BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem

Menurut Riyanto, dkk (2009 : 21-22) Dengan berbagai pendekatan, beragam pula istilah “sistem” didefinisikan., Sistem adalah suatu pengorganisasian yang saling berinteraksi, saling bergantung dan terintegrasi dalam kesatuan variabel atau komponen. Terdapat dua kelompok pendekatan sistem, yaitu menekankan pada prosedur dan komponen atau elemennya. Pendekatan sistem yang lebih menekankan pada prosedur mendefinisikan sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkelompok dan bekerjasama untuk melakukan kegiatan pencapaian sasaran tertentu. Makna dari prosedur sendiri, yaitu urutan yang tepat dari tahapan-tahapan instruksi. Sedangkan pendekatan yang menekankan pada komponen mendefinisikan sistem sebagai kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu *“Serangkaian atau tatanan elemen-elemen yang diatur untuk mencapai tujuan yang ditentukan sebelumnya melalui pemrosesan informasi”*.

II.2. Pengertian Sistem Pakar

Menurut Rika Rosnelly (2012: 10) Pakar adalah seorang individu yang memiliki pengetahuan khusus, pemahaman pengalaman dan metode-metode yang

digunakan untuk memecahakan persoalan dalam bidang tertentu. Seorang pakar memiliki kemapuan seorang kepakaran, yaitu :

1. Dapat mengenali dan merumuskan suatu masalah.
2. Menjelaskan solusi dari suatu masalah.
3. Restrukturisasi pengetahuan.
4. Belajar dari pengalaman.
5. Memahami batas kemampuan.

II.3. Gangguan (Penyakit) Gigi

Menurut Handrie Noprisson (2013 : 1) Sakit gigi biasanya merujuk kepada rasa sakit di sekitar gigi atau rahang terutama sebagai akibat dari kondisi gigi. Dalam banyak kasus, sakit gigi disebabkan oleh masalah gigi, seperti rongga gigi, gigi retak, suatu akar gigi terekspos, atau penyakit gusi. Namun, gangguan dari (bersama Temporo-mandibula) sendi rahang juga dapat menyebabkan sakit yang disebut sebagai "sakit gigi". Tingkat keparahan sakit gigi dapat berkisar dari ringan hingga kronis, tajam dan menyiksa. Rasa sakit dapat diperburuk oleh mengunyah atau dingin atau panas. Sebuah ujian lisan menyeluruh, yang mencakup gigi *X-ray*, dapat membantu menentukan apakah sakit gigi datang dari masalah gigi atau rahang dan penyebabnya.

Kadang-kadang, sakit gigi mungkin disebabkan oleh masalah yang tidak berasal dari gigi atau rahang. Sakit di sekitar gigi dan rahang dapat gejala penyakit jantung (seperti angina atau serangan jantung), telinga (seperti eksternal infeksi telinga atau bagian dalam), dan sinus (udara bagian-bagian dari tulang pipi).

Misalnya, sakit angina (pasokan darah beroksigen yang tidak memadai ke otot jantung karena penyempitan pembuluh darah ke jantung) biasanya terletak di bagian dada atau lengan. Namun, pada beberapa pasien dengan angina, rasa sakit sakit gigi atau rahang adalah satu-satunya gejala dari masalah hati mereka. Infeksi dan penyakit telinga dan sinus juga dapat menyebabkan rasa sakit di sekitar gigi dan rahang. Oleh karena itu, evaluasi oleh dokter gigi dan dokter kadang-kadang diperlukan untuk mendiagnosa penyakit medis yang menyebabkan "sakit gigi."

III.4. Forward Chaining

Menurut Rika Rosnelly (2012: 57) Forward chaining merupakan pencocokan fakta atau pernyataan dimulai dari bagian sebelah kiri (IF dulu). Dengan kata lain, penalaran dimulai dari fakta terlebih dahulu untuk menguji hipotesis.

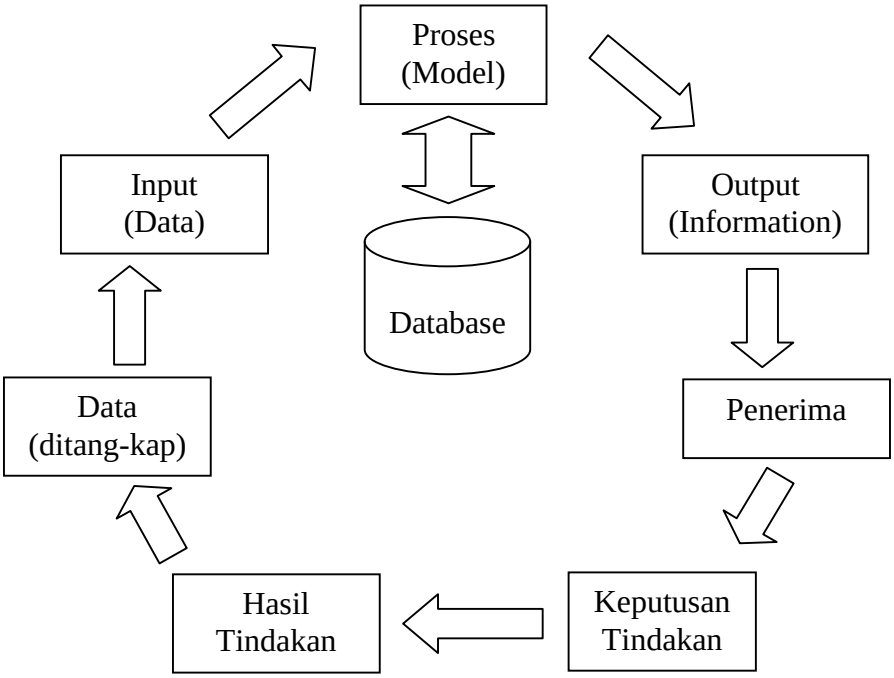
Tabel II.1. Contoh Aturan-aturan

No	Aturan
R-1	IF A & B THEN C
R-2	IF C THEN D
R-3	IF A & E THEN F
R-4	IF A THEN G
R-5	IF F & G THEN D
R-6	IF G & E THEN H
R-7	IF C & H THEN I
R-8	IF I & A THEN J
R-9	IF G THEN J
R-10	IF J THEN K

III.5. Data Dan Informasi

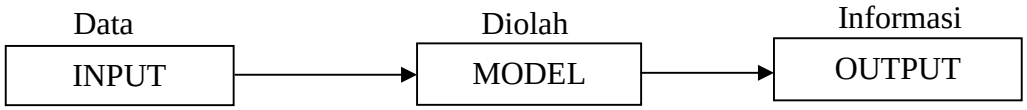
Menurut Riyanto, dkk. (2009 : 24) Data merupakan bentuk yang masih mentah yang belum dapat bercerita banyak, sehingga perlu diolah lebih lanjut.

Data diolah melalui model tertentu menjadi informasi yang dapat dimanfaatkan oleh penerima dalam membuat keputusan dan melakukan tindakan, yang berarti melakukan suatu tindakan lain yang akan membuat sejumlah data kembali. Data yang masih belum diolah akan disimpan dalam bentuk *database*. Data yang disimpan ini nantinya dapat diambil kembali untuk diolah kembali menjadi informasi. Data tersebut akan ditangkap sebagai input, diproses kembali lewat suatu model tertentu dan seterusnya membentuk suatu siklus. Siklus ini oleh *John Burch* disebut dengan siklus informasi (*information cycle*).



Gambar II.1. Siklus Informasi
(Sumber : Riyanto, dkk. : 2009 : 24)

Menurut Riyanto, dkk. (2009 : 23) Agar menjadi informasi yang berguna, data perlu diolah melalui sebuah siklus. Siklus ini disebut siklus pengolahan data (*data processing life cycle*).



Gambar II.2. Siklus Pengolahan Data
(Sumber : Riyanto, dkk. : 2009 : 23)

II.6. Unified Modelling Language

Menurut Adi Nugroho (2009 : 5) Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan. Adapun tujuan pemodelan (dalam kerangka pengembangan sistem/perangkat lunak aplikasi) adalah sebagai sarana visualisasi dan komunikasi antar anggota tim pengembang (saat seorang analis/perancang sistem/perangkat lunak bekerja dalam tim yang beranggotakan beberapa/banyak anggota) serta sebagai sarana dokumentasi (bermanfaat untuk menelaah perilaku sistem secara seksama dan bermanfaat untuk menguji (*testing*) sistem yang telah selesai dikembangkan).

Dalam hal ini sebagai perancang sistem/perangkat lunak menggambarkan komponen-komponen sistem/perangkat lunak dalam bentuk-bentuk geometri tertentu, misalnya untuk menggambarkan suatu kelas (*class*) dalam aplikasi membentuk empat persegi panjang, untuk menggambarkan hubungan antarkelas menggunakan garis lurus.

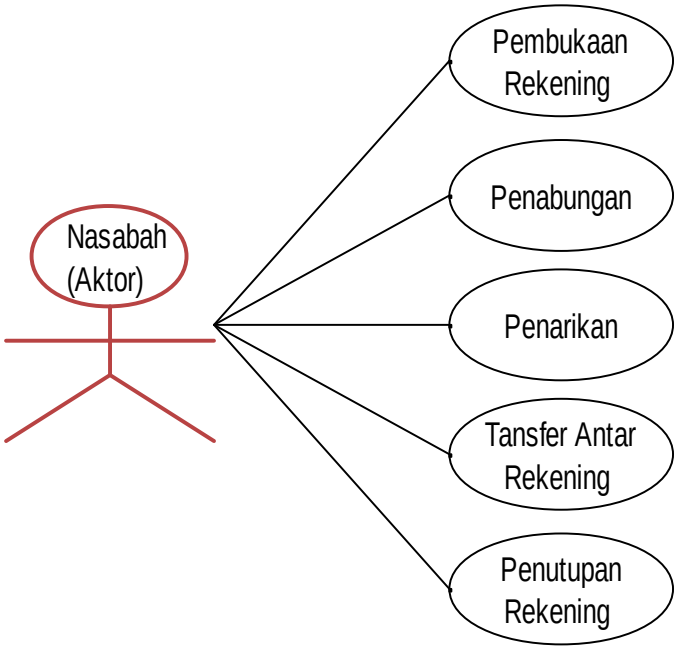
II.6.1. Use Case Diagram

Menurut Adi Nugroho (2009 : 7) Segala sesuatu yang secara akademis dikembangkan pada umumnya berawal dari suatu konsep. Analisis kebutuhan ini adalah tahap konseptualisasi, yaitu suatu tahap yang mengharuskan analis dan

perancang sistem untuk berusaha tahu secara pasti mengenai hal yang menjadi kebutuhan dan harapan pengguna sehingga kelak aplikasi yang dibuat memang akan digunakan oleh pengguna (*user*) serta akan memuaskan kebutuhan dan harapannya.

Selanjutnya, *use case diagram* tidak hanya sangat penting pada saat analisis, tetapi juga sangat penting dalam tahap perancangan (*design*), untuk mencari kelas-kelas yang terlibat alam aplikasi, dan untuk melakukan pengujian (*testing*).

Saat akan mengembangkan *use case diagram*, hal yang pertama kali harus dilakukan adalah mengenali *actor* untuk sistem yang sedang dikembangkan. Dalam hal ini, ada beberapa karakteristik untuk para *actor*, yaitu *actor* yang ada di luar sistem yang sedang dikembangkan dan *actor* yang berinteraksi dengan sistem yang sedang dikembangkan dapat dilihat pada gambar II. 3 berikut ini :



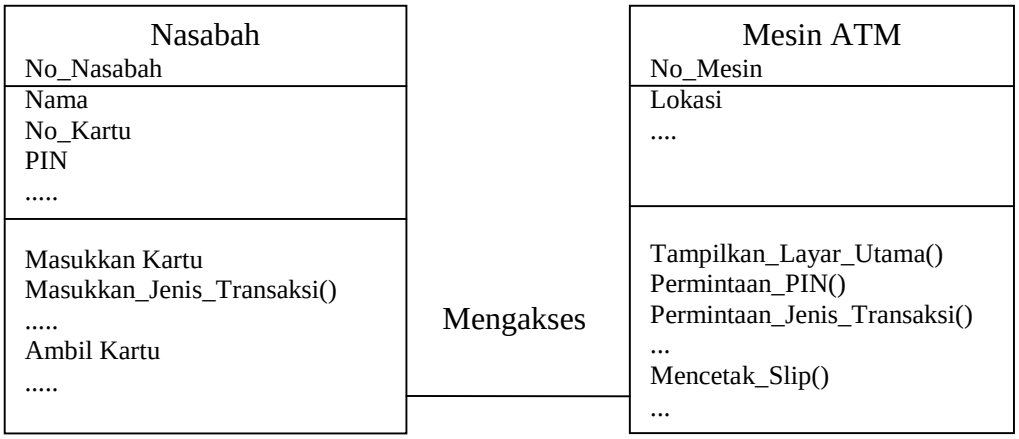
Gambar II.3. Contoh Use Case Diagram
(Sumber : Adi Nugroho ; 2009 : 8)

II.6.2. Class Diagram

Menurut Adi Nugroho (2009 : 18) *Class* didefenisikan sebagai kumpulan/himpunan objek yang memiliki kesamaan dalam atribut/properti, perilaku (operasi), serta cara berhubungan dengain objek lain.

Menurut Adi Nugroho (2009 :17) Selain itu, juga mendefenisikan objek sebagai konsep, abstraksi dari sesuatu dengan batas nyata, sehingga dapat menggambarkan secara sistematis. Pemahaman objek memiliki dua fungsi, yaitu :

1. Memudahkan untuk mempelajari secara seksama hal-hal yang ada di dunia nyata.
2. Menyediakan suatu dasar yang kuat dalam implementasi ke dalam sistem terkomputerisasi dapat dilihat pada gambar II. 4 berikut ini :

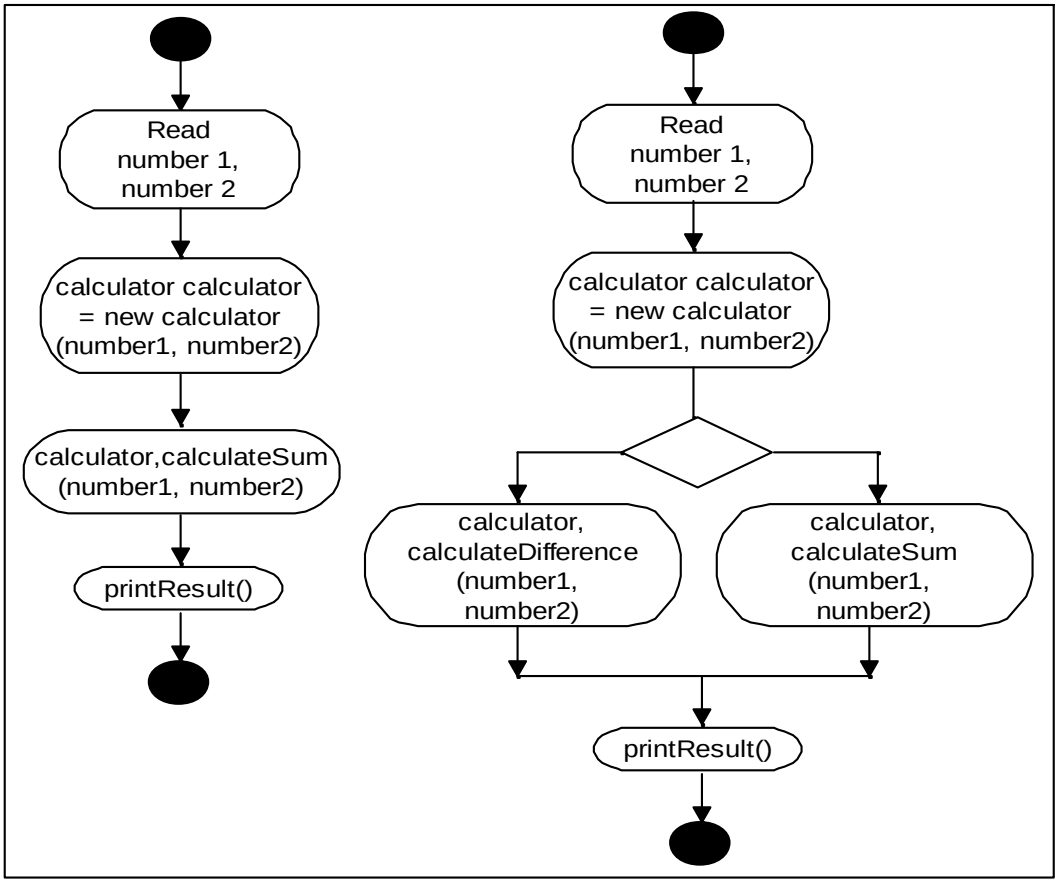


Gambar II.4. Contoh Class Diagram
(Sumber : Adi Nugroho ; 2009: 39)

II.6.3. Activity Diagram

Menurut Adi Nugroho (2009 : 10) Apakah langkah yang harus lakukan selanjutnya setelah membuat use case diagram, use case diagram merupakan gambaran menyeluruh dan pada umumnya sangatlah tidak terperinci. Oleh karena

itu, harus memperinci lagi perilaku sistem untuk masing-masing use case yang ada. Apa perkakas (tool) yang bisa gunakan, jika kasus cukup sederhana, mungkin bisa menggunakan skenario seperti yang tercantum berikut, sementara jika kasusnya cukup kompleks, mungkin bisa menggunakan activity diagram agar bisa mendapatkan gambaran yang lebih menyeluruh dapat dilihat pada gambar II. 5 berikut ini :



Gambar II.5. Contoh Activity Diagram
(Sumber : Adi Nugroho ; 2009 : 74)

II.6.4. *Sequence Diagram*

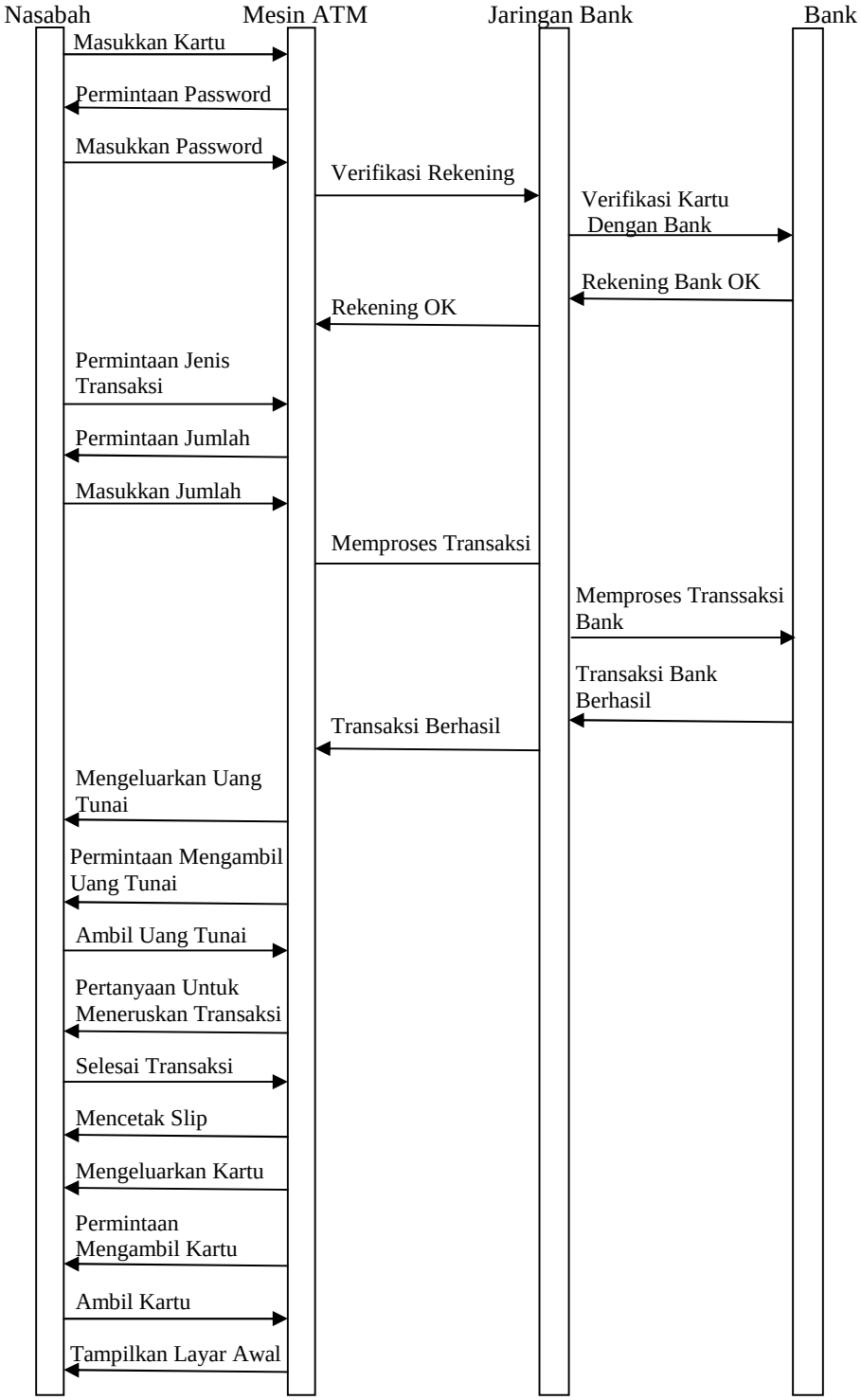
Diagram sekuensial atau *sequence diagram* digunakan untuk menunjukkan aliran fungsionalitas dalam *use case*. Diagram sekuensial adalah

diagram yang disusun berdasarkan urutan waktu. Membaca diagram sekuensial dari atas ke bawah. Setiap diagram sekuensial mempresentasikan suatu aliran dari beberapa aliran di dalam *use case*.

Jadi dengan kata lain sekuensial diagram menunjukkan aliran fungsionalitas berdasarkan urutan waktu serta kejadian yang nantinya akan menetukan metode/fungsi atribut masing-masing. Dimana fungsi-fungsi tersebut akan diterapkan pada suatu kelas/objek.

Perhatikan gambar II.6. dimana terlihat pengelompokkan *event-event* serta fungsi masing-masing atribut tersebut. Di dalam diagram terlihat jelas bagaimana aliran suatu proses kejadian dimana seorang nasabah yang akan melakukan transaksi dengan sebuah mesin ATM. Dari diagram tersebut mengetahui *event-event* yang terjadi, seperti : Nasabah memasukkan kartu ATM, Mesin ATM merespon dengan meminta *password* atau PIN, dan selanjutnya.

Dapat melihat setiap fungsi atribut dan *event-event* apa saja yang terjadi. Sehingga melalui diagram sekuensial ini, dapat merancang suatu program aplikasi yang baik, sehingga dalam menghadapi sebuah kasus yang benar-benar kompleks diagram sekuensial ini sangat membantu. Gambar contoh *sequence diagram* dapat dilihat pada gambar II. 6 berikut ini :



Gambar II.6. Contoh Sequence Diagram
(Sumber : Adi Nugroho ; 2009 : 36)

II.7. Microsoft Visual Basic

Menurut Wahana Komputer (2010:2) *Visual basic* merupakan salah satu bahasa pemograman yang andal dan banyak digunakan oleh pengembang untukmembangun berbagai macam aplikasi *windows*. *Visual basic 2008* merupakan aplikasi pemograman yang menggunakan teknologi. *NET Framework 3.5*. Teknologi. *NET Framework 3.5* merupakan komponen *windows* yang terintegrasi serta mendukung pembuatan, penggunaan aplikasi, dan halaman *web*. Teknologi .*Net Framework 3.5* mempunyai 2 komponen utama, yaitu *CLR (Common Language Runtime)* dan *Class Library*. *CLR* digunakan untuk menjalankan aplikasi yang berbasis.*NET*, sedangkan Library adalah kelas pustaka atau perintah yang digunakan untuk membangun aplikasi.

II.8. Microsoft SQL Server

Menurut Ema Utami dan Anggi Dwi Hartanto (2012 : 63) Bahasa query merupakan bahasa khusus yang digunakan untuk melakukan manipulasi dan menanyakan pertanyaan *(query)* yang berhubungan dengan bahasa pemrograman, dimana bahasa *query* tidak memiliki kemampuan untuk menyelesaikan banyak masalah seperti bahasa pemrograman pada umumnya. Dalam pemrograman basis data, salah satu bahasa yang harus di kuasai adalah SQL. SQL merupakan bahasa komputer standar yang digunakan untuk berkomunikasi dengan sistem manajemen basis data relasional (RDBMS).

II.9. *Database* Dan *ERD*

Menurut Yuniar Supardi (2008 : 9) *Desain database* merupakan pekerjaan yang penting dalam pembuatan atau pengembangan sistem, karena desain *database* akan mendapatkan susunan data atau *table* yang efektif dan efisien. Alat desain *database* yang popular ada dua, yaitu : ERD (*Entity Relationship Diagram*) dan Normalisasi. Jika memakai *Normalisasi* harus mendapatkan Data Dasar (Dokumen Dasar), sedangkan ERD tidak perlu. Dalam desain ERD terbagi dua tahapan yaitu: *Preliminary* Desain (Disain Awal) dan *Final Design* (Disain Akhir). Tetapi disain Akhir dari ERD juga berisi Normalisasi.

II.10. Data *Dictionary* (kamus data)

Menurut Budi Raharjo (2011 ; 59) Dalam suatu rancangan *database*, data *dictionary* digunakan untuk menjelaskan atau mendeskripsikan kolom-kolom pada masing-masing tabel yang akan dibuat ke dalam *database*. Deskripsi kolom yang dimaksud di sini meliputi tipe data, lebar karakter atau digit, serta keterangan tentang kunci relasi

II.11. Normalisasi

Menurut Yuniar Supardi (2008 : 10) tahapan normalisasi terdiri dari beberapa bentuk, yaitu:

1. Bentuk Tak Normal (UNF / *Un Normal Form*).
2. Bentuk Normal Pertama (1 NF / *First Normal Form*).

Bentuk Normal pertama memiliki ciri: Data berbentuk *file-file* (*file* datar), *record* disusun sesuai kedatangan, masih mungkin terjadi penyimpangan data (anomali data). Anomali data dapat berupa insert *anomali, delete anomali, update, anomali, dan redundancy data* (data duplikat).

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

   Bentuk Normal kedua memiliki ciri; Tidak terjadi anomali data, setiap *field*/atribut bukan kunci harus tergantung fungsi (*Functional Depedency*) terhadap *field*/atribut kunci, masih mungkin terjadi *transitive dependency* (*field* bukan kunci tergantung pada *field* bukan kunci dalam satu *table*). Model objek mencapai bentuk normal kedua, sehingga penulis mendesain mulai bentuk normal ketiga dan bentuk normal boyce codd. Sedangkan untuk bentuk tak normal sudah dari dokumen dasar berupa Faktur, Nota, dan laporan *Stock of Name*.

4. Bentuk Normal Ketiga (3 NF / *Third Normal Form*).

   Table yang memenuhi Bentuk Normal Ketiga harus tidak terdapat *Transitive Depedency*. Bentuk normal ketiga dari sistem *inventory* :

5. Bentuk Normal *Boyce Codd* (BCNF / *Boyce Codd Normal Form*).

   Karena tak ada *field* bukan kunci tergantung secara parsial (bagian) kunci dalam satu tabel, maka bentuk normal ketiga juga merupakan bentuk BCNF.

6. Normal yang lebih tinggi.

Setelah 3NF, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Bentuk Normal Boyce-Code (BCNF) adalah versi 3NF yang lebih teliti dan berhubungan dengan tabel relasional yang mempunyai banyak kunci kandidat, kunci kandidat gabungan, dan kunci kandidat yang saling tumpang tindih.