BAB II

LANDASAN TEORI

II.1. Sistem Pendukung Keputusan

Sistem pendukung keputusan (SPK) atau *Decision Support Systems* (DSS) adalah sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data yang digunakan untuk membantu pengambilan keputusan pada situasi yang semiterstruktur dan situasi yang tidak terstruktur di mana tak seorang pun tahu secara pasti bagaimana seharusnya keputusan dibuat (Alter, 2002). Konsep DSS dikemukakan pertama kali oleh Scott-Morton pada tahun 1971 (Turban, McLean, dan Wetherbe, 1999). Beliau mendefinisikan cikal bakal DSS tersebut sebagai "Sistem berbasis komputer yang interaktif, yang membantu pengambil keputusan menggunakan data dan model untuk memecahkan persoalan-persoalan tidak terstruktur" (Abdul Kadir; 2014; 108).

Sistem Pendukung Keputusan (SPK) biasanya dibangun untuk mendukung solusi atas suatu masalah atau untuk suatu peluang. Aplikasi Sistem Pendukung Keputusan (SPK) digunakan dalam pengambilan keputusan. Aplikasi Sistem Pendukung Keputusan (SPK) menggunakan CBIS (*Computer Based Information Systems*) yang fleksibel, interaktif, dan dapat diadaptasi, yang dikembangkan untuk mendukung solusi atas masalah manajemen spesifik yang tidak terstruktur.

Menurut Bonczek, dkk., dalam buku *Decision Support Systems And Intelligent Systems* (Turban; 2005; 137) mendefenisikan sistem pendukung keputusan sebagai sistem berbasis komputer yang terdiri dari tiga komponen yang

saling berinteraksi, sistem bahasa (mekanisme untuk memberikan komunikasi antara pengguna dan komponen sistem pendukung keputusan lain), sistem pengetahuan (respositori pengetahuan domain masalah yang ada pada sistem pendukung keputusan atau sebagai data atau sebagai prosedur), dan sistem pemrosesan masalah (hubungan antara dua komponen lainnya, terdiri dari satu atau lebih kapabilitas manipulasi masalah umum yang diperlukan untuk pengambilan keputusan) (Dicky Nofriansyah; 2014; 1).

II.1.1. Karakteristik Sistem Pendukung Keputusan

Karakteristik dari sistem pendukung keputusan yaitu (Abdul Kadir; 2014; 108) :

1. Menawarkan keluwesan, kemudahan beradaptasi, dan tanggapan yang cepat.
2. Memungkinkan pemakai memulai dan mengendalikan masukan dan keluaran.
3. Dapat dioperasikan dengan sedikit atau tanpa bantuan pemrogram profesional.
4. Menyediakan dukungan untuk keputusan dan permasalahan yang solusinya tak dapat ditentukan di depan.
5. Menggunakan analisis data dan perangkat pemodelan yang canggih.

II.1.2. Komponen Sistem Pendukung Keputusan

Secara garis besar sistem pendukung keputusan dibangun oleh tiga komponen utama yaitu (Dicky Nofriansyah; 2014; 3-4) :

1. Subsistem Data (*Database*)

   Subsistem data merupakan komponen sistem pendukung keputusan yang berguna sebagai penyedia data bagi sistem. Dta tersebut disimpan untuk diorganisasikan dalam sebuah basis data yang diorganisasikan oleh suatu sistem yang disebut dengan sistem manajemen basis data (*Database Management System*).

2. Subsistem Model (*Model Base)*

   Model adalah suatu tiruan dari alam nyata. Kendala yang sering dihadapi dalam merancang model adalah bahwa model yang dirancang tidak mampu mencerminkan seluruh variabel alam nyata, sehingga keputusan yang diambil tidak sesuai dengan kebutuhan. Oleh karena itu, dalam menyimpan berbagai model harus diperhatikan dan harus dijaga fleksibilitasnya. Hal lain yang harus diperhatikan adalah pada setiap model yang disimpan hendaknya ditambahkan rincian keterangan dan penjelasan yang komprehensif mengenai model yang dibuat.

3. Subsistem Dialog (*User System Interface*)

   Subsistem dialog adalah fasilitas yang mampu mengintegrasikan sistem yang terpasang dengan pengguna secara interaktif, yang dikenal dengan subsistem dialog. Melalui subsistem dialog sistem

diimplementasikan sehingga pengguna dapat berkomunikasi dengan sistem yang dibuat.

II.2. Metode *Profile Matching*

Profile Matching merupakan suatu proses yang sangat penting dalam manajemen SDM di mana terlebih dahulu ditentukan kompetensi (kemampuan) yang diperlukan oleh suatu jabatan. Kompetensi kemampuan tersebut haruslah dapat dipenuhi oleh pemegang atau calon yang akan dinilai kinerjanya. Dalam proses *Profile Matching* secara garis besar merupakan proses membandingkan antara kompetensi individu ke dalam kompetensi jabatan sehingga dapat diketahui perbedaan kompetensinya (disebut juga *gap*), Semakin kecil *gap* yang dihasilkan maka bobot nilainya semakin besar berarti memiliki peluang lebih besar untuk karyawan menempati posisi tersebut (Pelita Informatika Budi Darma; Volume. 5 Nomor. 2; 2013; 13).

II.2.1. Perhitungan *Profile Mathing*

Dalam proses *profile matching* secara garis besar merupakan proses membandingkan antara setiap kriteria setiap penilaian dalam sebuah proposal usulan penelitian yang diajukan sehingga diketahui perbedaan skornya (disebut juga *gap*), semakin kecil *gap* yang dihasilkan maka bobot nilainya semakin besar yang berarti memiliki peluang lebih besar untuk prioritas kelayakan/kelulusan. Nilai gap dapat dihitung menggunakan persamaan (1). Sedangkan pembobotan

nilai gap ditentukan berdasarkan Tabel II.1 (Journal Speed – Sentra Penelitian Engineering dan Edukasi; Volume. 6 No. 1; 2014; 61).

$$GAP = Profile\ proposal - Profile\ ideal \dots\dots\dots\dots\dots\dots\dots\dots (1)$$

Langkah selanjutnya adalah menghitung nilai *core factor* dan *secondary factor*. *Core factor* merupakan kriteria penilaian yang paling utama harus terkandung dalam sebuah proposal penelitian. Perhitungan *core factor* menggunakan persamaan (2).

Tabel II.1. Pembobotan Nilai GAP

No.	Selisih	Bobot	Keterangan
1.	0	5	Tidak ada selisih skor kriteria
2.	1	4,5	Kriteria kelebihan 1 level
3.	-1	4	Kriteria kekurangan 1 level
4.	2	3,5	Kriteria kelebihan 2 level
5.	-2	3	Kriteria kekurangan 2 level
6.	3	2,5	Kriteria kelebihan 3 level
7.	-3	2	Kriteria kekurangan 3 level

Sumber : Edi Faizal(2014;61)

$$NCF = \frac{\sum NC(kriteria)}{\sum IC} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (2)$$

Keterangan :

NCT : Nilai rata-rata *core factor*

NC : Jumlah total nilai *core factor*

IC : Jumlah item *core factor*

Sedangkan *secondary factor* merupakan item-item selain yang ada pada faktor utama *(core factor). Secondary factor* dihitung menggunakan persamaan (3).

$$NSF = \frac{\sum NS(kriteria)}{\sum IS} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Keterangan :

NST : Nilai rata-rata *secondary factor*

NS : Jumlah total nilai *secondary factor*

IS : Jumlah item *secondary factor*

Selanjutnya perhitungan nilai total berdasar nilai dari *core* dan *secondary factor* yang digunakan sebagai kriteria penilaian yang berpengaruh terhadap kelulusan proposal penelitian. Perhitungan dapat dilakukan menggunakan persamaan (4).

$$N\ (Tot\_kriteria) = (x)\%NCF + (x)\%NSF \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Keterangan :

NCT : Nilai rata-rata *core factor*

NST : Nilai rata-rata *secondary factor*

NT : Nilai total kriteria penilaian

Langkah terakhir adalah perhitungan ranking, yang dilakukan dengan menggunakan persamaan

$$Ranking\ = (x)\%N1 + (x)\%N2 + (x)\%Nn \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

Keterangan :

N1, N2, Nn : Nilai total per kriteria

(x)% : Persentase nilai kriteria

II.3. Pengertian Basis Data

Basis data adalah suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktivitas untuk memperoleh informasi. Basis data dimaksudkan untuk mengatasi problem pada sistem yang memakai pendekatan berbasis berkas. Untuk mengelola basis data diperlukan perangkat lunak yang disebut *Database Management System* (DBMS). DBMS adalah perangkat lunak sistem yang memungkinkan para pemakai membuat, memelihara, mengontrol, dan mengakses basis data dengan cara yang praktis dan efisien. DBMS dapat digunakan untuk mengakomodasikan berbagai macam pemakai yang memiliki kebutuhan akses yang berbeda-beda (Abdul kadir; 2014; 218).

Pendefinisian basis data meliputi spesifikasi dari tipe data, struktur dan batasan dari data atau informasi yang akan disimpan. Istilah-istilah dalam basis data yaitu (Jurnal ECOTIPE; Volume 1; No 1; 2014; 38):

1. Enterprise: suatu bentuk organisasi seperti Bank, Sekolah, Rumah Sakit, Pabrik, Kantor dan sebagainya.
2. Entitas: suatu objek yang dapat di bedakan dari lainnya yang dapat di wujudkan dalam basis data. Kumpulan dari entitas disebut himpunan entitas.
3. Atribut dan elemen data: karakteristik dari suatu entitas.
4. Record data: kumpulan suatu elemen data yang saling berhubungan.
5. Tabel: kumpulan data atau informasi.

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

Menurut pendapat Kronke (2006:37-40) *Entity-Relationship Diagram* (ERD) adalah suatu pemodelan konseptual yang didesain secara khusus untuk mengidentifikasikan entitas yang menjelaskan data dan hubungan antardata, yaitu dengan menuliskan dalam *cardinality*. Sementara seolah-olah teknik diagram atau alat peraga memberikan dasar untuk desain *database* relasional yang mendasari sistem informasi yang dikembangkan. ERD bersama-sama dengan detail pendukung merupakan model data yang pada gilirannya digunakan sebagai spesifikasi untuk *database*. Dalam sistem *Entity Relationship Diagram* (ERD), terdapat beberapa istilah penting diantaranya (Jurnal Informatika; Vol. II No. 1; 2015; 215) :

1. Entitas (*Entity*)

   Suatu entitas yang dapat berupa orang, tempat, obyek, atau kejadian yang dianggap penting bagi perusahaan, sehingga segala atributnya harus dicatat dan disimpan dalam basis data.

2. Atribut (*Atribute*)

   Setiap entitas mempunyai karakterisik tertentu yang dinamakan dengan atribut.

3. Relasi (*relationship*)

   Hubungan antara dua atau lebih entitas yang saling berkaitan. Menurut Romney (2009:596) ada tiga tipe relasi (*relationship*), yaitu:

   a. *One-to-one relationship* (1:1)

      Dimana maximum *cardinality* setiap *entity* adalah 1.

Contoh : Satu nasabah bank hanya memiliki satu *account*.

b. *One-to-many relationship* (1:N).

Dimana maximum *cardinality* dari suatu *entity* adalah 1 dan maximum *cardinality* dari *entity* lain adalah N.

Contoh : Satu nasabah bank dapat memiliki lebih dari satu *account*.

c. *Many-to-many relationship* (M:N).

Dimana maximum *cardinality* kedua *entity* yang berhubungan adalah N.

Contoh : Satu nasabah dapat memilliki beberapa *account* dan satu *account* dapat dimiliki oleh beberapa nasabah (rekening bersama).

4. *Identifier*

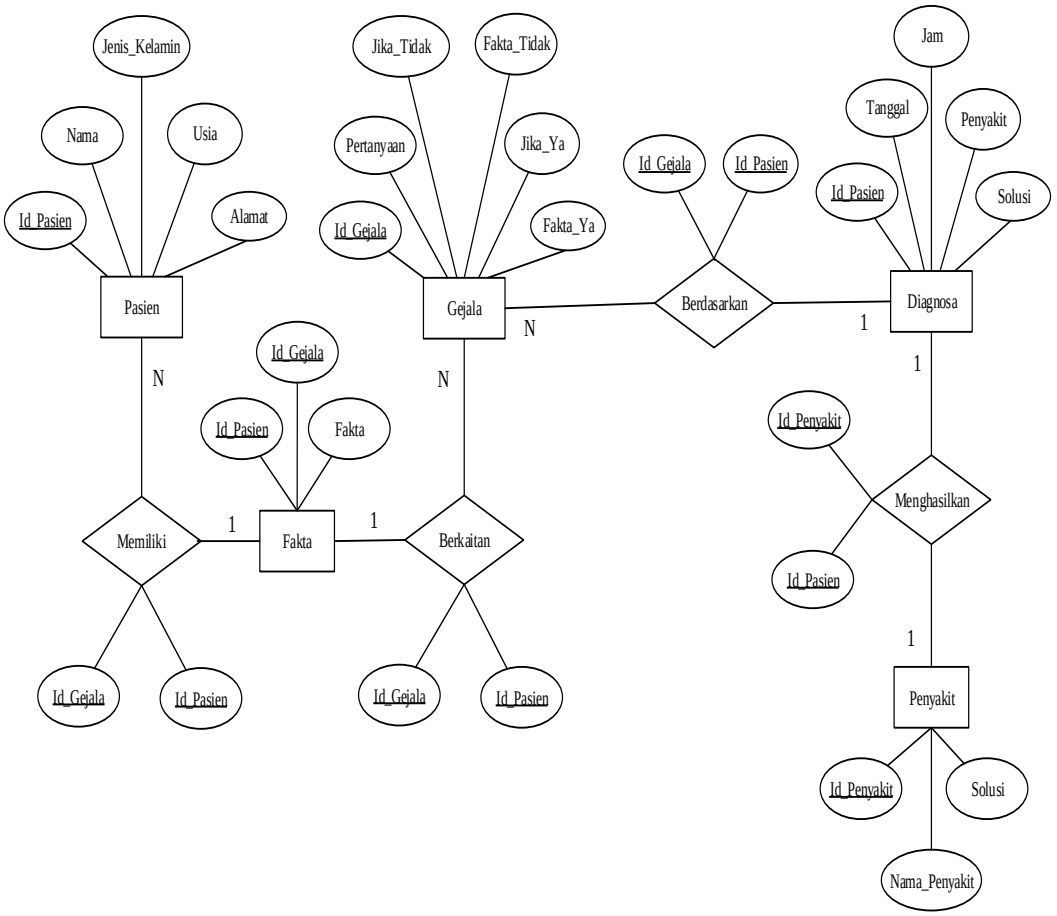
Merupakan nama *atribute* yang digunakan untuk mengidentifikasi suatu entitas, Ada tiga jenis *identifier* diantaranya *Primary Key*, *Secondary Key*, dan *Foreign Key*. Berikut ini penjelasan dari *Primary Key* dan *Foreign Key*:

a. *Primary Key* merupakan suatu kode identifikasi yang bersifat unik yang ditunjukkan oleh masing-masing *record* dalam sistem. Tujuan dari *Primary key* adalah untuk menunjukkan lokasi tiap catatan di dalam suatu *file* mengenai catatan-catatan serupa.

b. *Foreign Key* merupakan *attribute* yang merupakan *Primary key* dari relasi lain yang ditarik/dihubungkan ke suatu relasi.

5. Kardinalitas (*Cardinality*)

   Merupakan kendala-kendala yang timbul dalam hubungan antar entitas.



Gambar II.1. *Entity Relationship Diagram* (ERD)

(Sumber : Jurnal Informatika; Vol. II No. 1; 2015; 216)

II.5. Kamus Data

Kamus data adalah suatu daftar data elemen yang terorganisir dengan definisi yang tetap dan sesuai dengan sistem, sehingga *user* dan analis sistem mempunyai pengertian yang sama tentang *input*, *output*, dan komponen *data*

store. Kamus data ini sangat membantu analis sistem dalam mendefinisikan data yang mengalir di dalam sistem, sehingga pendefinisian data itu dapat dilakukan dengan lengkap dan terstruktur. Pembentukan kamus data dilaksanakan dalam tahap analisis dan perancangan suatu sistem. Pada tahap analisis, kamus data merupakan alat komunikasi antara *user* dan analis sistem tentang data yang mengalir di dalam sistem, yaitu tentang data yang masuk ke sistem dan tentang informasi yang dibutuhkan oleh *user*. Sementara itu, pada tahap perancangan sistem kamus data digunakan untuk merancang *input*, laporan dan database (E-Journal Teknik Elektro dan Komputer; 2014).

Kamus data digunakan untuk membantu para pemakai mengerti mengenai aplikasi yang akan dikembangkan secara terinci dan mengorganisasikan semua elemen data yang terkait serta tidak mengalami kesulitan dalam memahami pemodelan sistem yang dikembangkan secara logika. Contoh kamus data, antara lain (Seminar Nasional Informatika; 2015) :

tb_kategori = @kodekategori + namakategori
tb_menu = @kodemenu + @@kodekategori + namamenu + harga
tb_users = @username + password + nama
tb_mpesan = @nomorpesan + tglpesan + username + nomormeja
tb_pesan = @@nomorpesan + @@kodemenu + + banyak + harga + status

II.6. Normalisasi

Menurut Martin (1975), Normalisasi diartikan sebagai suatu teknik yang menstrukurkan/ mendekomposisi data dalam cara-cara tertentu untuk mencegah

timbulnya permasalahan pengolahan data dalam basis data. Permasalahan yang dimaksud adalah berkaitan dengan penyimpangan-penyimpangan *(anomalies)* yang terjadi akibat adanya kerangkapan data dalam relasi dan in-efisiensi pengolahan (Edy Sutanta ; 2011 : 174).

Proses normalisasi menghasilkan relasi yang optimal, yaitu (Martin, 1975) : (Edy Sutanta ; 2011 : 175)

1. Memiliki struktur *record* yang konsisten secara logik;
2. Memiliki struktur *record* yang mudah untuk dimengerti;
3. Memiliki struktur *record* yang sederhana dalam pemeliharaan;
4. Memiliki struktur *record* yang mudah ditampilkan kembali untuk memenuhi kebutuhan pengguna;
5. Minimalisasi kerangkapan data guna meningkatkan kinerja sistem.

Secara berturut-turut masing-masing level normal tersebut dibahas berikut ini, dimulai dari bentuk tidak normal. (Edy Sutanta ; 2011 : 176-179)

1. Relasi bentuk tidak normal (*Un Normalized Form* / UNF)

   Relasi-relasi yang dirancang tanpa mengindahkan batasan dalam defisi basis data dan karakteristik *Relational Database Management System* (RDBM) menghasilkan relasi *Un Normalized Form* (UNF). Bentuk ini harus di hindari dalam perancangan relasi dalam basis data. Relasi *Un Normalized Form* (UNF) mempunyai kriteria sebagai berikut.

a. Jika relasi mempunyai bentuk *non flat file* (dapat terjadi akibat data disimpan sesuai dengan kedatangannya, tidak memiliki struktur tertentu, terjadi duplikasi atau tidak lengkap)
b. Jika relasi membuat *set atribut* berulang *(non single values)*
c. Jika relasi membuat *atribut non atomic value*

2. Relasi bentuk normal pertama (*First Norm Form* / 1NF)

   Relasi disebut juga *First Norm Form* (1NF) jika memenuhi kriteria sebagai berikut.

   a. Jika seluruh atribut dalam relasi bernilai *atomic (atomic value)*
   b. Jika seluruh atribut dalam relasi bernilai tunggal *(single value)*
   c. Jika relasi tidak memuat set atribut berulang
   d. Jika semua record mempunyai sejumlah atribut yang sama.

   Permasalahan dalam *First Norm Form* (1NF) adalah sebagai berikut.

   a. Tidak dapat menyisipkan informasi parsial
   b. Terhapusnya informasi ketika menghapus sebuah *record*

3. Bentuk normal kedua *(Second Normal Form /* 2NF)

   Relasi disebut sebagai *Second Normal Form (*2NF) jika memenuhi kriteria sebagai berikut

   a. Jika memenuhi kriteria *First Norm Form (*1NF)
   b. Jika semua atribut nonkunci *Functional Dependence* (FD) pada *Primary Key* (PK)

   Permasalahan dalam *Second Normal Form* / 2NF adalah sebagai berikut:

a. Kerangkapan data *(data redundancy)*
b. Pembaharuan yang tidak benar dapat menimbulkan inkonsistensi data *(data inconsistency)*
c. Proses pembaharuan data tidak efisien

Kriteria tersebut mengidentifikasikan bahwa antara atribut dalam *Second Normal Form* masih mungkin mengalami *Third Norm Form*. Selain itu, relasi *Second Normal Form* (2NF) menuntut telah didefinisikan atribut *Primary Key* (PK) dalam relasi. Mengubah relasi *First Norm Form* (1NF) menjadi bentuk *Second Normal Form* (2NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

a. Identifikasikan *Functional Dependence* (FD) relasi *First Norm Form* (1NF)
b. Berdasarkan informasi tersebut, dekomposisi relasi *First Norm Form* (1NF) menjadi relasi-relasi baru sesuai *Functional Dependence* nya. Jika menggunakan diagram maka simpul-simpul yang berada pada puncak diagram ketergantungan data bertindak *Primary Key* (PK) pada relasi baru

4. Bentuk normal ketiga (*Third Norm Form* / 3NF)

Suatu relasi disebut sebagai *Third Norm Form* jika memenuhi kriteria sebagai berikut.

a. Jika memenuhi kriteria *Second Normal Form* (2NF)
b. Jika setiap atribut nonkunci tidak (TDF) *(Non Transitive Dependeny)* terhadap *Primary Key* (PK)

Permasalahan dalam *Third Norm Form* (3NF) adalah keberadaan penentu yang tidak merupakan bagian dari *Primary Key* (PK) menghasilkan duplikasi rinci data pada atribut yang berfungi sebagai *Foreign Key* (FK) (duplikasi berbeda dengan keterangan data).

Mengubah relasi *Second Normal Form* (2NF) menjadi bentuk *Third Norm Form* (3NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

a. Identifikasi TDF relasi *Second Normal Form* (2NF)
b. Berdasarkan informasi tersebut, dekomposisi relasi *Second Normal Form* (2NF) menjadi relasi-relasi baru sesuai TDF-nya.

5. Bentuk normal *Boyce-Cood (Boyce-Codd Norm Form* / BCNF)

   Bentuk normal *Boyce-Codd Norm Form* (BCNF) dikemukakan oleh R.F. Boyce dan E.F. Codd. Suatu relasi disebut sebagai *Boyce-Codd Norm Form* (BCNF) jika memenuhi kriteria sebagai berikut.

   a. Jika memenuhi kriteria *Third Norm Form* (3NF)
   b. Jika semua atribut penentu (determinan) merupakan CK

6. Bentuk normal keempat (*Forth Norm Form* / 4NF)

   Relasi disebut sebagi *Forth Norm Form* (4NF) jika memenuhi kriteria sebagai berikut.

   a. Jika memenuhi kriteria *Boyce-Codd Norm Form*.
   b. Jika setiap atribut didalamnya tidak mengalami ketergantungan pada banyak nilai.

7. Bentuk normal kelima (*Fifth Norm Form* / 5NF)

Suatu relasi memenuhi kriteria *Fifth Norm Form* (5NF) jika kerelasian antar data dalam relasi tersebut tidak dapat direkonstruksi dari struktur relasi yang sederhana.

8. Bentuk normal kunci domain (*Domain Key Norm Form* / DKNF)

   Relasi disebut sebagai *Domain Key Norm Form* (DKNF) jika setiap batasan dapat disimpulkan secara sederhana dengan mengetahui sekumpulan nama atribut dan domainnya selama menggunkan sekumpulan atribut pada kuncinya.

Tabel II.2. Contoh Proses Normalisasi

Kode_Supplier	Status	Kota	Kode_Barang	Jumlah_Barang
S01	10	Jakarta	B01	100
			B02	150
			B03	200
S02	20	Surabaya	B04	250
			B05	200
S03	30	Yogyakarta	B06	150
			B07	100

Sumber : Edhy Sutanta (2011 : 179)

II.7. *SQL Server*

SQL *(Structured Query Language)* adalah bahasa yang digunakan untuk mengakses basis data yang tergolong relasional. Standar SQL mula-mula didefinisikan oleh ISO (*International Standards Organization*) dan ANSI (*the American National Standards Institute*), yang dikenal dengan sebutan SQL86. Sesungguhnya SQL tidak terbatas hanya untuk mengambil data (*query*), tetapi

juga dapat dipakai untuk menciptakan tabel, menghapus tabel, menambahkan data ke tabel, menghapus data di tabel, mengganti data di tabel, dan berbagai operasi yang lain (Abdul Kadir, 2014 : 242).

Microsoft SQL Server adalah salah satu aplikasi DBMS yang sudah sangat banyak digunakan oleh para pemrogram aplikasi basis data. Contoh DBMS lainnya adalah: mYsql (*freeware*), PostgreSQL (*freeware*), MS Access dari Microsoft, DB2 dari IBM, Oracle dan Oracle Corp, Dbase, FoxPro (Priyanto Hidayatullah, 2012 : 176).

II.8. *Microsoft Visual Basic .NET*

Pada tahun 1991 Microsoft mengeluarkan Visual Basic, pengembangan dari Basic yang berubah dari sisi pembuatan antarmukanya. Visual Basic sampai sekarang masih menjadi salah satu bahasa pemrograman terpopuler di dunia. Pada akhir tahun 1999, Teknologi .NET diumumkan. Microsoft memosisikan teknologi tersebut sebagai *platform* untuk membangun XML *Web services* memungkinkan aplikasi tipe apa pun dapat berjalan pada siustem komputer dengan tipe manapun dan dapat mengambil data yang tersimpan pada server dengan tipe apa pun melalui Internet.

Visual Basic .NET adalah Visual Basic yang direkayasa kembali untuk digunakan pada *platform* .NET sehingga aplikasi yang dibuat menggunakan Visual Basic .NET dapat berjalan pada sistem komputer apa pun, dan dapat

mengambil data dari server dengan tipe apa pun asalkan terinstal .NET Framework (Priyanto Hidayatullah, 2012 : 4-5).

Visual Basic .NET layak untuk dijadikan pilihan karena mempunyai cukup banyak kelebihan. Beberapa kelebihan Visual Basic .NET antara lain (Priyanto Hidayatullah, 2012 : 7-8) :

1. Sederhana dan mudah dipahami

   Seperti pada VB, bahasa yang digunakan pada VB .NET sangat sederhana sehingga lebih mudah dipahami bagi mereka yang masih awam terhadap dunia pemrograman.

2. Mendukung GUI

   VB .NET bisa membuat *software* dengan antarmuka grafis yang lebih *user friendly*.

3. Menyederhanakan *deployment*

   VB .NET mengatasi masalah *deployment* dari aplikasi berbasis Windows yaitu DLL Hell dan registrasi COM (*Component Object Model*). Selain itu tersedia wizard yang memudahkan dalam pembuatan *file setup*.

4. Menyedarhanakan pengembangan perangkat lunak

   Ketika terjadi kesalahan penulisan kode dari sisi sintaks (bahasa), maka VB .NET langsung menuliskan kesalahannya pada bagian *Message* Windows sehingga *Programmer* dapat memperbaiki kode dengan lebih cepat.

5. Mendukung penuh OOP

Memiliki fitur bahasa pemrograman berorientasi objek seperti *inheritence* (pewarisan), *encapsulation* (pembungkusan), dan *polymorphism* (banyak bentuk).

6. Mempermudah pengembangan aplikasi berbasis Web

   Disediakan desainer form web. Selain itu disediakan layanan web XML sehingga memungkinkan suatu aplikasi "berkomunikasi" dengan aplikasi lainnya dari berbagai *platform* menggunakan protokol Internet terbuka.

7. Migrasi ke VB .NET dapat dilakukan dengan mudah

   Jika anda sudah mengembangkan aplikasi di VB, maka konversi ke VB .NET dapat anda jalankan dengan mudah.

8. Banyak digunakan oleh *programmer- programmer* di seluruh dunia.

   Salah satu keuntungannya adalah jika kita memiliki masalah/ pertanyaan, maka kita bisa tanyakan kepada *programmer-programmer* lain di seluruh dunia melalui forum-forum di Internet.

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

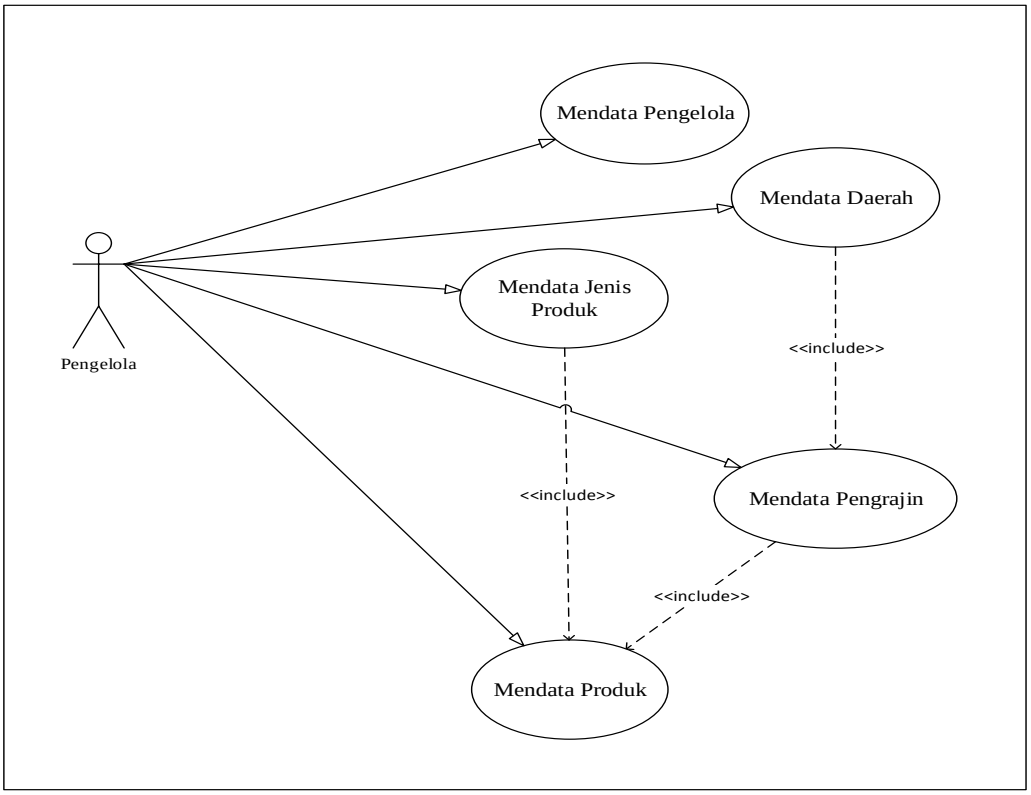
Unified Modeling Language (UML) adalah Bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek. UML ini berfungsi untuk membantu para developer untuk menggambarkan alur dari sebuah sistem yang akan dibangun, gambaran mengenai alur sistem tersebut akan terwakili oleh simbol-simbol yang ada dalam digram–diagram (JTI; Vol 7 No.1; 2015).

Unified Modeling Language (UML) disebut sebagai bahasa pemodelan bukan metode. Kebanyakan metode terdiri paling sedikit prinsip, bahasa pemodelan dan proses. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat. Menurut Rosa dan Shalahuddin (2013:133) UML (*Unified Modeling Language*) adalah salah satu standar bahasa visual yang banyak digunakan di dunia industri untuk mengidentifikasi *requirement,* membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML hanya berfungsi untuk melakukan pemodelan, jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak diguakan pada metodologi berorientasi objek. (Jurnal Informatika; Vol. II No. 1; 2015; 216).

II.9.1. *Use Case* Diagram

Use Case Diagram menurut Widodo (2011:10) Diagram *use case* bersifat statis, yang memperlihatkan himpunan *Use Case* dan aktor-aktor (suatu jenis khusus dari kelas) dan menggambarkan apa saja aktifitas yang dilakukan oleh suatu sistem dari sudut pandang pengamatan luar. Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna, yang menjadi persoalan itu apa yang

dilakukan bukan bagaimana melakukannya. *Use Case Diagram* menggambarkan fungsionalitas yang diharapkan dari sistem (Jurnal Informatika; Vol. II No. 1; 2015; 217).



Gambar II.2. *Use Case* Diagram
(Sumber : Jurnal Informatika dan Komputer (JIKO); Vol. 1 No. 1; 2016)

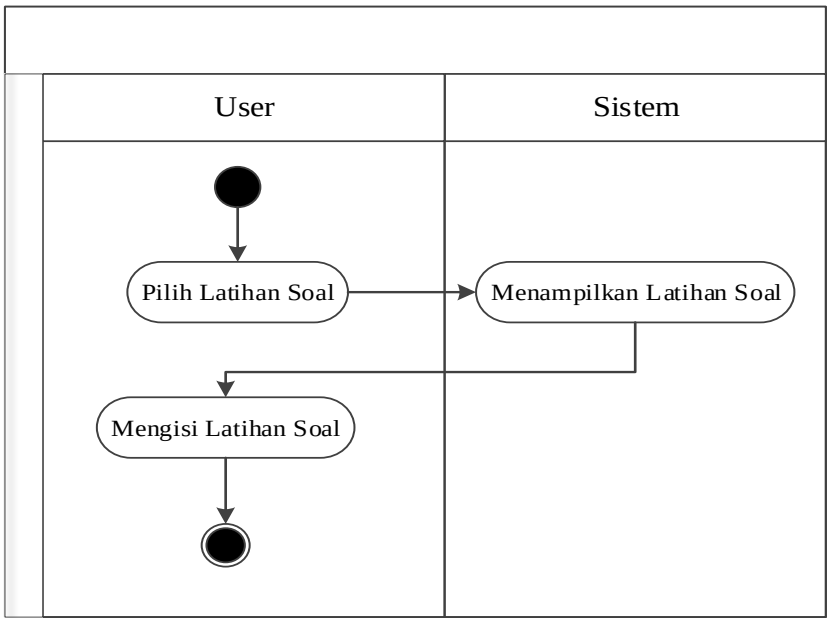
II.9.2. *Activity* Diagram

Activity diagram sesungguhnya merupakan bentuk khusus dari *state machine* yang bertujuan memodelkan komputasi-komputasi dan aliran-aliran kerja yang terjadi dalam sistem / perangkat lunak yang sedang dikembangkan. *Activity* diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang

dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi (JTI; Vol 7 No.1; 2015).

Diagram aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut (Jurnal Ilmiah Komputer dan Informatika (KOMPUTA); Edisi. 01 Volume. 01; 2014) :

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.



Gambar II.3. *Activity* Diagram
(Sumber : JTI; Vol 7 No.1; Juni 2015)

II.9.3. *Class* Diagram

Class Diagram menurut Munawar (2005:28) merupakan himpunan dari objek-objek yang sejenis. Sebuah objek memiliki keadaan sesaat (*state*) dan perilaku (*behavior*). *State* sebuah objek adalah kondisi objek tersebut yang dinyatakan dalam *attribute*. Sedangkan perilaku suatu objek mendefinisikan bagaimana sebuah objek bertindak dan memberikan (Jurnal Informatika; Vol. II No. 1; 2015).

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah

fungsi-fungsi yang dimiliki oleh suatu kelas. Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut (Jurnal Ilmiah Komputer dan Informatika (KOMPUTA); Edisi. 01 Volume. 01; 2014) :

1. Kelas Main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

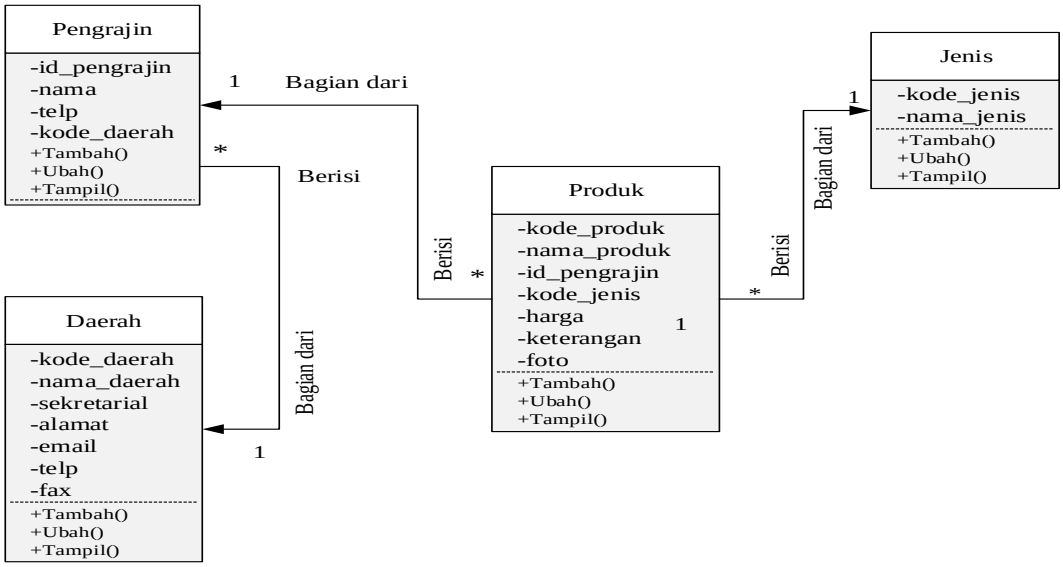
2. Kelas yang menangani tampilan sistem

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

3. Kelas yang diambil dari pendefinisian *usecase*

   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*.

4. Kelas yang diambil dari pendefinisian data



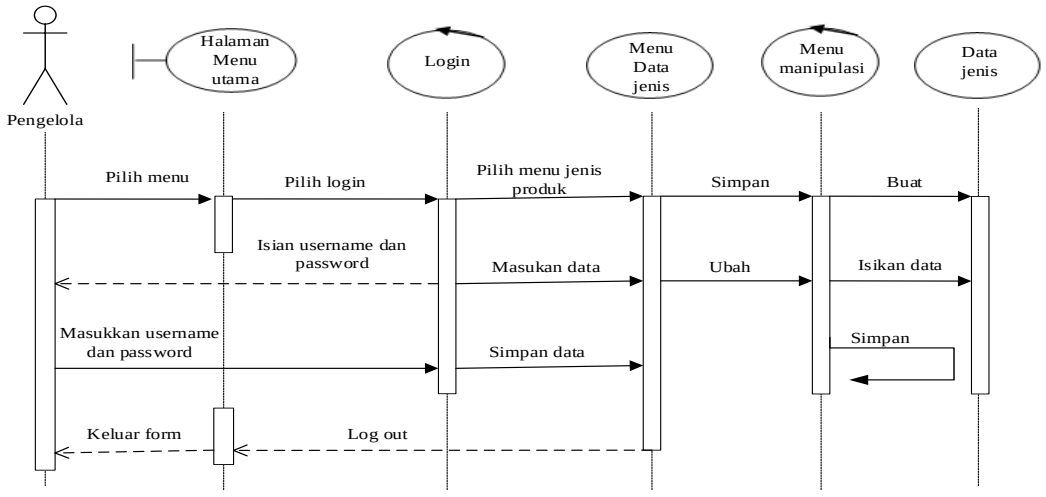
Gambar II.4. *Class* Diagram

(Sumber : Jurnal Informatika dan Komputer (JIKO); Vol. 1, No. 1; 2016)

II.9.4. *Sequence* Diagram

Squence diagram menurut Munawar (2005:187) adalah grafik dua dimensi dimana obyek ditujukan dalam dimensi horizontal, sedangkan *lifeline* ditunjukan dalam dimensi vertikal (Jurnal Informatika; Vol. II No. 1; 2015).

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri antar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence* diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu (JTI; Vol 7 No.1; 2015).



Gambar II.5. *Sequence* Diagram

(Sumber : Jurnal Informatika dan Komputer (JIKO); Vol. 1, No. 1; 2016)