

BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi Akuntansi

Sistem informasi akuntansi adalah kumpulan sumber daya, 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 juga merupakan sistem yang paling penting di organisasi dan merubah cara menangkap, memproses, menyimpan, dan mendistribusikan informasi (Xu, 2009). Saat ini, digital dan informasi *online* semakin digunakan dalam sistem informasi akuntansi. Organisasi perlu menempatkan sistem di lini depan, dan mempertimbangkan baik segi sistem ataupun manusia sebagai faktor yang terkait ketika mengatur sistem informasi akuntansi.

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 sumber daya menjadi barang dan jasa.

4. Siklus keuangan. Kejadian-kejadian yang berkaitan dengan perolehan dan manajemen dana-dana modal, termasuk kas. (Agustinus Mujilan, 2012, 3).

II.1.1. Manfaat Dan Tujuan Sistem Informasi Akuntansi

Adapun tujuan dan manfaat Sistem Informasi Akuntansi adalah sebagai berikut :

1. Mengamankan harta / kekayaan perusahaan, Harta / kekayaan di sini meliputi kas perusahaan, persediaan barang dagangan, termasuk aset tetap perusahaan.
2. Menghasilkan beragam informasi untuk pengambilan keputusan. misal, pengelola toko swalayan memerlukan informasi mengenai barang apa saja yang diminati oleh konsumen. Membeli barang yang kurang laku berarti kas akan terjebak dalam persediaan dan berarti kehilangan kesempatan untuk membeli barang dagangan yang laku.
3. Menghasilkan informasi untuk pihak eksternal. Setiap pengelola usaha memiliki kewajiban untuk membayar pajak. Besarnya pajak yang dibayar tergantung pada omset penjualan (jika pengelola memilih menggunakan norma dalam perhitungan pajaknya) atau tergantung pada laba rugi usaha (jika pengelola memilih untuk tidak menggunakan norma dalam perhitungan pajaknya).
4. Menghasilkan informasi untuk penilaian kinerja karyawan atau divisi. Sistem informasi dapat juga dimanfaatkan untuk penilaian kinerja karyawan atau divisi.

5. Menyediakan data masa lalu untuk kepentingan audit (pemeriksaan). Data yang tersimpan dengan baik sangat memudahkan proses audit (pemeriksaan).
6. Menghasilkan informasi untuk penyusunan dan evaluasi anggaran perusahaan. Anggaran merupakan alat yang sering digunakan perusahaan untuk mengendalikan pengeluaran kas.
7. Menghasilkan informasi yang diperlukan dalam kegiatan perencanaan dan pengendalian. Selain berguna untuk membandingkan informasi yang berkaitan dengan anggaran dan biaya standar dengan kenyataan seperti yang telah dikemukakan (Anastasia Diana ; 2011 : 6).

II.1.2. Kualitas Data Sistem Informasi Akuntansi

Seperti telah disebutkan di atas bahwa kualitas informasi merupakan hal yang sangat penting dalam sistem informasi (Xu, 2009). Informasi yang baik akan tergantung pula dengan data yang baik. Untuk membentuk sistem yang mampu menyediakan data dan informasi yang baik dan dapat dipercaya membutuhkan suatu usaha yang tidak mudah.

Xu (2009) menggaris bawahi hal penting dalam kaitannya dengan kualitas data.

1. Personil yang kompeten (competent personel) agar sistem sesuai atau cocok digunakan.
2. Kontrol input (input control) merupakan hal bagian dari kontrol yang penting, apalagi dalam kasus transaksi *online*.

3. Jika perlu sediakan manajer kualitas data (DQ manager) yang bertanggung jawab pada kualitas data dalam sistem informasi akuntansi. (Agustinus Mujilan, 2012, 4).

II.2. Anuitas

Anuitas atau yang sering disebut dengan annuity adalah perhitungan bunga dengan mengalikakan persentase bunga dikalikan dengan saldo akhir pinjaman secara tahunan. Secara umum ada dua bentuk pembayaran anuitas yaitu pembayaran anuitas yang dilakukan pada setiap awal periode yang disebut dengan anuitas awal (*anuitas due*) yang digunakan dalam penelitian ini dan pembayaran anuitas yang dilakukan pada setiap akhir periode disebut anuitas akhir (*anuitas immediate*). Tingkat bunga efektif, lamanya pembayaran, dan frekuensi pembayaran adalah faktor yang menentukan suatu transaksi pembayaran yang dilakukan secara berulang-ulang. Pada suatu transaksi jangka panjang, pembayaran biasanya tidak hanya dilakukan sekali dalam setahun tetapi dapat juga dilakukan pembayaran secara berulang kali (n kali) dalam setahun yang disebut banyak pembayaran. Berbagai keputusan finansial yang diambil harus mempertimbangkan suatu ide dasar bahwa akumulasi dari suatu anuitas yang memberikan pembayaran n tahun. Tingkat (Suku) bunga selama n tahun adalah variabel random yang independen dan identik secara distribusi. (Hery:2011)

Besarnya angsuran dengan menggunakan metode anuitas dapat dihitung dengan menggunakan rumus sebagai berikut :

$$A = iM \frac{(1+i)^n}{(1+i)^n - 1}$$

Gambar III.1. Rumusan Metode Anuitas

Bank A memberikan kredit sebesar Rp 12.000.000,- selama 3 tahun (36 bulan) dengan suku bunga 15% pertahun. Berapa jumlah yang diangsur tiap bulan oleh pihak penerima kredit, dan beri detail angsuran pokok dan bunga.

Dari rumus diatas maka didapatkan

$$\text{Angs. Bulanan} = 12.000.000,- \times \frac{15\%}{12} \times \frac{1}{\left(1 - \frac{1}{\left(1 + \frac{15\%}{12}\right)^{36}}\right)}$$

Maka didapatkan besar angsuran yang harus dibayar setiap bulan

Rp. 415.983,94

Analisa tabelnya sebagai berikut :

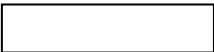
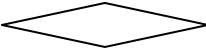
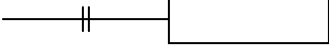
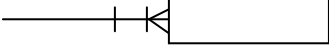
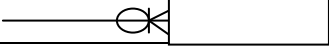
Tabel II.1. Jadwal Angsuran Kredit Dengan Bunga Anuitas

	A	B	C	D	E
1	Plafond	12.000.000,00	Rupiah	Perhitungan Kredit Bunga Anuitas	
2	jml. Angs	36	kali		
3	rate	15%	per tahun		
4					
5	Angsuran	Angsuran	Angsuran	Total	Sisa
6	ke	Pokok	Bunga	Angsuran	Pinjaman pokok
7	0				12.000.000,00
8	1	265.983,94	150.000,00	415.983,94	11.734.016,06
9	2	269.308,74	146.675,20	415.983,94	11.464.707,32
10	3	272.675,10	143.308,84	415.983,94	11.192.032,22
11	4	276.083,54	139.900,40	415.983,94	10.915.948,68
12	5	279.534,58	136.449,36	415.983,94	10.636.414,09
36	29	376.631,21	39.352,73	415.983,94	2.771.586,96
37	30	381.339,11	34.644,84	415.983,94	2.390.247,86
38	31	386.105,84	29.878,10	415.983,94	2.004.142,01
39	32	390.932,17	25.051,78	415.983,94	1.613.209,85
40	33	395.818,82	20.165,12	415.983,94	1.217.391,03
41	34	400.766,55	15.217,39	415.983,94	816.624,47
42	35	405.776,14	10.207,81	415.983,94	410.848,34
43	36	410.848,34	5.135,60	415.983,94	0,00
44					
45		12.000.000,00	2.975.421,91	14.975.421,91	238.033.753,10

II.3. Entity Relationship Diagram

Entity Relationship Diagram (ERD) adalah bagian yang menunjukkan hubungan antara entity yang ada dalam sistem. Simbol-simbol yang digunakan dapat dilihat dari tabel II.2. (Jurnal Momentum Vol 17 No.2 : 2015: Hal : 70).

Tabel II.2. Simbol Yang Digunakan Pada Entity Relationship Diagram (ERD)

SIMBOL	KETERANGAN
	Entity
	Atribut Dan Entity
	Atribut Dan Entity Dengan Key (Kunci)
	Relasi Atau Aktifitas Antar Entity
	Hubungan Satu Dan Pasti
	Hubungan Banyak Dan Pasti
	Hubungan Satu Tapi Tidak Pasti
	Hubungan Banyak Tapi Tidak Pasti

(Sumber : Jurnal Momentum Vol 17 No.2 : 2015: Hal : 70)

II.4. Kamus Data

Kamus Data (KD) adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Kamus data selain digunakan untuk dokumentasi dan mengurangi redundansi, juga dapat digunakan untuk:

1. Memvalidasi diagram aliran data dalam hal kelengkapan dan keakuratan
2. Menyediakan suatu titik awal untuk mengembangkan layar dan laporan-laporan
3. Menentukan muatan data yang disimpan dalam file-file
4. Mengembangkan logika untuk proses-proses diagram aliran data

KD dibuat pada tahap analisis sistem dan digunakan baik pada tahap analisis maupun pada tahap perancangan sistem. Pada tahap analisis sistem, KD dapat digunakan sebagai alat komunikasi antara analisis sistem dengan pemakai sistem tentang data yang mengalir di sistem, yaitu tentang data yang masuk ke sistem dan tentang informasi yang dibutuhkan oleh pemakai sistem. Pada tahap perancangan sistem, KD digunakan untuk merancang input, merancang laporan-laporan dan database. Kamus data dibuat berdasarkan arus data yang ada di DFD. KD mendefinisikan elemen data dengan fungsi sebagai berikut:

1. Menjelaskan arti aliran data dan penyimpanan data dalam DFD
2. Mendeskripsikan komposisi paket data yang bergerak melalui aliran (misalnya alamat diuraikan menjadi kota, negara dan kode pos)
3. Mendeskripsikan komposisi penyimpanan data
4. Menspesifikasikan nilai dan satuan yang relevan bagi penyimpanan dan aliran.

5. Mendeskripsikan hubungan detail antar penyimpanan (yang akan menjadi titik perhatian dalam *entity-relationship diagram*). (Jurnal : Jurusan teknik elektro politeknik negeri lhokseumawe : 2010)

II.5. Normalisasi

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 relasional.

1. Bentuk Nornal Pertama (1 NF)

Contoh yang kita gunakan di sini adalah sebuah perancangan yang mendapatkan barang dari sejumlah pemasok. Masing-masing pemasok berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu pemasok dan masing-masing kota mempunyai kode status tersendiri. Masing-masing pemasok bisa menyediakan banyak barang. Tabel relasionalnya dapat dituliskan sebagai berikut :

PEMASOK (P#, Status, Kota, b#, qty) di mana

p# : kode pemasok (kunci utama)

status : kode status kota

Kota : nama kota

b# : barang yang dipasok

qty : jumlah barang yang dipasok.

Sebuah tabel relasional secara defenisi selalu berada dalam bentuk normal pertama. Semua nilai pada kolom-kolomnya adalah atomi. Ini berarti kolom-kolom tidak mempunyai nilai berulang. Tabel II.3. menunjukkan tabel pemasok dalam 1 NF.

Tabel II.3. Normalisasi Pertama Pemasok

P#	Status	Kota	B#	Qty
P1	20	Yogyakarta	B1	300
P1	20	Yogyakarta	B2	200
P1	20	Yogyakarta	B3	400
P1	20	Yogyakarta	B4	200
P1	20	Yogyakarta	B5	100
P1	20	Yogyakarta	B6	100
P2	10	Medan	B1	300
P2	10	Medan	B2	400
P3	10	Medan	B2	200
P4	20	Yogyakarta	B2	200
P4	20	Yogyakarta	B4	300
P4	20	Yogyakarta	B5	400

Sumber : (Janner Simarmata, dkk, 2010)

2. Bentuk Normal Kedua (2 NF).

Defenisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1 NF, tetapi tidak pada 2 NF, sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada 1 NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada kunci utama. Ini berarti bahwa setiap kolom bukan kunci harus tergantung pada seluruh kolom yang membentuk kunci utama. Tabel pemasok berada pada 1 NF, tetapi tidak pada 2 NF karena status dan kota tergantung secara fungsional hanya pada kolom p# dari kunci gabungan (p#, b#). Ini dapat digambarkan dengan membuat daftar ketergantungan fungsional.

$P\# \longrightarrow \text{Kota, Status}$

$\text{Kota} \longrightarrow \text{Status}$

$(P\#, B\#) \longrightarrow \text{qty}$

Proses mengubah tabel 1 NF ke 2 NF adalah :

- Tentukan sembarang kolom penentu selain kunci gabungan dan kolom-kolom yang ditentukannya.
- Buat dan beri nama tabel baru untuk masing-masing penentu dan kolom-kolom yang ditentukan.
- Pindahkan kolom-kolom yang ditentukan dari tabel asal ke tabel baru penentu akan menjadi kunci utama pada tabel baru.
- Hapus kolom yang baru dipindahkan dari tabel asal, kecuali penentu yang akan berfungsi sebagai kunci tamu.
- Tabel asal bisa diberi nama baru.

Pada contoh, kita memindahkan kolom $p\#$, status, dan kota ke tabel baru yang disebut pemasok2. Kolom $p\#$ menjadi kunci utama tabel ini. Tabel II.4. menunjukkan hasilnya.

Tabel II.4. Tabel Bentuk Normal Kedua (2NF).

Pemasok2

Barang

P#	Status	Kota
P1	20	Yogyakarta
P2	10	Medan
P3	10	Medan
P4	20	Yogyakarta
P5	30	Bandung

P#	B#	Qty
P1	B1	300
P1	B2	200
P1	B3	400
P1	B4	200
P1	B5	100
P1	B6	100
P2	B1	300
P2	B2	400
P3	B2	200
P4	B2	200
P4	B4	300
P4	B5	400

Sumber : (Janner Simarmata, dkk, 2010)

3. Bentuk Normal Ketiga (3 NF).

Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional hanya pada kunci utama. Secara definisi, sebuah tabel berada pada bentuk normal ketiga (3 NF) jika tabel sudah berada pada 2 NF dan setiap kolom yang bukan kunci tidak tergantung secara transitif pada kunci utamanya. Dengan kata lain, semua atribut bukan kunci tergantung secara fungsional hanya pada kunci utama. Tabel barang sudah dalam bentuk normal ketiga. Kolom bukan kunci, qty, tergantung sepenuhnya pada kunci utama (p#, b#). Pemasok masih berada pada 2 NF, tetapi belum berada pada 3 NF karena dia mengandung ketergantungan transitif. Ketergantungan transitif terjadi ketika sebuah kolom bukan kunci, yang ditentukan oleh kunci utama, menentukan kolom lainnya. Konsep ketergantungan transitif dapat digambarkan dengan menunjukkan ketergantungan fungsional pada pemasok2, yaitu :

Pemasok2. p# $\longrightarrow$ Pemasok2, status

Pemasok2. p# $\longrightarrow$ Pemasok2, kota

Pemasok2. kota $\longrightarrow$ Pemasok2, status

Perlu dicatat bahwa pemasok2, status ditentukan, baik oleh kunci utama p#, maupun kolom bukan kunci, kota

Proses mengubah tabel menjadi 3 NF adalah :

- a. Tentukan semua penentu selain kunci utama dan kolom yang ditentukannya.
- b. Buat dan beri nama tabel baru untuk masing-masing penentu dan kolom yang ditentukannya.
- c. Pindahkan kolom yang ditentukan dari tabel asal ke tabel baru. Penentu menjadi kunci utama tabel baru.
- d. Hapus kolom yang baru saja dipindahkan dari tabel asal, kecuali penentu yang akan berfungsi sebagai kunci tamu.
- e. Tabel asal bisa diberi nama baru.

Untuk mengubah PEMASOK2 menjadi 3 NF, kita membuat tabel baru yang disebut KOTA_STATUS dan memindahkan kolom kota dan status ke tabel baru. Status dihapus dari tabel diberi nama baru PEMASOK_KOTA. Tabel II.5 menunjukkan hasilnya.

Tabel II.5. Tabel Bentuk Normal Ketiga (3 NF)

PEMASOK_KOTA		KOTA_STATUS	
P#	Kota	Kota	Status
P1	Yogyakarta	Yogyakarta	20
P2	Medan	Medan	10
P3	Medan	Bandung	30
P4	Yogyakarta	Semarang	40
P5	Bandung		

Sumber : (Janner Simarmata, dkk, 2010)

4. Bentuk Normal Boyce Code (BCNF)

Setelah 3 NF, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Banyak praktisi berpendapat bahwa menempatkan entitas pada 3 NF sudah cukup karena sangat jarang entitas yang berada pada 3 NF bukan merupakan 4 NF dan 5 NF. Lebih lanjut, mereka berpendapat bahwa keuntungan yang didapat mengubah entitas ke 4 NF dan 5 NF sangat kecil sehingga tidak perlu dikerjakan. Bentuk Normal Boyce- Code (BCNF) adalah versi 3 NF lebih teliti dan berhubungan dengan tabel relasional yang mempunyai (a) banyak kunci kandidat (b) kunci kandidat gabungan, dan (c) kunci kandidat yang saling tumpang tindih.

BCNF didasarkan pada konsep penentu. Sebuah kolom penentu adalah kolom di mana kolom-kolom lain sepenuhnya tergantung secara fungsional. Sebuah tabel relasional berada pada BCNF jika dan hanya setiap penentu adalah kunci kandidat.

5. Bentuk Normal Keempat (4 NF)

Sebuah tabel relasional berada pada bentuk normal keempat (4 NF) jika dia dalam BCNF dan semua ketergantungan multivalued merupakan ketergantungan fungsional.

Bentuk normal keempat (4 NF) didasarkan pada konsep ketergantungan multivalued (MVD). Sebuah ketergantungan multivalued terjadi ketika dalam sebuah tabel relasional yang mengandung setidaknya tiga kolom, satu kolom mempunyai banyak baris bernilai sama, tetapi kolom lain bernilai berbeda.

Defenisi secara formal diberikan oleh C.J. Date, yaitu :

Misalnya, ada sebuah tabel relasional R dengan kolom A, B dan C, Maka $R.A \twoheadrightarrow R.B$ (kolom A menentukan kolom B).

Adalah benar jika dan hanya jika himpunan nilai B yang cocok dengan pasangan nilai A dan nilai C pada R hanya tergantung pada nilai A dan tidak tergantung pada nilai C.

MVD selalu terjadi dalam pasangan, yaitu $R.A \twoheadrightarrow R.B$ dipenuhi jika dan hanya jika $R.A \twoheadrightarrow R.C$ dipenuhi pula.

6. Bentuk Normal Kelima (5 NF).

Sebuah tabel berada pada bentuk normal kelima jika dia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil.

Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*). Ketergantungan gabungan berarti sebuah tabel, setelah dekomposisi menjadi tiga atau lebih tabel yang lebih kecil,

harus dapat digabungkan kembali untuk membentuk tabel asal. Dengan kata lain 5 NF menunjukkan ketika sebuah tabel tidak dapat dideskomposisi lagi (Janner Simarmata, 2012).

II.6. Basis Data (*Database*)

Secara sederhana database (basis data/ pangkalan data) dapat diungkapkan sebagai suatu pengorganisasian data dengan bantuan komputer yang memungkinkan data dapat diakses dengan mudah dan cepat (Kadir, 2004). Pengertian akses dapat mencakup pemerolehan data maupun manipulasi data seperti menambah serta menghapus data. Dengan memanfaatkan komputer, data dapat disimpan dalam media pengingat yang disebut *hard disk*. Dengan menggunakan media ini, keperluan kertas untuk menyimpan data dapat dikurangi. Selain itu, data menjadi lebih cepat untuk diakses terutama jika dikemas dalam bentuk database.

Pengaplikasian database dapat kita lihat dan rasakan dalam keseharian kita. Database ini menjadi penting untuk mengelola data dari berbagai kegiatan. Misalnya, kita bisa menggunakan mesin ATM (anjungan tunai mandiri/ *automatic teller machine*) bank karena bank telah mempunyai database tentang nasabah dan rekening nasabah. Kemudian data tersebut dapat diakses melalui mesin ATM ketika bertransaksi melalui ATM. Pada saat melakukan transaksi, dalam konteks database sebenarnya kita sudah melakukan perubahan (*update*) data pada database di bank.

Ketika kita menyimpan alamat dan nomor telepon di HP, sebenarnya juga telah menggunakan konsep database. Data yang kita simpan di HP juga

mempunyai struktur yang diisi melalui formulir (*form*) yang disediakan. Pengguna dimungkinkan menambahkan nomor HP, nama pemegang, bahkan kemudian dapat ditambah dengan alamat *email*, alamat *web*, nama kantor, dan sebagainya.

Pemahaman tentang database ini dapat didekatkan pada konsep akuntansi. Kita bisa umpamakan bahwa ketika kita melakukan proses akuntansi secara manual, kita menuliskan suatu catatan ke dalam lajur dan kolom buku. Mulai dari jurnal, buku besar, buku pembantu kita memasukkan catatan satu demi satu. Melihat buku akuntansi tersebut, sebenarnya kita sudah melihat konsep database, yang jika dikelola dengan komputer masih diperlukan penyesuaian dalam membentuk kolom-kolomnya. (Mujilan, 2012)

II.6.1. Model Database

Model database yang saat ini banyak digunakan adalah model database relational. Imam (2008) menyebutkan “Model database ini disusun dalam bentuk tabel dua dimensi yang terdiri dari baris (*record*) dan (*field*), pertemuan antara baris dengan kolom disebut item data (*data value*). Tabel-tabel yang ada dihubungkan (*relationship*) sedemikian rupa menggunakan *field-field* kunci (*key field*) sehingga dapat meminimalkan duplikasi data.”

Model database relational ini dapat kita kenal konsepnya mulai dari yang paling sederhana misalnya dengan penerapan program aplikasi *excel*. Meskipun untuk pengelolaan database secara luas *excel* jarang digunakan dan kurang mencukupi, namun untuk melihat konsep database dan konsep membangunnya program ini dapat dimanfaatkan. *Excel* mempunyai baris yang disebut *raw* dan mempunyai kolom. Kemudian item data merupakan sel atau

pertemuan antara baris dan kolom. Tabel-tabel dapat diumpakan apabila kita menggunakan tabel dalam suatu *sheet* tertentu. Data dari berbagai tabel dapat diambil dari tabel lain menggunakan perintah *look up* yang berdasarkan kode kunci tertentu. Kode kunci tersebut berada pada suatu kolom tertentu, yang dalam konsep database relational disebut sebagai *key field* tadi.

II.6.2. Struktur Database

Untuk memahami konteks database kita perlu memahami istilah dan hal-hal yang terkait dengan database. Dalam berbagai program aplikasi database terdapat kesamaan ataupun sedikit perbedaan di dalamnya. Seseorang yang mempelajari database dengan program aplikasi tertentu harus memperhatikan struktur dan karakteristik sesuai dengan bahasa dalam aplikasi tersebut. Namun demikian, secara umum terdapat karakteristik sebagai berikut:

1. Nama *file*

Nama *file* adalah nama yang digunakan untuk mengidentifikasi adanya data yang disimpan dalam komputer dan digunakan untuk pemanggilan data. *File* yang dikelola akan muncul dalam komputer dengan ekstensi sesuai dengan program aplikasinya. *File* tersebut dapat digunakan untuk menandakan adanya *file* database, ataupun *file table*. Database dan table akan saling terkait, meskipun cara menyimpan dalam komputer akan mengalami sedikit perbedaan pada beberapa aplikasi. Misalnya Ms. Access akan menyimpan dengan *file* yang dapat kita lihat adalah *file* databasenya. Di program *MySQL* nama database ini akan menjadi *folder*. Sementara di *FoxPro* nama database dapat menjadi *file* tersendiri. *Table* cara menyimpannya juga berbeda, dalam Ms. Access mungkin kita tidak melihat

nama *table* secara kasat mata karena akan dikelola di dalam *file* database. Di dalam *MySQL* kita bisa melihat beberapa nama *file* terkait dengan pengelolaan *table*. Dan di dalam *FoxPro table* ini dapat menjadi nama *file* terpisah dan dapat dikenali pula sebagai *free table*.

2. Database

Database sebenarnya merupakan nama untuk menampung berbagai *table* di dalamnya. Konsep ini akan sama dalam berbagai program aplikasi. Misalnya kita membangun database akuntansi dengan nama database “*akun_base*”. Di dalam *akun_base* akan diorganisasi berbagai *table* yang terkait dengan kegiatan akuntansi misalnya tabel: rekening, pelanggan, jurnal, buku induk, dan administrator program, dan sebagainya. Setiap data yang masuk tidaklah dicatat dalam database, namun di dalam masing-masing *table* yang sesuai.

3. Table

Table merupakan tempat untuk menyimpan data sesuai dengan kelompok data. Setiap isi *table* mengandung data yang mempunyai karakteristik dalam penggunaannya. Untuk mempermudah pengolahan biasanya pembangun database mengkategorikan *table* sesuai dengan data isinya sebagai berikut :

a. Master table

Master table berisi data tentang hal-hal utama dalam kegiatan database. *Table* ini berisi record yang relatif permanen atau seringkali menjadi acuan ketika mengoperasikan transaksi. Dalam master tabel identitas record menjadi penting dan diusahakan merupakan data atau kode yang bersifat unik. Unik dapat diartikan bahwa tidak ada dalam satu *table* berisi kode yang sama.

Disain kode menjadi penting di sini. Misalnya dalam mendisain nama akun dalam database akuntansi, maka kode akun menjadi sangat penting artinya. Dalam *table* berisi nama barang, maka kode barang menjadi hal penting. Contoh lain dalam database akademik, tabel *master* dapat berupa : mahasiswa, daftar dosen, daftar kurikulum.

b. *Transaction table*

Tabel transaksi digunakan untuk menyimpan data dalam menjalankan suatu kegiatan atau bisnis. Data ini seringkali akan bertambah dalam kesehariannya ketika terjadi transaksi yang sesuai dengannya. Secara lebih mudah dapat dipahami dalam akuntansi seringkali mencatat transaksi dalam jurnal. Terkait hal tersebut, transaksi ini dicatat dalam tabel jurnal. Dalam mencatat transaksi ini, kita harus menyesuaikan kode data tertentu dengan kode yang terdapat dalam *master table*.

c. *Tabulation table*

Tabulasi data dapat digunakan untuk menyimpan data seperti halnya master data namun bersifat sebagai data pembantu ketika menginput formulir baik untuk data master maupun transaksi. Misalnya untuk memetakan keterangan hobi, jenis kelamin, nama golongan, nama level manajemen, dan sebagainya. Dengan konsep penamaan field yang baik mungkin saja table tabulasi ini dapat digunakan untuk memuat berbagai kelompok data. Misalnya fieldnya berupa kode dan keterangan. Contoh kelompok gender dengan L = laki-laki; P = perempuan. Kelompok level dengan M = Manajer, O = operator, S = seller

d. *Temporary table*

Temporary adalah data sementara yang digunakan untuk membantu ketika terjadi proses transaksi. Data ini dapat saja langsung dihapus ketika transaksi selesai terproses. Misalnya digunakan untuk mempermudah perhitungan, penyimpanan data sementara sebelum diproses setuju ke database. Misalnya: ketika terjadi transaksi di depan kasir, data-data pertama akan ditangkap dan dimasukkan dalam file temporary sebelum akhirnya kasir melakukan perintah “ok” yang menandakan data transaksi siap untuk disimpan atau diproses dalam komputer. Ketika masa tunggu ini, data masih dapat diedit, dibatalkan, ataupun ditambah. Sementara ketika sudah masuk ke sistem, edit atau penambahan akan membutuhkan prosedur tertentu. Seandainya dianalogikan dengan sistem akuntansi maka proses edit data yang telah masuk ke sistem dapat digunakan prosedur seperti halnya melakukan jurnal koreksi.

4. *Field*

Field adalah penanda untuk kolom data. Jika dalam *excel* penanda tersebut adalah kolom A, B, dan seterusnya, sementara dalam konsep *table* dalam database maka nama *field* memegang peranan penting. Dalam konsep table dalam database, ketika memanggil dengan nama *field* tertentu maka data-data di dalamnya akan muncul. Pengolahan dapat dilakukan dengan membuat *filter*, misalnya berdasarkan kode tertentu, berdasarkan *record* tertentu. Misalnya kita ingin memanggil record terkait nama karyawan Andi. Dalam database Andi ini diberi ID: 11001. Sehingga kita bisa menggunakan konsep filtrasi untuk memanggil

personalia dengan kode ID 11001. Apabila data ketemu, maka kita dapat menggunakan data berdasarkan *field-field* yang ada, misalnya nama, tempat lahir, tanggal lahir, alamat, dan sebagainya yang mengacu pada ID 11001.

Dalam mengatur setting *field*, biasanya akan terkait hal-hal sebagai berikut:

- a. *Field type* : tipe *field* ini dapat terkait apakah *field* tersebut akan berisi data berupa *key field* (*primary key*, *secondary key*), atau *descriptor*. *Primary key* akan berisi ID atau kode pokok yang akan digunakan dalam mengidentifikasi *record*, sehingga data di dalam *field* tersebut tidak diijinkan untuk memiliki lebih dari satu data yang sama. *Secondary* adalah subset dari *key* utama. Misalnya saja kode mata kuliah dalam satu semester tidak boleh terdapat lebih dari satu pada ID atau NIM yang sama. *Descriptor* adalah *field* berisi data yang akan merupakan satu kesatuan dengan yang lainnya sebagai penjelasan akan adanya *record* atau ID tertentu.
- b. *Data type*: tipe data merupakan jenis data yang dapat dimasukkan dalam *field*. Hal ini dapat dibagi secara umum sebagai karakter/ *text*, numerik, tanggal dan sebagainya.
- c. *Field Size*. Penting untuk memahami ukuran *field* yang akan digunakan dalam menampung data. Dalam pengembangan sistem harus dapat memperkirakan berapa lebar ukuran *field* yang efektif. Apabila terlalu lebar akan terjadi banyak spasi kosong dan berpengaruh pada ukuran *file* yang disimpan. Sementara apabila terlalu sempit akan terdapat data yang tidak tersimpan. Misalnya ketika kita menghitung bahwa nama

menggunakan ukuran 40 karakter sudah memenuhi untuk *field* data kita. Konsekuensinya, apabila terdapat nama di atas 40 karakter maka akan terpotong menjadi 40 karakter. Konsekuensi lain adalah nama tersebut diinput hingga memuat maksimal 40 karakter yaitu dengan mengadakan singkatan nama.

5. *Records*

Records merupakan baris data. Karena satu baris data biasanya mengindikasikan satu kesatuan data tertentu, maka satu *record* ada yang menyebut satu data. Misalnya keterangan mengenai biodata Andi disimpan dalam satu record beridentitas ID 11001, maka untuk menyebutkan satu kesatuan data seputar Andi dalam baris tertentu ada yang menyebutkan sebagai *record* data Andi. Dalam konsep database masing-masing *record* memiliki nomor identitas tersendiri baik itu identitas yang diberikan komputer ataupun yang diinputkan secara manual. Sehingga dalam konteks tertentu dapat digunakan konsep nomor *record*, ID otomatis, ID *primary key*. (Mujilan, 2012)

II.7. Bahasa Pemograman *Microsoft Visual Studio 2010*

Visual Basic 2010 adalah inkarnasi dari bahasa visual basic yang sangat populer dan telah dilengkapi dengan fitur serta fungsi yang setara dengan bahasa tingkat lainnya seperti C++. Anda dapat menggunakan visual basic 2010 untuk membuat aplikasi windows, mobile, web, dan office atau kode yang telah ditulis oleh orang lain dan kemudian dimasukkan ke program lainnya. Visual basic menyediakan berbagai tools dan fitur canggih yang memungkinkan dapat menulis

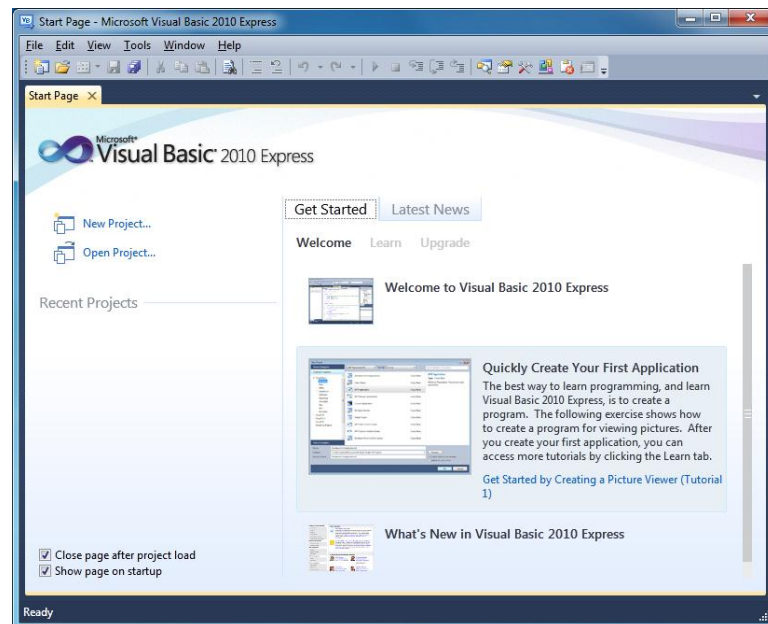
kode, menguji dan menjalankan program tunggal atau terkadang serangkaian program yang terkait dengan satu aplikasi.

Tabel II.6. Jendela Aplikasi Visual Studio 2010

No	Bagian	Keterangan
1	Title Bar	Menampilkan nama aplikasi yang sedang terbuka.
2	Mneu Bar	Menampilkan daftar perintah yang memungkinkan anda dapat menulis, mengedit, menyimpan, mencetak, menguji dan menjalankan program visual basic.
3	Standard Toollbar	Berisi Tombol yang menjalankan perintah yang sering digunakan seperti open project, new project, save, cut, copy, paste, dan undo
4	Toolbox	Berisi komponen NET yang dapat anda gunakan untuk mengembangkan antarmuka pengguna grafis untuk program visual studio 2010.
5	Area Kerja Utama	Menampilkan item yang sedang di kerjakan
6	Solution Explorer	Menampilkan elemen dari visual basic solution, yaitu nama yang diberikan kepada program visual basic dan item lainnya yang dihasilkan oleh visual basic 2010 sehingga program akan mengeksekusi dengan benar.
7	Properties Window	Setiap Objek dalam program visual basic memiliki seperangkat karakteristik yang disebut sifat-sifat objek.

Untuk melihat tampilan visual basic 2010 dapat dilihat pada gambar II.2.

sebagai berikut :



Gambar II.2. Tampilan Utama Visual Basic 2010

Sumber : (Christopher Lee:2014)

II.8. SQL Server 2008

SQL Server 2008 adalah sebuah RDBMS (*Relational Database Management System*) yang di-develop oleh Microsoft, yang digunakan untuk menyimpan dan mengolah data. Pada *SQL Server 2008*, kita bisa melakukan pengambilan dan modifikasi data yang ada dengan cepat dan efisien. Pada *SQL Server 2008*, kita bisa membuat objek-objek yang sering digunakan pada aplikasi bisnis, seperti membuat database, *table*, *function*, *stored database*, *trigger* dan *view*. Selain objek, kita juga menjalankan perintah *Sql (Structured Query Language)* untuk mengambil data. (Elex Media Competindo, 2010,101)

II.9. UML (*Unified Modelling Language*)

Unified Modelling Language (UML) menyediakan beberapa notasi dan artifak standar yang bisa digunakan sebagai alat komunikasi bagi para pelaku dalam proses analisis dan desain. Artifak dalam UML adalah informasi dalam berbagai bentuk yang digunakan atau dihasilkan dalam proses pengembangan perangkat lunak. Contohnya adalah *source code* yang dihasilkan oleh pemrograman. Yang perlu diperhatikan untuk menjaga konsistensi antar artifak selama proses analisis dan desain adalah bahwa setiap perubahan terjadi pada suatu artifak yang harus juga dilakukan pada artifak lainnya. (Sumber: Evi Triandini Dan I gede Suardika, 2012)

II.9.1. Diagram-*Diagram UML*

1. *Diagram Use Case (Use Case Diagram)*

Pengertian *use case* menurut Jhon Satzinger, 2010 dalam buku *system analysis and design in a changing world* menyatakan bahwa “*use case* adalah sebuah kegiatan yang dilakukan oleh system, biasanya dalam menanggapi permintaan dari pengguna system. (Evi Triandini Dan I gede Suardika, 2012)

Notasi Use Case Diagram dapat dilihat pada gambar berikut ini:

Notasi Use Case Diagram



Gambar II.3. Notasi Use Case Diagram
(sumber : Evi Triandini dan I Gede Suardika)

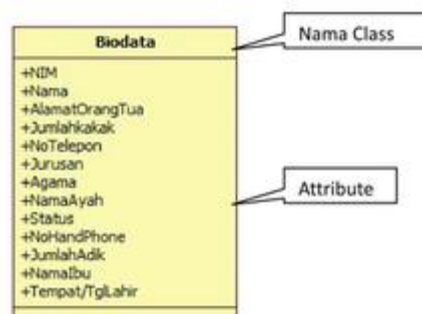
2. Diagram Kelas (*Class Diagram*)

Menurut John Satzinger, dalam buku *system analysis and design in a changing world* menyatakan bahwa “dalam uml ada dua jenis *class diagram* yaitu: domain class diagram dan design class diagram

1. *Domain class diagram*

Focus domain class diagram adalah pada sesuatu dalam lingkungan kerja pengguna, bukan pada class perangkat lunak yang nantinya akan dirancang.

Berikut uni adalah gambar dari domain class diagram:

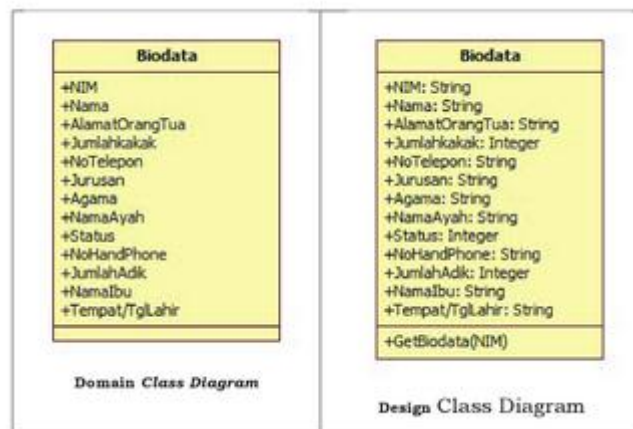


Gambar II.4. domain class diagram
(sumber : Evi Triandini dan I Gede Suardika)

2. Design Class Diagram

Tujuan utamanya adalah untuk mendokumentasikan dan menggambarkan kelas-kelas dalam pemrograman yang nantinya akan dibangun. Design class diagram menggambarkan kelas berorientasi objek yang dibutuhkan dalam pemrograman, navigasi diantara kelas, atribut names dan propertinya serta method names dan propertinya.

Gambar dari class diagram dapat dilihat pada gambar di bawah ini:

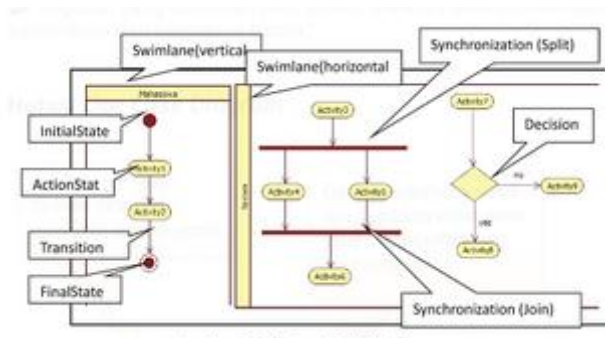


Gambar II.5. Class diagram
(sumber : Evi Triandini dan I Gede Suardika)

3. Diagram Aktivitas (*Activity Diagram*)

Menurut John Satzinger, dalam buku *system analysis and design in a changing world* menyatakan bahwa "activity diagram adalah sebuah diagram alur kerja yang menjelaskan berbagai kegiatan pengguna (atau pun sistem), orang yang melakukan masing-masing aktivitas, dan aliran sekuensial dari aktivitas-aktivitas tersebut.

Notasi activity diagram adalah sebagai berikut:



Gambar II.6. Notasi activity diagram
(sumber : Evi Triandini dan I Gede Suardika)

Penjelasan dari gambar diatas adalah sebagai berikut:

Tabel II.7. Keterangan Notasi activity diagram

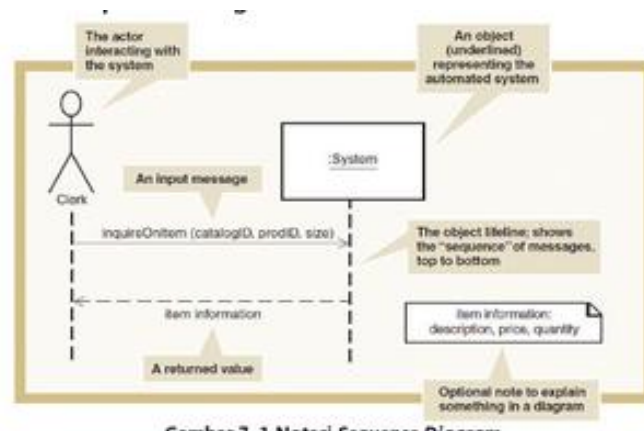
Notasi	Keterangan
Swimlane	Mewakili agen yang melakukan aktivitas karena dalam alur kerja umumnya mempunyai agen yang berbeda dengan yang melakukan langkah yang berbeda dari proses alur kerja.
Initialstate	Awal dari alur kerja
ActionState	Melambangkan aktivitas tersendiri dalam alur kerja
Transition	Melambangkan dari urutan diantara aktivitas
Final State	Akhir dari alur kerja
Sychronization	Membagi alur kerja menjadi beberapa alur yang berbarengan ataupun menggabungkan lagi alur yang berbarengan
Decision	Titik pengambilan keputusan dimana aliran proses tersebut akan mengikuti satu jalur atau jalur lainnya

(Sumber : Evi Triandini dan I Gede Suardika)

4. Sequence Diagram

Menurut John Satzinger, dalam buku system analysis and design in a changing world menyatakan bahwa” System Sequence Digram (SSD) adalah diagram output serta urutan interaksi antara pengguna dan system untuk sebuah use case.

Berikut ini adalah gambar dari sequence diagram :



Gambar II.7. sequence diagram
(sumber : Evi Triandini dan I Gede Suardika)

Penjelasan dari gambar diatas adalah sebagai berikut:

- Actor*, mewakili seprang aktor (orang atau peran yang berinteraksi dengan sistem)
- Kontak berlabel, system adalah objek yang mwakili keseluruhan sistem yang terotomatisasi
- Garis putus-putus vertikal (*lifelines*) adalah perpanjangan objek tersebut, baik aktor maupun objek, sepanjang durasi dari sequence diagram
- Anak panah antara *lifeline* mewakili *message* yang dikirim atau diterima oleh aktor dari sistem

- e. *Message* diberi label untuk menggambarkan maksud message dan input apa pun yang sedang dikirim. *Message* dipertimbangkan sebagai sebuah inti yang diminta pada tujuan objek, kebanyakan seperti perintah.