2. LANDASAN TEORI

2.1 Sistem Informasi Manajemen

Sistem informasi manajemen adalah suatu sistem pada tingkat manajemen yang dibuat oleh manusia yang terdiri dari komponen-komponen untuk menyajikan informasi yang tepat, akurat, ekonomis, dan relevan. Oleh karena itu, untuk mencapai tujuan yang diinginkan, maka dilakukan serangkaian kegiatan sistem informasi. Kegiatan-kegiatan yang dilakukan dalam sebuah sistem informasi:

- *Input*

  Elemen-elemen yang masuk ke dalam suatu sistem dan siap diolah atau diproses
- Proses

  Pengolahan *input* menjadi *output* yang diinginkan
- *Output*

  Hasil yang diperoleh yang dapat dimanfaatkan dari *output* sistem untuk mencapai tujuan yang diinginkan
- *Feedback*(Umpan balik)

  Informasi tentang pelaksanaan sistem tersebut
- Pengendalian

  Suatu kegiatan untuk menjain bahasa informasi yang disajikan sesuai dengan yang diharapkan

2.2 Metode FIFO

Metode FIFO mengasumsikan bahwa barang yang dibeli lebih awal adalah barang pertama yang dijual. Dengan metode FIFO biaya untuk pembelian barang yang dibeli lebih dahulu dianggap akan menjadi harga pokok penjualan. Pada metode FIFO, persediaan akhir ditentukan dengan mengambil harga perolehan per unit dari pembelian paling akhir dan bergerak mundur sampai semua unit dalam persediaan mendapat harga perolehan.

Pool of Cost

Cost of Goods Available for Sale

Date	Explanation	Units	Unit Cost	Total Cost
01/01	Begining Inventory	100	$10	$1,000
04/15	Purchase	200	11	2,200
08/24	Purchase	300	12	3,600
11/27	Purchase	400	13	5,200
	Total	1,000		12,000

Step 1
Ending Inventory

Date	Unit	Unit Cost	Total Cost
11/27	400	$13	$5,200
08/24	50	12	600
Total	450		5,600

Step 2
Cost of Goods Sold

Cost of goods available for sale	12,000
Less: Ending Inventory	5,800
Cost of Goods Sold	6200

2.3 *Structured Query Language (SQL)*

Menurut Soetam Rizky (2004), *Structured Query Language* (SQL) adalah sebuah bahasa yang dipergunakan untuk mengakses data dalam basis data relasional. Beberapa poin penting mengenai SQL *Server* :

- Merupakan sebuah *Relational Database Management System* (RDBMS)
- *Database* bertugas melayani permintaan *client* atau proses untuk menggunakan sumber daya *database* seperti data, memori dan proses *server.*
- Sebuah *Database Server* bersifat *multiuser* : beberapa *user* dapat melakukan koneksi secara bersamaan melalui jaringan ke *database server*, di mana nantinya :
  - Semua *database* akan disimpan ke *server*
  - Setiap *user* dapat mengirimkan permintaan ke *server* untuk mengambil data, mengubah, atau menghapusnya dengan mengirimkan perintah SQL dan *server* akan melayani permintaan tersebut.

Beberapa elemen dari *System Management Database Relational* seperti dijelaskan sebagai berikut :

- *Database* – sekumpulan data yang berisi informasi dan saling berhubungan.
- Tabel – kumpulan dari *item-item* data yang diorganisir menjadi baris dan kolom yang memiliki subjek yang sama, merupakan salah satu komponen penting dari suatu *database.*
- *Record* (*Tuple*) – kumpulan data yang terdiri dari satu atau lebih suatu *field.*
- *Field (Attribute)* – kumpulan data yang mempunyai / menyimpan fakta yang sama/sejenis untuk setiap baris pada tabel.
- *Query* – sekumpulan perintah SQL (*Structured Query Language*) yang dirancang untuk memanggil kelompok record tertentu dari satu tabel atau lebih untuk melakukan operasi pada tabel.

Secara umum, SQL terdiri dari dua bahasa, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML). Implementasi DDL dan DML berbeda untuk tiap *sistem manajemen basis data* (SMBD), namun secara umum implementasi tiap bahasa ini memiliki bentuk standar yang ditetapkan ANSI. Artikel ini menggunakan bentuk paling umum yang dapat digunakan pada kebanyakan SMBD.

2.3.1 *Data Definition Language (DDL)*

Bagian dari *database management system* yang dipakai untuk mendefinisikan dan mengatur semua atribut dan *property* dari sebuah *database*. DDL mengijinkan spesifikasi tidak hanya kumpulan dari tabel-tabel tetapi juga informasi tentang tabelnya yang terdiri atas:

- Skema untuk setiap tabel.
- Nilai domain yang berhubungan dengan setiap atribut.
- Integritas *constraint*.
- Keamanan dan autorisasi informasi dari setiap tabel.
- Struktur penyimpanan secara fisik dari setiap tabel.

Bentuk umum dari DDL :

a. CREATE <Nama objek>

Digunakan untuk membuat suatu *database, table, view, index, procedure, trigger* dan sebagainya. Contoh :

```
CREATE TABLE [nama tabel]
(
        [nama field] [tipe data] [constraint],
        [nama field] [tipe data],
        ... ,
        [nama field] [tipe data]
)

CREATE TABLE Pembeli
(
        ID_Pembeli varchar(8),
        Nama_Pembeli varchar(20),
        ... ,
        [nama field] [tipe data]
)
```

Macam – macam *constraint* yang biasa digunakan, yaitu:

- ***Primary Key*** – dengan tujuan untuk menjaga integritas data, bahwa sebuah *Primary Key* tidak boleh mempunyai duplikat dan secara otomatis tidak *"Null"*.
- ***Foreign Key*** – *field* pada sebuah tabel yang menunjukkan bahwa *field* tersebut adalah *Primary Key* pada tabel yang lain.
- ***Check*** - dipakai untuk menjamin integritas domain dengan cara membatasi nilai-nilai yang dapat diterima sebuah kolom.
- ***Unique*** –hampir sama dengan *constraint primary key*, dimana keduanya digunakan untuk menerapkan integritas entitas. Perbedaannya adalah *field* boleh *"Null"* dan boleh lebih dari 1 dalam satu tabel.
- ***Not Null*** – *field* yang bersangkutan tidak boleh bernilai kosong.

b. ALTER <nama objek>

Digunakan untuk mengubah atribut pada tabel yang sudah ada. Semua data dari penambahan atribut baru akan diisi *null*.

- **ALTER TABLE** nama_table [**add/drop/alter column/constraint** nama_tabel/nama_constraint]
- **ALTER TABLE** TMahasiswa **DROP COLUMN** alamat_mahasiswa

c. DROP <nama objek>

Digunakan untuk menghapus suatu objek. Menghapus *database* dilakukan dengan perintah:

- **DROP DATABASE** [nama database]
- **DROP DATABASE** TugasAkhir

Menghapus tabel dilakukan dengan menggunakan *query*:

- **DROP TABLE** [nama tabel]
- **DROP TABLE** TMahasiswa

2.3.2 *Data Manipulation Language (DML)*

Merupakan perintah – perintah yang digunakan untuk menampilkan, menambah, mengubah dan menghapus dalam objek–objek yang didefinisikan oleh DDL.

- **INSERT**

  Untuk menambah satu atau beberapa data di dalam suatu tabel:

  INSERT INTO [nama_tabel] ([daftar_field]) **VALUES** ([daftar_nilai])

  INSERT INTO test(nama,alamat,password) **VALUES (**’test’,’alamat’,’pass’);

- **UPDATE**

  Untuk mengubah data pada satu baris, beberapa baris atau semua baris dalam table atau view. Perintah **Set** menyatakan kolom yang akan digunakan dan nilai-nilai barunya:

  UPDATE [nama_tabel] **SET** [nama_field]=[nilai]

  UPDATE test **SET** pass = ‘deny’ **WHERE** nama=’denygunawan’

- **DELETE**

  Untuk menghapus satu atau beberapa baris dari sebuah tabel.

  DELETE FROM [nama_tabel]

  DELETE FROM test **WHERE** nama='denygunawan'

- **SELECT**

  Untuk menampilkan isi dari tabel pada *SQL Server* diperlukan perintah *select*.

  SELECT daftar_select
  FROM daftar_tabel
  [WHERE kondisi_pencarian**]**
  [GROUP BY daftar_group_by**]**
  [HAVING kondisi_pencarian**]**
  [ORDER BY daftar_order**[ASC|DEC]]**

  SELECT * **FROM** TMahasiswa
  SELECT nama **FROM** TMahasiswa **WHERE** ID_mahasiswa=06105

Distinct - Digunakan untuk menampilkan data kembar sebanyak 1 kali.

Where - Untuk menampilkan data yang memenuhi kriteria atau kondisi tertentu.

Order By - Untuk menampilkan data secara berkelompok. Ada 2 macam, yaitu ASC (*ascending*) dan DESC (*descending*).

Kondisi dapat dihubungkan dengan operator logika, misalnya AND dan OR.

Contoh:

Diasumsikan terdapat tabel user yang berisi data sebagai berikut.

username	**passwd**	**tanggal_lahir**	**jml_transaksi**	**total_transaksi**
Aris	6487AD5EF	09-09-1987	6	10.000
Budi	97AD4erD	01-01-1994	0	0
Charlie	548794654	06-12-1965	24	312.150
Daniel	FLKH947HF	24-04-1980	3	0

Erik 94RER54 17-08-1945 34 50.000

Contoh 1: Tampilkan seluruh data.

```
SELECT *
FROM user
```

Contoh 2: Tampilkan pengguna yang tidak pernah bertransaksi.

```
SELECT *
FROM user
WHERE total_transaksi = 0
```

Contoh 3: Tampilkan username pengguna yang bertransaksi kurang dari 10 dan nilainya lebih dari 1.000.

```
SELECT username
FROM user
WHERE jml_transaksi < 10 AND total_transaksi > 1000
```

Contoh 4: Tampilkan total nominal transaksi yang sudah terjadi.

```
SELECT SUM(total_transaksi) AS total_nominal_transaksi
FROM user
```

Contoh 5: Tampilkan seluruh data diurutkan berdasarkan jumlah transaksi terbesar ke terkecil.

```
SELECT *
FROM user
ORDER BY jml_transaksi DESC
```

2.3.3 Fungsi aggregat

Digunakan pada bagian SELECT. Syarat untuk fungsi aggregat diletakkan pada bagian **HAVING**, bukan WHERE.

Beberapa SMBD memiliki fungsi aggregat, yaitu fungsi-fungsi khusus yang melibatkan sekelompok data (aggregat). Secara umum fungsi aggregat adalah:

- SUM untuk menghitung total nominal data.
- COUNT untuk menghitung jumlah kemunculan data.
- AVG untuk menghitung rata-rata sekelompok data.
- MAX dan MIN untuk mendapatkan nilai maksimum/minimum dari sekelompok data.

2.3.4 *Subquery*

Ada kalanya *query* dapat menjadi kompleks, terutama jika melibatkan lebih dari satu tabel dan/atau fungsi aggregat. Beberapa SMBD mengizinkan penggunaan *subquery*. Contoh:

Tampilkan username pengguna yang memiliki jumlah transaksi terbesar.

```
SELECT username
FROM user
WHERE jml_transaksi =
(
SELECT MAX(jml_transaksi)
FROM user
)
```

2.4 *SQL Server*

Microsoft SQL Server adalah sebuah sistem manajemen basis data relasional (RDBMS) produk Microsoft. Bahasa *query* utamanya adalah Transact-SQL yang merupakan implementasi dari SQL standar ANSI/ISO yang digunakan oleh Microsoft dan Sybase. Umumnya SQL Server digunakan di dunia bisnis yang memiliki basis data berskala kecil sampai dengan menengah, tetapi kemudian berkembang dengan digunakannya SQL Server pada basis data besar.

Microsoft SQL Server dan Sybase/ASE dapat berkomunikasi lewat jaringan dengan menggunakan protokol TDS (Tabular Data Stream). Selain dari itu, Microsoft SQL Server juga mendukung ODBC (Open Database Connectivity), dan mempunyai *driver* JDBC untuk bahasa pemrograman Java. Fitur yang lain dari SQL Server ini adalah kemampuannya untuk membuat basis data *mirroring* dan *clustering*. Sumber diambil dari *website* :

2.5 *Entity Relationship Diagram (ERD)*

Menurut Roomney (2003) , definisi *Entity Relationship Diagram* (ERD) adalah *"A graphical technique for portraying a database schema"*. Diagram Hubungan Entitas atau *entity relationship diagram* merupakan model data berupa notasi grafis dalam pemodelan data konseptual yang menggambarkan hubungan antara penyimpan. Model data sendiri merupakan sekumpulan cara, peralatan

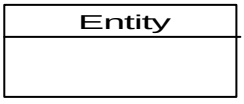
untuk mendeskripsikan data-data yang hubungannya satu sama lain, serta batasan konsistensi. Model data terdiri dari model hubungan entitas dan model relasional. Diagram hubungan entitas ditemukan oleh Peter Chen dalam buku Entity Relational Model-Toward a Unified of Data. Chen mencoba merumuskan dasar-dasar model dan setelah itu dikembangkan dan dimodifikasi oleh Chen dan banyak pakar lainnya. Pada saat itu diagram hubungan entitas dibuat sebagai bagian dari perangkat lunak yang juga merupakan modifikasi khusus, karena tidak ada bentuk tunggal dan standar dari diagram hubungan entitas.

2.5.1 Kegunaan

Diagram hubungan entitas digunakan untuk mengkonstruksikan model data konseptual, memodelkan struktur data dan hubungan antar data dan mengimplementasikan basis data secara logika maupun secara fisik dengan DBMS (Database Management system). Dengan diagram hubungan entitas ini dapat diuji model dengan mengabaikan proses yang harus dilakukan. Diagram hubungan entitas dapat membantu dalam menjawab persoalan tentang data yang diperlukan dan bagaimana data tersebut saling berhubungan.

2.5.2 Entitas

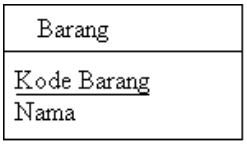
Entitas adalah suatu objek yang dapat didefinisikan dalam lingkungan pemakai, sesuatu yang penting bagi pemakai dalam konteks sistem yang akan dibuat. Sebagai contoh pelanggan, pegawai dll. Seandainya A adalah seorang pegawai maka A adalah isi dari pegawai, sedangkan jika B adalah seorang pelanggan maka B adalah isi dari pelanggan. Harus dibedakan entitas sebagai bentuk umum dari deskripsi tertentu dan isi entitas seperti A dan B dalam contoh di atas. Gambar simbol entitas dapat dilihat pada Gambar 2.1. Sumber diambil dari *website* : *id.wikipedia.org/wiki/Diagram_hubungan_entitas.*



Gambar 2.1. Simbol *entity*

2.5.3 Atribut

Entitas mempunyai elemen yang disebut atribut, dan berfungsi mendekripsikan karakter dari entitas. Atribut adalah properti atau karakteristik yang dimiliki oleh suatu entitas dimana properti atau karakteristik itu bermakna atau berarti bagi organisasi atau perusahaan, misalnya untuk pencatatan data pegawai di suatu instansi, entitas pegawai mungkin memiliki atribut-atribut nomor induk pegawai, nama, alamat, nomor telepon, gaji pokok dan lainnya. Setiap diagram hubungan entitas bisa terdapat lebih dari satu atribut. Entitas memiliki himpunan atribut yang berasosiasi dengannya. Gambar simbol atribut dapat dilihat pada Gambar 2.2.



Gambar 2.2. Simbol *attribute*

2.5.3.1 Macam-Macam Atribut

Atribut terdiri dari atribut sederhana atau atormis, atribut komposit, atribut berharga tunggal. atribut *null-value*, atribut kunci, atribut bernilai banyak dan atribut turunan. Masing-masing atribut memiliki ciri tersendiri. Atribut atormis tidak dapat dibagi-bagi menjadi atribut yang sederhana.

Atribut komposit adalah atribut yang dapat dipecah menjadi atribut lain, misalnya atribut alamat dapat dipecah menjadi atribut jalan, kecamatan, kelurahan,kota serta kode pos. atribut komposit digunakan pada *database* untuk kemudahan menjawab pertanyaan-pertanyaan tertentu dalam *database* atribut berharga tunggal mempunyai satu harga untuk entitas tertentu, atribut *null-value* tidak mempunyai nilai, atribut kunci merupakan atribut unik dari suatu entitas dan nilai dari atribut kunci akan berbeda untuk masing-masing entitas.atribut bernilai banyak adalah atribut yang entitasnya lebih dari satu, misalnya adalah atribut hobi. Atribut hobi ini bisa terdiri dari atribut berenang, atribut voli dan atribut berbelanja.atribut turunan merupakan atribut yang didapat dari atribut lainnya. Pada entitas pegawai terdapat atribut nomor induk yang biasanya terkandung nilai

tahun masuk, misalnya NIP =5195025, berarti pegawai yang bersangkutan masuk pada tahun 1995, maka jika ditambahkan atribut Lama_Kerja pada entitas Pegawai, atribut Lama_Kerja dapat dihitung dengan cara mengurangkan tahun dimana perhitungan dilakukan (misalnya 2005) dengan tahun mahasiswa yang bersangkutan masuk ke Instansi (Hasilnya 10 tahun).

2.5.4 Relasi

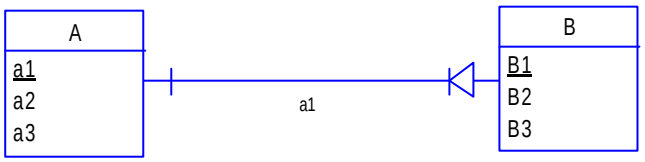
Relasi adalah hubungan *entity* yang satu dengan *entity* yang lain. Dalam relasi dikenal tiga istilah, yaitu :

1. *Cardinality*

Jumlah maksimum atau minimum dari elemen yang diizinkan pada setiap sisi dari suatu *relationship*.

2. *Dependency*

Ketergantungan sebuah *entity* dengan entity induknya. Sehingga jika *entity* induknya dihapus, maka *entity* anaknya akan ikut terhapus secara otomatis. Gambar relasi bersifat *dependency* dapat dilihat pada Gambar 2.3.



Gambar 2.3. Relasi bersifat *dependency*

3. *Mandatory*

Menandai apakah semua *record* dari sebuah *entity* harus berelasi dengan *record* dari *entity* yang lain.

Ada beberapa macam *cardinality,* yaitu :

1. *One-to-one*

Menghubungkan antara sebuah entity dengan sebuah *entity* lain yang berbeda. Gambar *cardinality One-to-one* dapat dilihat pada Gambar 2.4.

Gambar 2.4. Relasi *one to one*

2. One-to-many

Menghubungkan antara satu anggota *entity* dengan beberapa anggota *entity* lain yang berbeda. Gambar relasi *one-to-many* dapat dilihat pada Gambar 2.5.

Gambar 2.5. Relasi *one-to-many*

3. Many-to-many

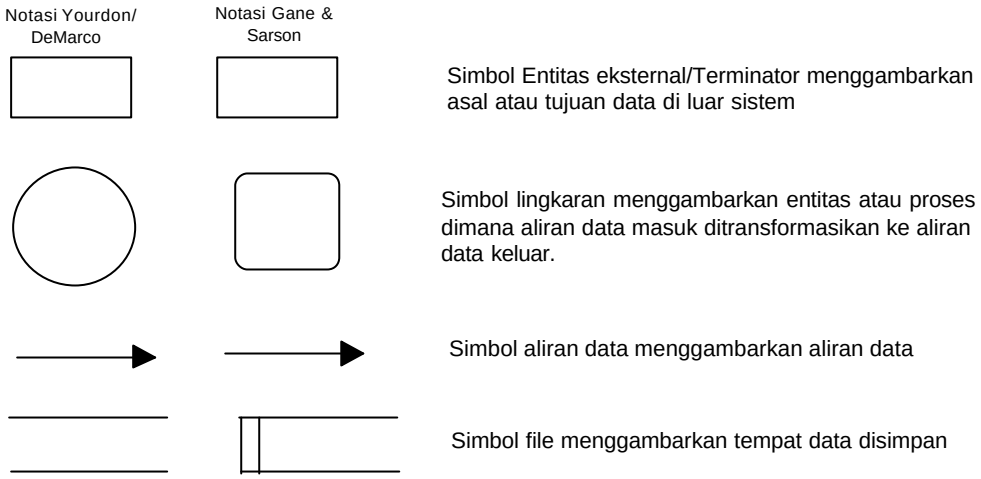
Menghubungkan beberapa *entity* dengan beberapa anggota *entity* lain yang berbeda. Gambar relasi *many-to-many* dapat dilihat pada Gambar 2.6.

Gambar 2.6. Relasi *many-to-many*

2.6 *Data Flow Diagram* (DFD)

Data Flow Diagram (DFD) adalah representasi grafik dari sebuah sistem. DFD menggambarkan komponen-komponen sebuah sistem, aliran-aliran data di mana komponen-komponen tersebut, dan asal, tujuan, dan penyimpanan dari data tersebut.

DFD dapat digunakan untuk membuat dokumentasi dari sistem informasi yang ada. Empat simbol yang digunakan dapat dilihat pada Gambar 2.7.



Gambar 2.7 Simbol-simbol pada DFD

Context Diagram adalah DFD yang menunjukkan batas-batas dari sebuah system informasi, yaitu top level view dari sebuah sistem. Untuk menggambar *Context Diagram*, hanya dibuat satu proses saja yang menggambarkan keseluruhan proses dari sistem tersebut dan beberapa kesatuan luar di sekelilingnya yang berhubungan. Contoh Context Diagram untuk DFD di atas adalah sebagai berikut:

- Dari DFD *level* 0, terkadang ada proses yang dapat dijabarkan lebih detail lagi.
- Dari DFD *level* 0 dapat dibuat sejumlah DFD *level* 1.

Demikian pula dari setiap *level* 1 tersebut jika masih dapat dijabarkan lagi menjadi DFD *level* 2,3 dan seterusnya. Sebuah DFD minimal digambarkan sampai level 0 saja, namun dalam kenyataan sebuah DFD umumnya pasti terdiri dari beberapa level. Setiap simpanan data hanya boleh menerima input dari proses dan memberikan output ke proses saja.

Data Flow Diagram adalah alat bantu berbentuk grafik yang berfungsi untuk menunjukkan aliran data dalam sebuah sistem.

2.7 Program Aplikasi Penunjang

Dalam menyelesaikan skripsi sistem informasi ini, penulis menggunakan bahasa pemrograman Microsoft Visual Basic 2005 sebagai bahasa *coding* dan *design interface* serta Microsoft SQL Server 2005 untuk menyimpan *database*.

2.8 Microsoft Visual Basic 2005

Microsoft Visual Basic 2005 adalah sebuah alat untuk mengembangkan dan membangun aplikasi yang bergerak di atas sistem .NET Framework, dengan menggunakan bahasa BASIC. Dengan menggunakan alat ini, para *programmer* dapat membangun aplikasi windows forms, aplikasi web berbasis ASP.NET, dan juga aplikasi *command-line.* Alat ini dapat diperoleh secara terpisah dari beberapa produk lainnya (seperti Microsoft Visual C++, Visual C#, atau Visual J#), atau juga dapat diperoleh secara terpadu dalam Microsoft Visual Studio .NET. Bahasa Visual Basic .NET sendiri menganut paradigma bahasa pemrograman berorientasi objek yang dapat dilihat sebagai evolusi dari Microsoft Visual Basic versi sebelumnya yang diimplementasikan di atas .NET Framework. Peluncurannya mengundang kontroversi, mengingat banyak sekali perubahan yang dilakukan oleh Microsoft, dan versi baru ini tidak kompatibel dengan versi terdahulu.