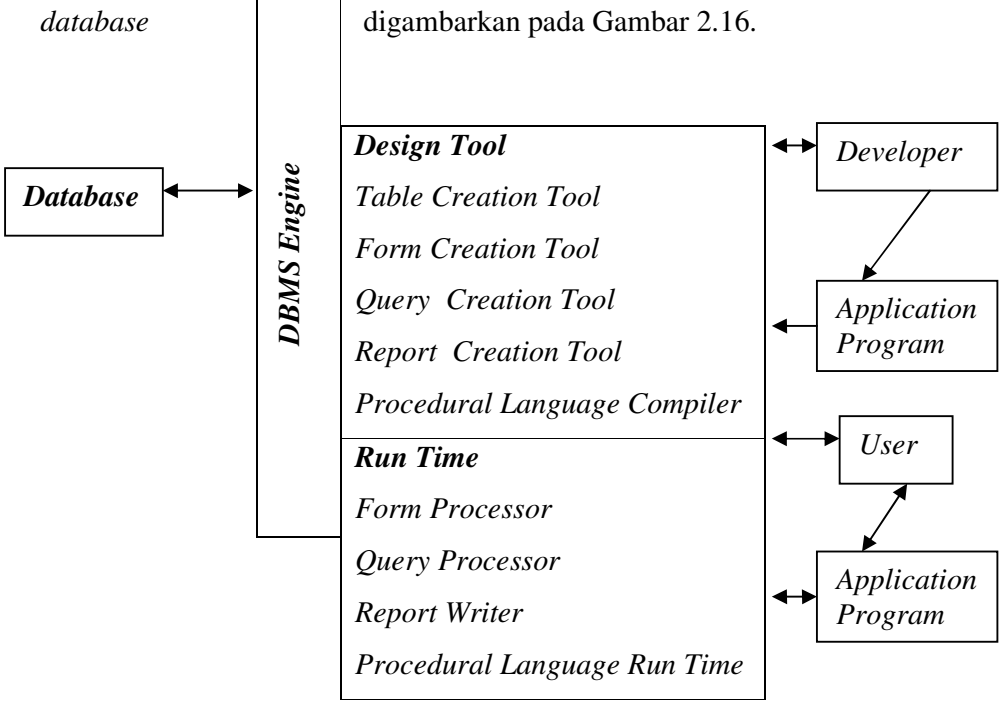
Menyimpan dan mengambil data didasarkan pada *attribute relationship*.

b. *Object-oriented*

Menyimpan dan mengambil data didasarkan pada *object-attributes*.

2.4.1.3. Komponen Sistem *Database*

Database diproses oleh DBMS, dimana digunakan oleh pengembang/*developer* dan *user*, baik langsung maupun tak langsung melalui *application program*. Untuk lebih jelasnya, komponen sistem *database* digambarkan pada Gambar 2.16.



Gambar 2.16. Komponen sistem *database*

Sumber: Tabel Komponen sistem *database*. (Kroenke, 1995).

2.4.2. *Database Management Systems* (DBMS)

Database ditujukan pada semua data yang tersimpan dengan menggunakan sumber daya komputer di dalam organisasi, sementara DBMS merupakan perangkat lunak aplikasi yang menyimpan struktur dari *database*, data, dan relasi data dalam *database* termasuk *form* dan laporan yang berhubungan dengan *database* (McLeod, Jr., Schell, 1979).

2.4.3. *Borland Delphi*

Merupakan perangkat pengembangan aplikasi yang sangat terkenal di lingkungan *windows*. Dengan perangkat ini dapat dibuat berbagai aplikasi *windows* mulai seperti permainan, *database*, multimedia dan masih banyak lagi. Dengan *Delphi* dapat dibuat aplikasi yang rumit dengan tidak terlalu banyak menuliskan kode karena sifatnya yang visual. *Delphi* menggunakan objek *pascal* sebagai dasar bahasa pemrograman.

Beberapa istilah yang dipergunakan dalam *Delphi* adalah:

- Aplikasi atau program, adalah sederetan kode yang digunakan untuk mengatur komputer agar dapat berjalan sesuai dengan keinginan pembuat program. Aplikasi dibedakan menjadi aplikasi *windows* dan aplikasi konsol. Aplikasi *windows* adalah aplikasi yang berjalan pada *windows* sedangkan aplikasi yang tidak berjalan pada *windows* contohnya DOS.
- *Form*, merupakan tampilan berbentuk jendela pada *windows* dengan menggunakan *form*, pemakai dapat melakukan interaksi dengan komputer.
- Proyek, merupakan tempat peletakan dari *Delphi* dalam aplikasi. Sebuah proyek membawahi sejumlah *form.*
- Komponen, merupakan alat yang telah disediakan *Delphi* untuk mempermudah pemrograman. *Form* juga merupakan salah satu komponen *Delphi*. Selain *form*, terdapat komponen lain seperti *edit text*, *button, ttime, memo* dan puluhan komponen lainnya. Komponen ada yang tampak dan ada yang tidak tampak dalam program. Komponen yang tampak disebut juga *control*.

Integrated Development Environment (IDE), merupakan bagian dari *Delphi* yang memfasilitasi rancangan tampilan visual untuk pemakai dan menuliskan kode. Bagian-bagian dari IDE antara lain: Menu utama, *speed bar*, jendela *Form*, *Object inspector*, dan *Componen Pallete*.

2.5. Perencanaan Sistem

Perencanaan suatu sistem *database* dimana proses yang terjadi haruslah stabil kecepatannya sebanding dengan banyaknya data. Maka dari itu, apabila perencanaannya salah maka akan berakibat pada lambannya akses data.

- Efisiensi *Program*

  Program hendaknya dibuat dengan seefisien mungkin dan *user friendly* terhadap pemakaiannya sehingga mempermudah pekerjaan dan kecepatan kerja dapat lebih optimal.

Dengan rancangan sistem *database* yang baik, maka pengembangan *software* dengan menggunakan *database* dapat lebih mudah dilakukan dan diorganisir. Selain itu, *software* aplikasi *database* yang dibuat dengan rancangan yang baik dapat memecahkan masalah pemakaian *software* yang tepat.

Tujuan dari pembuatan *software* aplikasi *database* adalah membuat pemakai *software* tersebut lebih efisien, dengan *software* yang telah dibuat, maka pemakai komputer dapat mempermudah pekerjaan, serta mendapatkan fasilitas-fasilitas yang sangat membantu pekerjaannya.

2.5.1. Analisis Data

Informasi berupa data-data yang terkumpul dari hasil *survey* dan pengumpulan data yang telah dilakukan pada perusahaan, adalah data yang detail bagaimana sistem tersebut dapat memenuhi kebutuhan pemakai. Setelah semua informasi diperlukan untuk merancang sistem diperoleh, informasi atau data-data tersebut dianalisis agar rancangan sistem yang dibuat dapat memberikan hasil yang semaksimal mungkin.

2.5.2. Perancangan *Database*

Pada tahap ini perancangan sistem difokuskan pada perangkat lunak dari sistem tersebut. Perancangan *database* lebih menjurus pada perancangan tabel-tabel apa yang akan dibuat, *field* dari tiap tabel tabel dan relasi antara satu tabel dengan tabel yang lain. Untuk perancangan *database* dapat dilakukan dengan beberapa metode:

a. *Entity Relationship Diagram* (ERD)

   Teknik perancangan ini bertujuan untuk membuat relasi antar *entity-entity* yang ada.

b. *Functional Dependencies* (FD)

   Teknik dari perancangan FD adalah dengan membuat relasi antara *field-field* secara langsung dan kemudian dilakukan penyederhanaan relasi untuk membuang relasi yang bersifat *redudant*.

c. *Data Flow Diagram* (DFD)

   Teknik dari DFD adalah dengan memberikan gambaran relasi-relasi yang ada, kemudian dengan aturan-aturan tertentu dapat menyederhanakan relasi tersebut.

2.5.3. Perancangan Struktur Data

Tahap ini akan mengatur bagaimana suatu *database* dapat menyimpan data pada masing masing tabel. Pada setiap tabel akan memiliki beberapa *field*, setiap *field* memiliki struktur data yang berbeda sesuai dengan kebutuhannya. Pada tahap perancangan struktur data, akan ditentukan jenis data yang akan digunakan pada tiap-tiap *field*, misalkan berupa numerik, karakter, *integer*, tanggal atau *currency.*

Besarnya kapasitas data yang akan diinputkan dalam hal ini tidak berpengaruh besar, karena data dapat terus ditambah tanpa khawatir kekurangan tempat dan *database* akan dapat menyesuaikan tempat secara otomatis sejumlah data yang diinputkan.yang baik dipengaruhi oleh banyak faktor, antara lain:

- Kehandalan Sistem

  Sistem *database* harus dipergunakan untuk menyimpan data dalam jumlah yang tak terbatas dan dapat dipakai dalam jangka waktu yang lama.

2.5.4. Perancangan *Input/Output* dan *User Interface*

Proses *input* data dilakukan dengan mengisi kotak *input*an yang telah disediakan berdasarkan nama *field* pada suatu tabel. Untuk memproses *input*an dapat dilakukan dengan penekanan tombol *keyboard* maupun *mouse*. Sedangkan *output*nya akan ditampilkan langsung ke monitor.

2.5.5. Implementasi Sistem Pada Bahasa Pemrograman

Bahasa pemrograman yang digunakan dalam membuat *program database* ini digunakan *Borland Delphi* 7 dengan menggunakan *database SQL Server 2000*, *program* ini dapat dijalankan pada sistem operasi *Windows XP Profesional.*

2.5.6. Uji Coba Sistem

Sebelum *program* yang dibuat layak pakai, maka perlu diadakan tahapan uji coba untuk mengetahui apakah sistem tersebut sudah layak pakai dalam arti tidak mempunyai kesalahan dalam proses pengisian *database* dan dalam melakukan kalkulasi. Dengan adanya ujicoba ini, juga dapat diketahui apakah *program* sudah berjalan sesuai dengan permintaan pemakai, maka dari itu *program database* ini dijalankan seakan-akan seperti dipakai sesungguhnya dalam kegiatan sehari-hari, dan dengan uji coba ini juga seringkali ditemukan kesalahan yang ditemukan oleh pemakai untuk kemudian diperbaiki. Dengan adanya uji coba sistem ini, maka kesalahan fatal dapat dihindari.

2.5.7. Pelatihan *Operator*

Setelah dilakukan uji coba berulang-ulang, maka kesalahan yang ditemukan segera dapat diperbaiki sehingga *program* dapat dipakai sesungguhnya dalam kegiatan sehari-hari. Supaya *operator* dapat menggunakan *program* ini dengan baik dan dapat mempelajarinya dengan cepat, maka akan dilakukan pelatihan terhadap *operator* yang bersangkutan. Dengan adanya pelatihan ini, maka *operator* dapat menggunakan *program* dengan benar dan dapat memaksimalkan kerjanya.