BAB II

LANDASAN TEORI

Suatu sistem adalah jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu.

Suatu sistem yang baik harus mempunyai tujuan dan sasaran yang tepat karena hal ini akan sangat menentukan dalam mendefinisikan masukan yang dibutuhkan sistem dan juga keluaran yang dihasilkan.

2.1. Konsep Dasar Sistem

Terdapat dua kelompok pendekatan di dalam pendefinisian sistem, yaitu yang menekankan pada prosedur dan yang menekankan pada komponen atau elemen–elemen. Pendekatan sistem yang lebih menekankan pada prosedur mendefinisikan sistem sebagai berikut:

> *"Suatu sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu"*. **[JOG99]**

Pendekatan sistem yang lebih menekankan pada komponen atau elemen-elemennya mendefinisikan sistem sebagai berikut:

"Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu". **[JOG99]**

2.2. Konsep Dasar Informasi

Informasi dapat diibaratkan sebagai darah yang mengalir di dalam tubuh manusia, seperti halnya informasi di dalam sebuah perusahaan yang sangat penting untuk mendukung kelangsungan perkembangannya, sehingga terdapat alasan bahwa informasi sangat dibutuhkan bagi sebuah lembaga pendidikan.

Informasi dapat didefinisikan sebagai berikut:

"Informasi adalah data yang yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya". **[JOG99]**

Dari pengertian di atas dapat disimpulkan informasi merupakan kumpulan data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerima. Tanpa suatu informasi, suatu sistem tidak akan berjalan dengan lancar dan akhirnya bisa mati. Suatu organisasi tanpa adanya informasi maka organisasi tersebut tidak bisa berjalan dan tidak bisa beroperasi.

2.3. Konsep Dasar Sistem Informasi

Telah diketahui bahwa informasi merupakan hal yang sangat penting bagi manajemen dalam pengambilan keputusan. Sistem informasi yang baik akan menciptakan komunikasi informasi yang dibutuhkan dan informasi harus memiliki nilai yang tinggi, yaitu informasi tersebut harus memenuhi kebutuhan

akan pemakai informasi tersebut pada berbagai tingkat organisasi dan dapat mengurangi faktor ketidakpastian.

Sistem informasi didefinisikan oleh Robert A. Leitch dan K. Roscoe Davis sebagai berikut:

> *"Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan suatu kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luat tertentu dengan laporan-laporan yang diperlukan"*. **[JOG99]**

2.3.1. Komponen Sistem Informasi

Untuk mendukung lancarnya suatu sistem informasi dibutuhkan beberapa komponen yang fungsinya sangat vital di dalam sistem informasi. Komponen-komponen sistem informasi tersebut adalah sebagai berikut:

1. Masukan (*Input*)

   Input disini adalah semua data yang dimasukan ke dalam sistem informasi. Dalam hal ini yang termasuk dalam *input* adalah dokumen-dokumen, formulir-formulir dan file-file.

2. Proses

   Proses merupakan kumpulan prosedur yang akan memanipulasi *input* yang kemudian akan disimpan dalam bentuk basis data dan seterusnya akan diolah menjadi suatu *output* yang akan digunakan oleh penerima.

3. Keluaran (*Output*)

   Output merupakan semua keluaran atau hasil dari model yang sudah diolah menjadi suatu informasi yang berguna dan dapat dipakai penerima.

4. Teknologi

   Teknologi disini merupakan bagian yang berfungsi untuk memasukan *input*, mengolah *input* dan menghasilkan *output*. Ada 3 bagian dalam teknologi ini yang meliputi perangkat keras, perangkat lunak dan perangkat manusia.

5. Basis data

   Basis data merupakan kumpulan data-data yang saling berhubungan satu dengan yang lain yang disimpan dalam perangkat keras komputer dan akan diolah menggunakan perangkat lunak.

6. Kendali

   Kendali dalam hal ini merupakan semua tindakan yang diambil untuk menjaga sistem informasi tersebut agar bisa berjalan dengan lancar dan tidak mengalami gangguan.

2.4. Perangkat Lunak

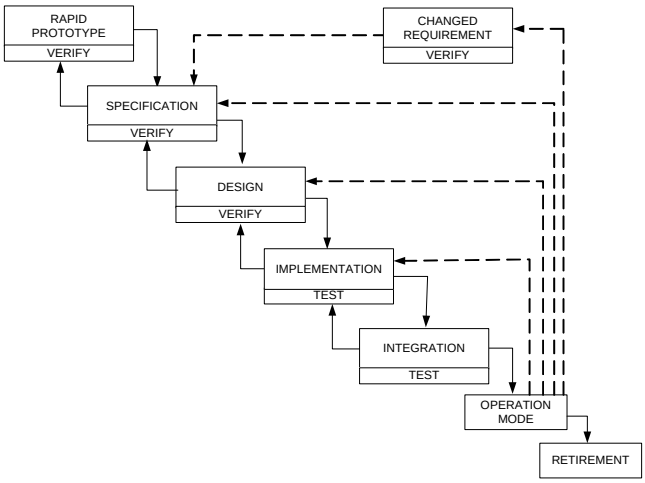
Perangkat Lunak adalah instruksi (program komputer) yang ketika dijalankan menyediakan fungsi dan tampilan yang diinginkan, struktur data yang memberikan kesempatan program untuk memanipulasi informasi dan dokumen yang mendeskripsikan operasi dan penggunaan program

2.4.1. Model Proses Perangkat Lunak

Terdapat beberapa model proses rekayasa perangkat lunak, diantaranya:

1. Model Waterfall

Model Waterfall berisi rangkaian aktivitas proses seperti spesifikasi kebutuhan, implementasi desain perangkat lunak, uji coba dan sebagainya diuraikan dan disajikan dalam proses yang terpisah. Setelah setiap langkah didefinisikan, langkah tersebut dihentikan, dan pengembangan dilanjutkan pada langkah berikutnya. Gambar ilustrasi dari Model Waterfall adalah sebagai berikut:

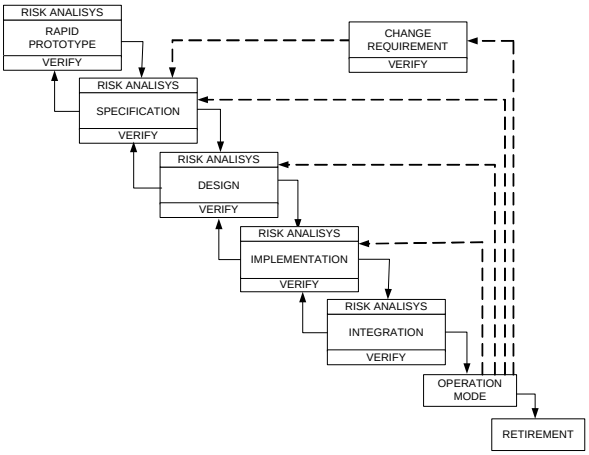


Gambar 2.1. Model *Waterfall*

2. Model *Spiral* Boehm

Model Spiral Boehm merupakan kombinasi antara *prototyping* model dan waterfall model. Setiap tahapan model ini selalu dilakukan *Risk*

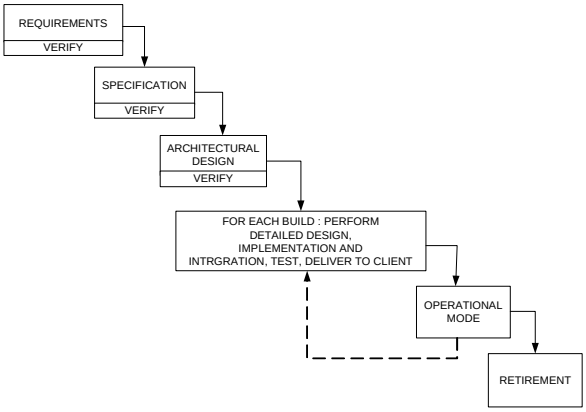
Analisys dan verifikasi atau testing. Gambar ilustrasi model Spiral Boehm adalah sebagai berikut:



Gambar 2.2. Model *Spiral* Boehm

3. Model Incremental

Model Incremental merupakan rekayasa perangkat lunak pembagian, hingga menghasilkan perangkat lunak yang lengkap. Proses membangun berhenti jika produk telah mencapai fungsi yang diharapkan. Gambar ilustrasi model Incremental adalah sebagai berikut:



Gambar 2.3. Model Incremental

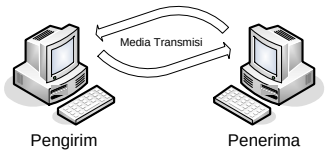
2.5. Jaringan Komputer

Jaringan komputer adalah sekelompok komputer otonom yang yang dihubungkan satu dengan yang lainnya dengan menggunakan protokol komunikasi melalui media transmisi atau media komunikasi sehingga dapat saling berbagi data, program-program, penggunaan bersama perangkat keras seperti *printer*, *hard disk* dan sebagainya.

2.5.1 Prinsip Dasar Jaringan Komputer

Prinsip dasar dalam sistem jaringan ini adalah proses pengiriman data atau informasi dari pengirim ke penerima melalui suatu media komunikasi tertentu.

Tujuan dibangunnya suatu jaringan komputer adalah untuk membawa data atau informasi dari sisi pengirim menuju penerima secara cepat dan tepat tanpa adanya kesalahan melalui media transmisi atau media komunikasi tertentu.



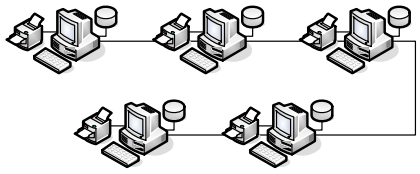
Gambar 2.4. Prinsip Dasar Jaringan Komputer

2.5.2. Arsitektur Jaringan Komputer

Sistem operasi jaringan sangat menentukan banyak arsitektur jaringan yang dibangun. Ada tiga macam arsitektur jaringan, yaitu *peer to peer*, *file server*, dan *client server*.

1. *Peer to peer*

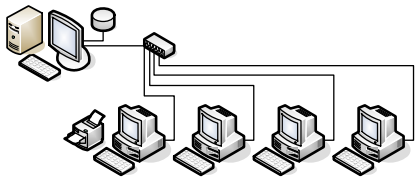
   Pada bentuk konektivitas *peer to peer*, setiap terminal memiliki peran dan derajar yang sama. Jaringan lokal dengan konektivitas *peer to peer* ini dibentuk dengan cara menghubungkan setiap terminal secara langsung sehingga masing-masing terminal dapat berbagi data, aplikasi dan *peripheral* lainnya. Pada konektivitas ini semua terminal dapat bertindak sebagai workstation atau *server*. Biasanya arsitektur ini digunakan pada perusahaan berskala kecil sampai menengah yang sudah terlanjur memiliki komputer-komputer personal antara 5 sampai 10 buah, dimana masing-masing komputer memiliki kelengkapan perangkat keras seperti *hard disk*, *memori*, *peripheral* lain, serta perangkat lunak seperti aplikasi-aplikasi perkantoran, atau sebuah sistem informasi perusahaan.



Gambar 2.5. Konektivitas *Peer to peer*

2. *File Server*

   Pada sistem *file server*, terdapat terminal khusus yang disebut sebagai *server* yang memiliki kapasitas hard disk yang sangat besar. *Server* tersebut akan bertindak sebagai tempat penyimpanan (file) bersama, namun tidak ada pelayanan komputasi.

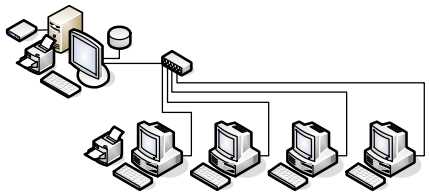


Gambar 2.6. Konektivitas *File Server*

3. *Client Server*

   Arsitektur jaringan *client server* merupakan pengembangan dari arsitektur *file server*. Arsitektur ini adalah model konektivitas pada jaringan yang mengenal adanya *server* dan *client*, dimana masing-masing memiliki fungsi yang berbeda satu sama lain. *Server* dapat berbagi pakai data, aplikasi dan *peripheral* seperti *harddisk, printer, modem* dan lain-lain. Oleh karena itu, tidak jarang juga tercipta sebutan *print server*, *communication server* dan lain sebagainnya. Prinsip kerjanya sangat

sederhana, dimana *server* akan menunggu permintaan dari *client*, memproses dan memberikan hasilnya kepada *client*. Sedangkan *client* akan mengirimkan permintaan ke *server*, menunggu proses dan melihat visualisasi hasil prosesnya.



Gambar 2.7. Konektivitas *Client Server*

Sistem *client server* ini menggunakan protocol TCP/IP (*Transmission Control Protocol/Internet Protocol*). Unix dan Windows NT merupakan contoh yang baik dari sistem operasi jaringan *client server*.

2.5.3. Tipe Jaringan Client Server

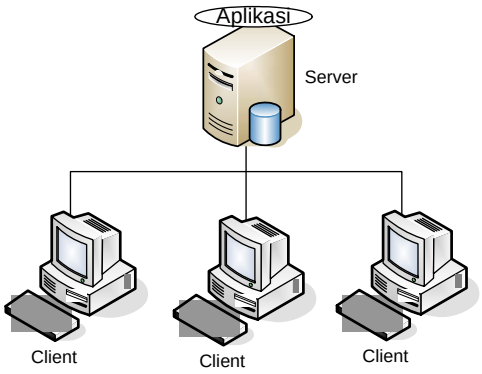
Berdasarkan tipe layanan yang diberikan *server* kepada *client*, jaringan *Client Server* dapat dibagi ke dalam banyak tipe, antara lain *server* berkas, *server* basis data, *server* transaksi, *server* groupware, *server* objek, dan *server* Web.

2.5.4 Arsitektur Client Server

1. Two Tier

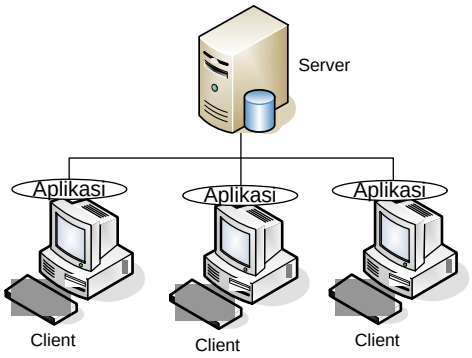
Arsitektur *two tier* merupakan arsitektur yang disebut Client Server dimana terdapat komputer sebagai client dan server yang berinteraksi melalui protokol dan media komunikasi tertentu Model arsitektur *Two Tier* dikelompokan menjadi dua macam yaitu:

a. Thin Client – Thick Server



Gambar 2.8 *Thin Client – Thick Server*

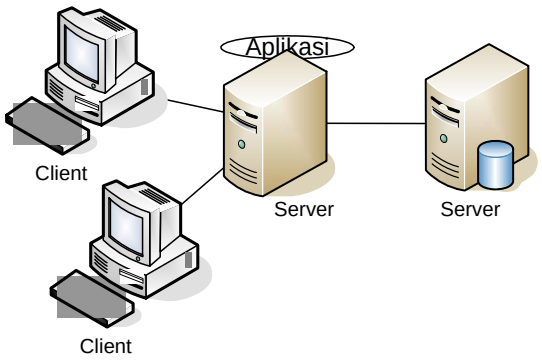
b. Thick Client – Thin Server



Gambar 2.9 *Thick Client – Thin Server*

2. Three Tier

Arsitektur Client Server ini memisahkan antara data (*Data Management Tier*), aplikasi (*Middle Tier*) dan penyajian (*Presentation Layer*) sebagaimana ditampilkan pada gambar dibawah ini :



Gambar 2.10 ***Arsitektur Three Tier***

a. Data Management Tier

Merupakan komputer server yang dikhususkan untuk menangani pengolahan basis data.

b. Middle Tier

Merupakan komputer server yang dikhususkan untuk menangani aplikasi-aplikasi dimana prosedur-prosedur dan perhitungan-perhitungan yang kompleks dilakukan di komputer.

c. Presentation Layer

Merupakan komputer client yang menjadi *interface* bagi pengguna untuk memasukkan data, mengajuan permintaan layanan kepada server dan melihat hasilnya..

3. n-Tier

Istilah *n-tier* menunjukan lapisan yang ada dalam sebuah aplikasi. Sebuah aplikasi terdiri dari beberapa komponen utama, yaitu lapisan *Presentation* (*Presentation Layer*), lapisan *application* (*Application Layer*) dan logika bisnis (*bussiness logic layer*) dan lapisan data (*Data layer*).

Lapisan *Presentation* menghubungkan antarmuka dengan pengguna aplikasi, dapat berupa model grafis atau berupa teks. Pengguna dapat berinteraksi dengan aplikasi tersebut menggunakan lapisan *presentation* ini. Lapisan *Application* berisi inti dari aplikasi dan lapisan data yang digunakan oleh aplikasi tersebut. Lapisan data dapat berbentuk satu atau lebih server basis data yang lokasinya tersebar dibeberapa tempat.

2.6. *Unified Modelling Language* (UML)

Unified Modelling Language (UML) adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, Sistem Operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut

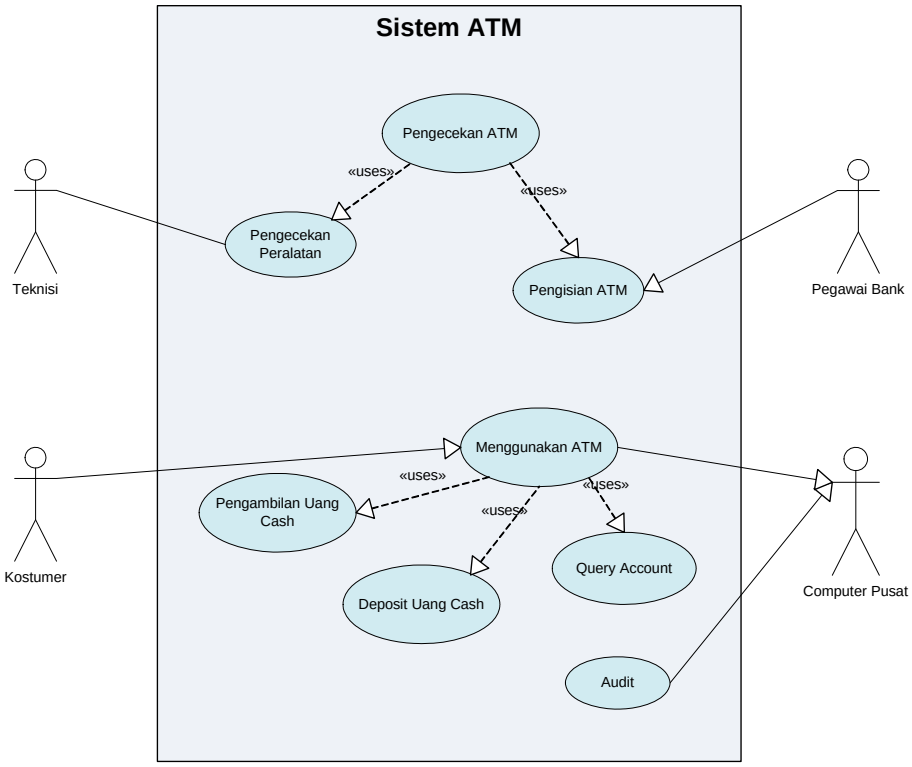
dapat dikombinasikan. Notasi UML terutama diturunkan dari tiga notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

2.6.1 Konsepsi Dasar UML

Abstraksi konsep dasar UML yang terdiri dari *structural classification*, *dynamic behavior*, dan *model management*, bisa kita pahami dengan mudah apabila kita melihat gambar diatas dari *Diagrams*. *Main concepts* bisa kita pandang sebagai *term* yang akan muncul pada saat kita membuat diagram. Dan *view* adalah kategori dari diagaram tersebut. Dalam UML mendefinisikan berbagai macam diagram, yaitu sebagai berikut:

a. *Use Case Diagram*

   Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. Berikut contoh dari *use case diagram :*



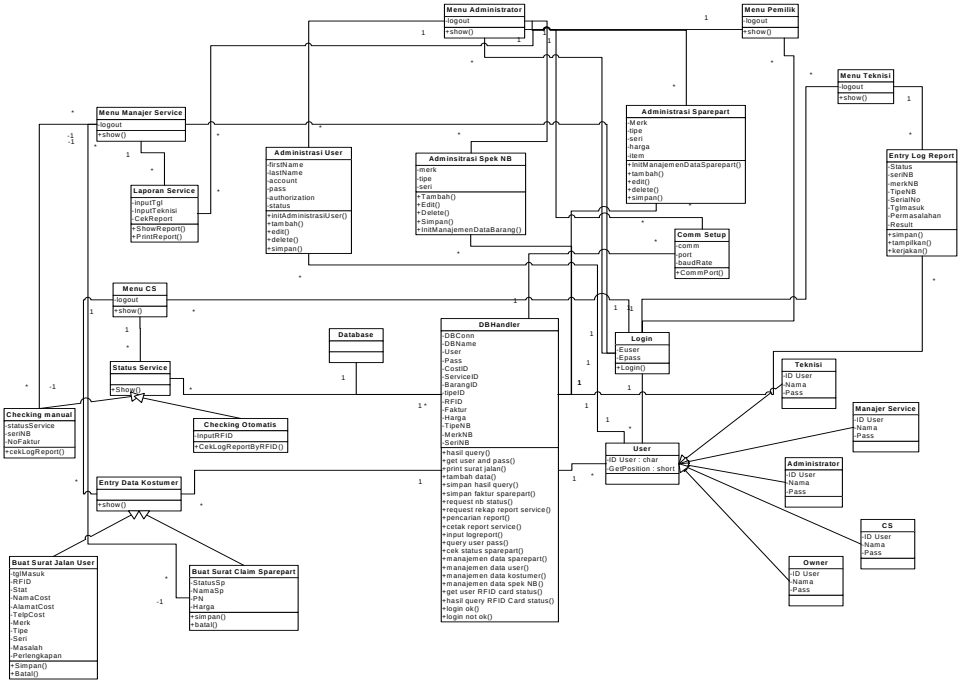
Gambar 2.11 *Use Case Diagram*

b. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan dan lain-lain. *Class* memiliki tiga area pokok

a. Nama (dan *stereotype*)
b. Atribut
c. Metode

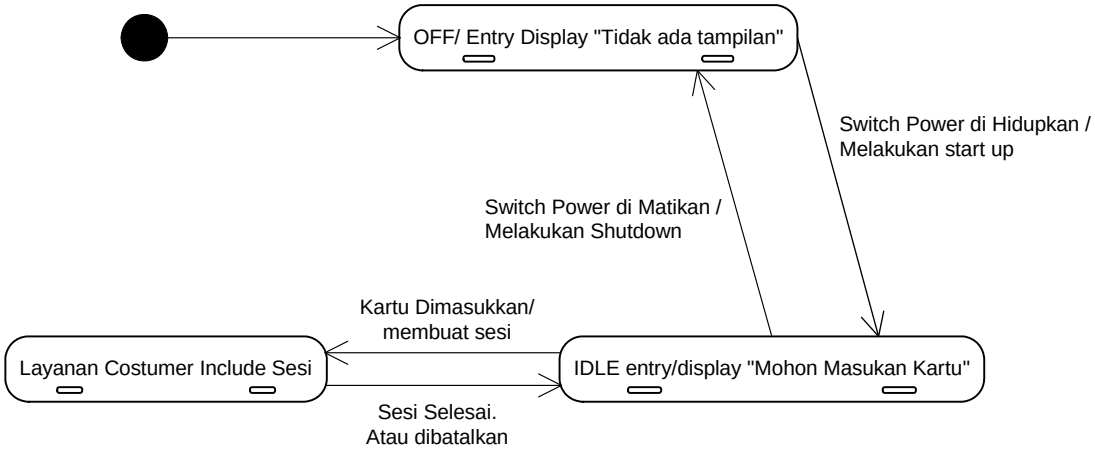
Berikut ini contoh gambar *class diagram*:



Gambar 2.12 Class Diagram

c. Statechart Diagram

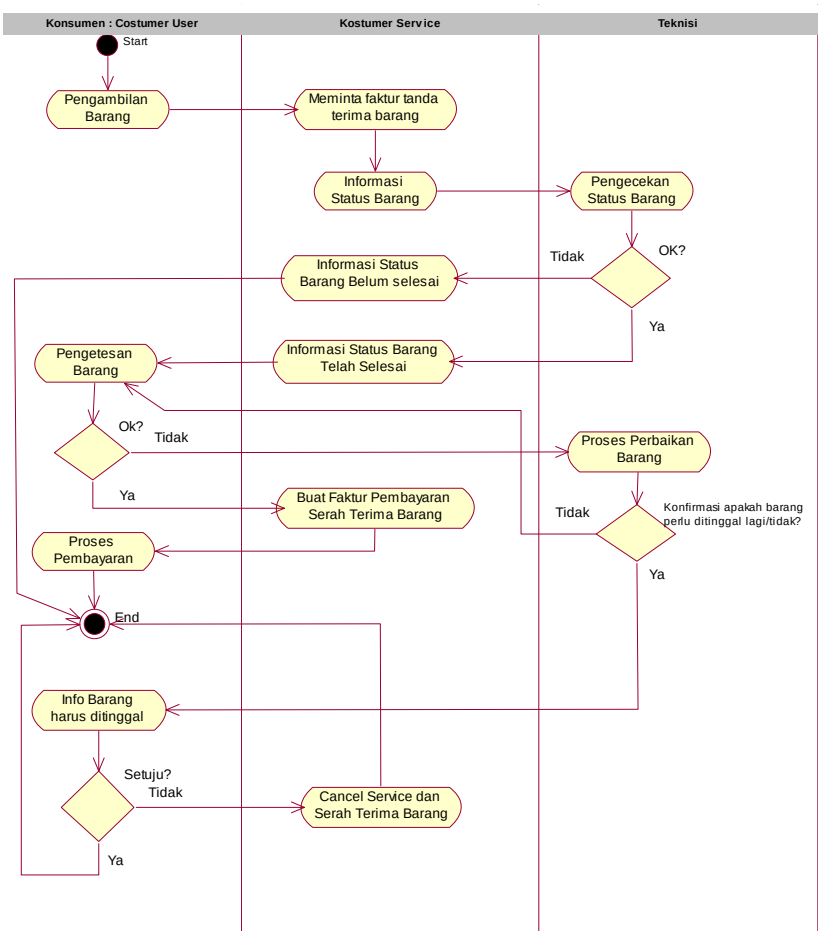
Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimulus* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*). Berikut ini contoh gambar dari *statechart diagram*:



Gambar 2.13 *Statechart Diagram*

d. Activity Diagram

Activity diagram merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). *Activity diagrams* 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 Sistem Operasi paralel yang mungkin terjadi pada beberapa eksekusi. Contoh *activity diagram* tanpa *swimlane*:

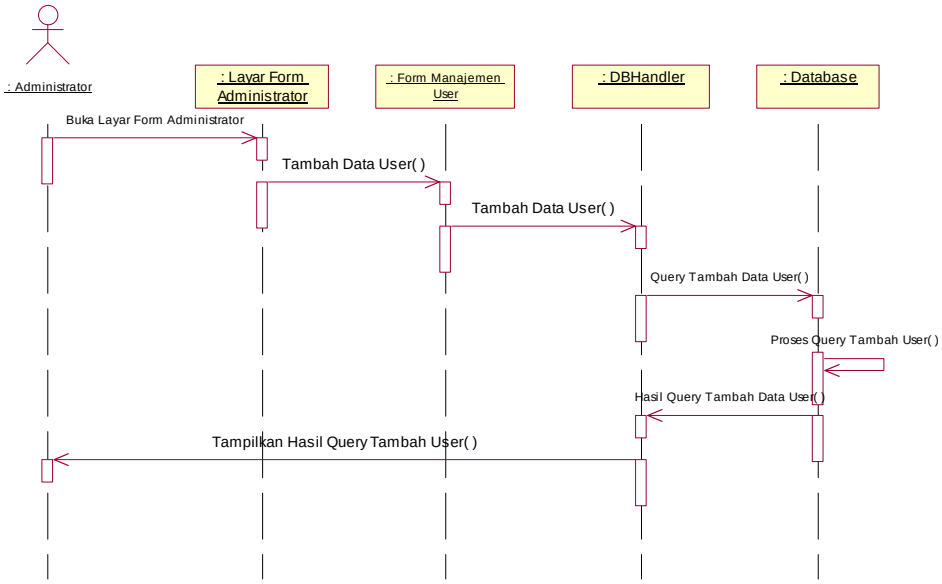


Gambar 2.14 *Activity Diagram*

e. Sequence Diagram

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 atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon

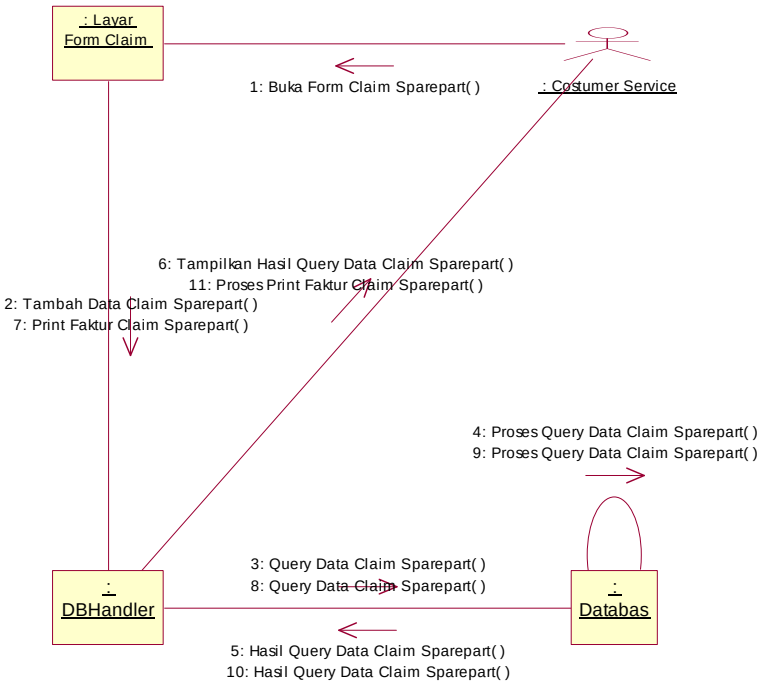
dari sebuah *event* untuk menghasilkan *output* tertentu. Contoh *sequence diagram*



Gambar 2.15 Sequence Diagram

f. Collaboration Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. *Messages* dari level yang sama memiliki *prefiks* yang sama.

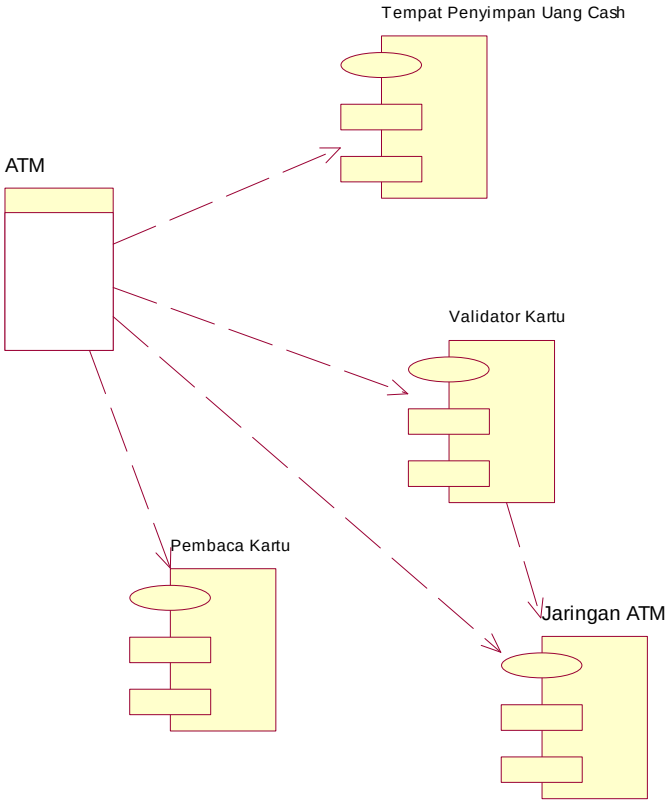


Gambar 2.16 *Collaboration Diagram*

g. Component Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

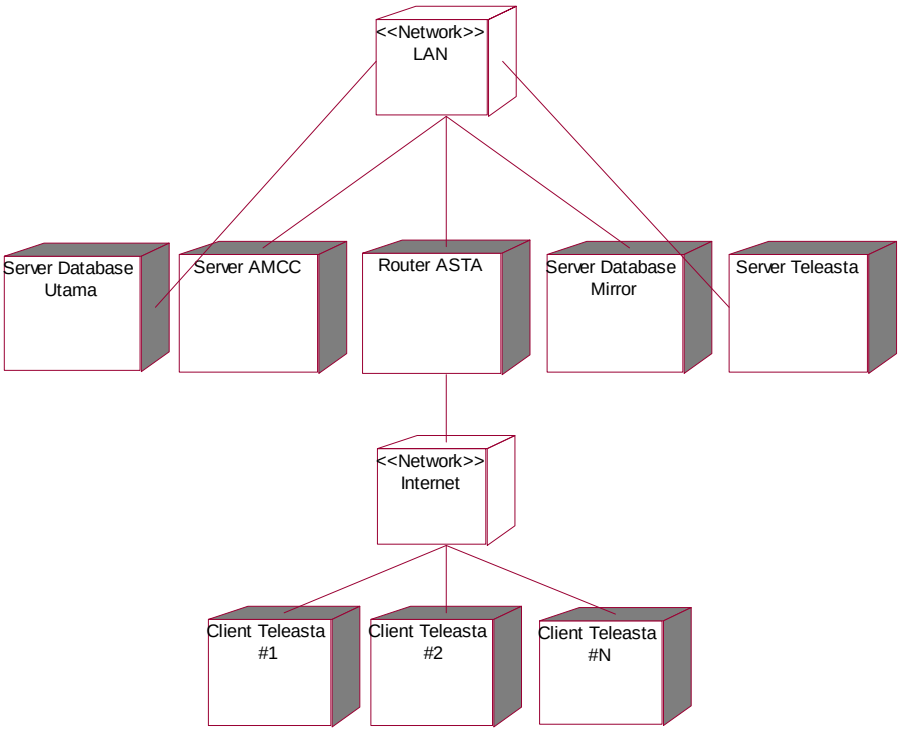
Contoh *component diagram*:



Gambar 2.17 *Component Diagram*

h. Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.



Gambar 2.18 *Deploy Diagram*

2.7. Database

Database adalah suatu koleksi data komputer yang terintegrasi, diorganisasikan dan disimpan dalam suatu cara yang memudahkan pengambilan kembali. Dua tujuan utama dari database adalah meminimumkan pengulangan dan mencapai *independensi* data. *Independensi* data adalah kemampuan untuk membuat perubahan dalam struktur data tanpa membuat perubahan pada program yang memproses data.

Database atau basis data terdiri dari dua kata yaitu basis yang artinya markas atau gudang atau tempat berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek.

2.7.1. Abstraksi Data

Abstraksi data merupakan tingkatan-tingkatan pengguna dalam memandang bagaimana sebenarnya data diolah dalam sebuah sistem database sehingga menyerupai kondisi yang sebenarnya dihadapi oleh pengguna sehari-hari. Sebuah DBMS seringkali menyembunyikan detail tentang bagaimana sebuah data disimpan dan dipelihara (diolah) dalam sebuah sistem database, dengan tujuan untuk memudahkan pengguna dalam menggunakan DBMS tersebut. Karena itu seringkali data yang terlihat oleh pemakai sebelumnya berbeda dengan yang tersimpan secara fisik. Ada tiga tingkatan atau level dalam abstraksi data ini, yaitu :

1. Level Fisik (*Physical Level*)

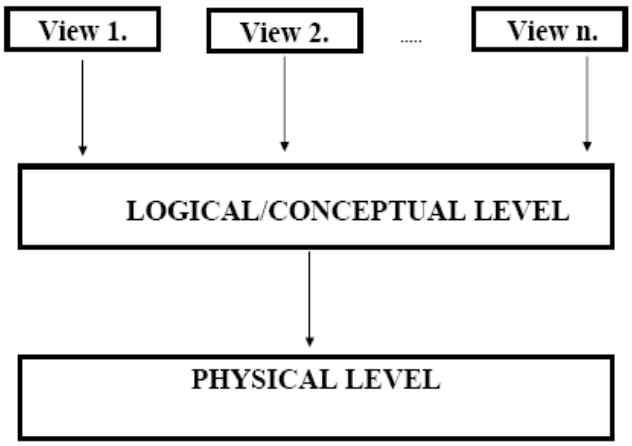
   Level abstraksi data yang paling rendah, yang menggambarkan bagaimana (how) data disimpan dalam kondisi sebenarnya. Level ini sangat kompleks karena struktur data dijelaskan secara rinci.

2. Level Konseptual (*Conceptual Level*)

   Level ini menggambarkan data apa yang disimpan dalam database dan menjelaskan bagaimana hubungan antar data secara keseluruhan. Seorang pengguna dalam level ini dapat mengetahui bahwa data mahasiswa disimpan pada tabel mahasiswa, tabel krs, tabel transkrip dan lain sebagainya. Level ini biasa di pakai oleh seorang Database Administrator (DBA).

3. Level Pandangan (*View Level*)

   Ini merupakan level yang tertinggi, hanya menggambarkan sebagian saja dari keseluruhan database sesuai dengan kebutuhan pengguna. Misalnya Bagian keuangan hanya membutuhkan data keuangan, jadi yang digambarkan hanya pandangan terhadap data keuangan saja, begitu juga dengan bagian akuntansi, hanya membutuhkan data akuntansi saja. Jadi tidak semua pengguna database membutuhkan seluruh informasi yang terdapat dalam database tersebut.



Gambar 2.19 Abstraksi Data

2.7.2. Model Database

Model database adalah suatu konsep yang terintegrasi dalam menggambarkan hubungan (*relationships*) antar data dan batasan-batasan (*constraint*) data dalam suatu sistem database. Model data yang paling umum, berdasarkan pada bagaimana hubungan antar *record* dalam database (*Record Based Data Models*), terdapat tiga jenis, yaitu :

1. Model Database Hierarki (*Hierarchical Database Model*)
2. Model Database Jaringan (*Network Database Model*)
3. Model Database Relasi (*Relational Database Model*)

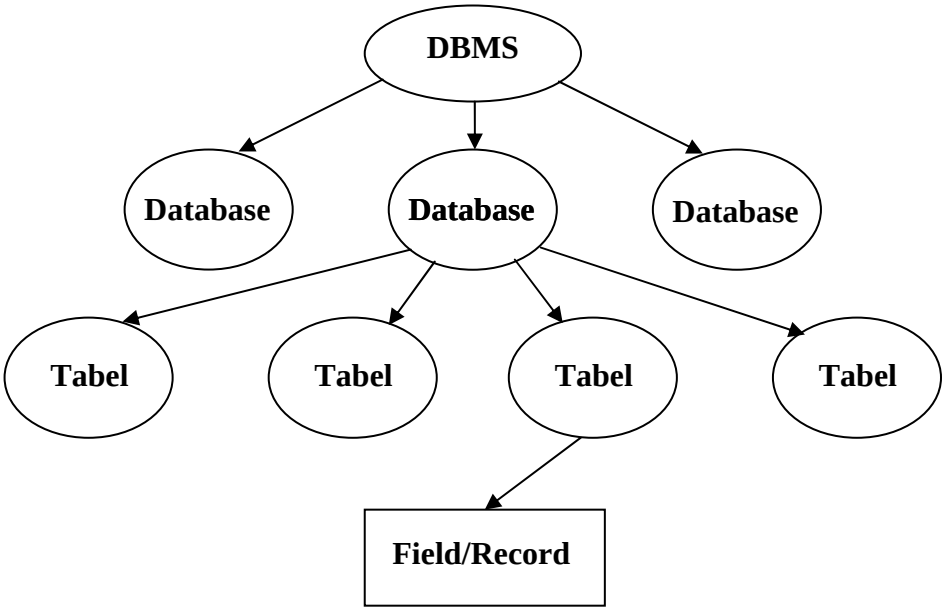
Model database hirarki dan jaringan merupakan model database yang tidak banyak lagi dipakai saat ini, karena adanya berbagai kelemahan dan hanya cocok untuk struktur hirarki dan jaringan saja. Kedua model database tersbut tidak mengakomodir untuk berbagai macam jenis persoalan dalam suatu sistem database. Yang paling banyak dipakai saat ini adalah model database relasi, karena mampu mengakomodir berbagai permasalahan dalam sistem database.

2.7.3. Model Database Relasional (*Relational Database Model*)

Model database relasional merupakan model database yang paling banyak digunakan saat ini, karena paling sederhana dan mudah digunakan serta yang paling penting adalah kemampuannya dalam mengakomodasi berbagai kebutuhan pengelolaan database. Sebuah database dalam model ini disusun dalam bentuk tabel dua dimensi yang terdiri dari baris (*record*) dan kolom (*field*), pertemuan antara baris dengan kolom disebut item data (*data value*), table-tabel yang ada di hubungkan (*relationship*) sedemikian rupa menggunakan *field-field* kunci (*Key field*) sehingga dapat meminimalkan duplikasi data.

2.7.4. Hierarki Database Relasional

Dalam suatu sistem database relasional, terdapat tingkatan-tingkatan data yang dapat digambarkan sebagai berikut:



Gambar 2.20 Hierarki Database Relasional

Deskripsi dari tingkatan-tingkatan tersebut dapat dijelaskan sebagai berikut:

1. DBMS (*Database Management System*) adalah paket program (*Software*) yang dibuat agar memudahkan dan mengefisienkan pemasukan, pengeditan, penghapusan dan pengambilan informasi terhadap database.
2. Database adalah suatu koleksi data komputer yang terintegrasi, diorganisasikan dan disimpan dalam suatu cara yang memudahkan pengambilan kembali.
3. Tabel merupakan sesuatu yang dapat diidentifikasi dari suatu sistem database, bisa berupa objek, orang, tempat, kejadian atau konsep yang informasinya akan disimpan di database.
4. *Field*/*Record* adalah kumpulan data *value* dari *attribute* yang berkaitan sehingga dapat menjelaskan sebuah tabel/ *entity* secara lengkap

2.7.5. Bahasa Database

Bahasa adalah sarana yang penting untuk berkomunikasi dan mempelajari apapun, begitu halnya dengan database, untuk membuat atau memanipulasi database diperlukan suatu bahasa tertentu. Bahasa database dibagi menjadi dua bagian, yaitu:

1. DDL (*Data Definition Language*)

   DDL (*Data Definition Language*) adalah suatu bahasa database yang menjelaskan data dan struktur data dalam suatu database. DDL merupakan bagian dari SQL (*Structured Query Language*: adalah suatu standar bahasa komputer ANSI untuk mengakses dan memanipulasi database) yang mengijinkan pengguna untuk menetapkan type-type data, struktur dan constraints dari data yang disimpan dalam database. DDL juga dapat melakukan operasi pembuatan, manipulasi dan penghapusan entitas, termasuk tabel, *view, index, user accounts* dan *roles*.

   Fungsi dan penjelasan dari DDL adalah sebagai berikut:

   a. digunakan untuk menjelaskan skema database sebagai suatu kumpulan definisi yang diekspresikan dalam DDL.
   b. Pernyataan (*Statement*) DDL disusun, memberikan hasil dalam suatu kumpulan tabel dan disimpan dalam suatu file khusus yang dinamakan kamus data dan direktori data.
   c. Direktori data terdiri dari Metadata.
   d. Struktur penyimpanan dan metode akses digunakan oleh sistem database yang dijelaskan dalam suatu kumpulan definisi dalam tipe

khusus dari DDL yang dinamakan penyimpanan data dan *definition language*.

2. DML (*Data Manipulation Language*)

   Bahasa yang digunakan oleh aplikasi dan atau secara langsung oleh pengguna untuk memanipulasi data dalam suatu database. Bahasa ini digunakan untuk mengambil, menyisipkan dan memodifikasi data informasi yang terdapat dalam database. Tujuan dari DML adalah untuk memberikan efisiensi dalam interaksi pengguna dengan komputer. Perintah-perintah dalam DML digunakan oleh semua pengguna database sebelum operasi rutin yang dilakukan oleh database. Dalam SQL, terdapat perintah-perintah DML seperti *Insert, Update* dan *Delete*.

   Fungsi-fungsi DML:

   a. menerima informasi dari database.
   b. Menyisipkan informasi baru kedalam database.
   c. Melakukan penghapusan infromasi dalam database.
   d. Memodifikasi informasi dalam database.

   Terdapat dua tipe DML, yaitu:

   a. Prosedural: pengguna dibutuhkan untuk menspesifikasi data apa yang dibutuhkan dan bagaimana untuk mendapatkannya.
   b. Non Prosedural: pengguna dibutuhkan untuk menspesifikasi hanya data apa yang dibutuhkan, bukan bagaimana untuk mendapatkannya. Sebagai contoh adalah SQL.

2.7.6. SQL

SQL (*Structured Query Language*) merupakan bahasa *query* yang paling banyak digunakan oleh DBMS dan *Development Tools* (Visual Basic, Delphi, Java, dll) dalam menyediakan media bagi penggunanya untuk berinteraksi dengan basis data. Bahasa ini dibangun dengan dasar aljabar relasional dan sedikit kalkulus relasional. Pada tahun 1992 telah ditetapkan standar awal dari bahasa SQL, terutama menyangkut sintaksnya (tata bahasanya), yang kemudian dikenal dengan SQL-92. dalam perjalannya, karena adanya kebutuhan-kebutuhan baru dalam pengelolaan dan pengolahan data, masing-masing pembuat DBMS melakukan pengembangan sendiri-sendiri, yang pada akhirnya melahirkan varian-varian SQL. Karena itu kendati secara prinsip sama, tapi terkadang kita harus menuliskan ekspresi SQL sedikit berbeda antara DBMS satu dengan yang lain.

2.7.6.1. Struktur Dasar Query

Sebuah ekspresi SQL dasar sebenarnya hanya terdiri dari 3 klausa,yaitu *select*, *from* dan *where*, dimana:

1. Klausa ***Select*** digunakan untuk menetapkan daftar atribut (Field) yang diinginkan sebagai hasil query. Klausa ini mengacu pada operasi projeksi dalam bahasa query normal.
2. Klausa ***From*** digunakan untuk menetapkan relasi atau tabel (atau gabungan tabel) yang akan ditelusuri selama *query* data dilakukan.

3. Klausa ***Where*** yang sifatnya opsional, digunakan sebagai predikat (kriteria) yang harus dipenuhi dalam memperoleh hasil *query*, Klausa ini mengacu pada operasi select dalam bahasa *query* normal.

Sintaks (format penulisan) dari ekspresi SQL dasar dengan tiga klausa tersebut adalah:

Select A1 [, A2, ..., An]

From t1 [, t2, ..., tm]

{ ***Where*** P }

2.7.7. MySQL

Merupakan software DBMS (*Database Management Systems*) yang *multithread* dan *multi-user*. MySQL AB membuat MySQL tersedia sebagai software gratis dibawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius.

MySQL merupakan software yang *open source* artinya memungkinkan untuk semua orang untuk menggunakan dan memodifikasi software. Setiap orang dapat men-download MySQL dari internet dan menggunakannya tanpa membayar apapun. Banyak para pengguna database menggunakan database MySQL

dikarenakan database ini selain *open source*, MySQL juga sangat cepat, reliable, dan mudah untuk digunakan, selain itu MySQL telah banyak menangani pembuatan software besar sehingga keampuhannya tidak dapat diragukan lagi.

2.8. RF Smart Card

RFID (*Radio Frequency Identification)* adalah suatu teknologi yang secara konseptual mirip dengan *Barcode* yang banyak dijumpai di banyak barang dagangan, atau *magnetic strips* yang banyak digunakan dilingkungan kartu kredit. Bedanya, *RFID* terdiri atas sebuah *chip* eletronik yang bisa dijejali dengan aneka macam data/informasi sesuai kebutuhan. Data itu misalnya saja berupa tipe barang, tanggal pembuatan, tujuan pengiriman barang, dan data-data lainnya. Perbedaan lainnya bilamana *barcode* memerlukan proses pemindaian sebelum datanya terbaca, *RFID* menggunakan teknologi radio pada frekuensi tertentu, dan frekuensi ini kemudian dibaca sebuah alat yang dinamakan *RFID Reader*. *RFID* terdiri dari 2 bagian yaitu *Transponder* dan *Reader*. *Transponder* bisa berupa *Card* (kartu), *Tag* (kancing), atau *Wristband* (jam tangan). Pada kartu ini hanya mempunyai tingkat frekuensi sebesar 125 KHz. Sedangkan *Reader* berfungsi untuk membaca nomor data ID yang dikirimkan oleh *Transponder*. Efisiensi tinggi yang diberikan oleh sistem *RFID* mendatangkan masalah keamanan dan privasi. Sistem *RF Smart Card* rentan terhadap serangan secara fisik, mudah dipalsukan, disadap, saluran komunikasinya mudah dibanjiri dengan *noise* sehingga dapat mematikan peralatan komunikasi. Serangan ini disebut serangan *Denial of Service* dengan tujuan menutup saluran komunikasi. Berdasarkan

lubang keamanan, keamanan dapat diklasifikasikan menjadi empat, yaitu : keamanan bersifat fisik, keamanan yang berhubungan dengan orang, keamanan dari data dan media, teknik komunikasi serta keamanan dalam operasi.

1. Keamanan yang bersifat fisik

   Keamanan yang bersifat fisik dilakukan penyerangan terhadap RFID yaitu antara lain berupa pemindahan material kartu maupun menukar kartu sehingga data yang ada di dalam kartu tidak sesuai dengan keadaan yang sebenarnya. Serangan Denial of Service juga dikategorikan dalam kelas ini yaitu dengan mengacaukan siaran (broadscast), menutup pesan yang akan masuk pada reader dengan cara membanjiri saluran komunikasi dengan noise yang banyak.

2. Keamanan yang berhubungan dengan personal

   Keamanan yang berhubungan dengan manusia pada sistem RFID adalah dengan adanya keteledoran perilaku manusia yang mempunyai akses ke RFID, misalkan memberikan password yang tidak berhak atau membuang kartu RFID yang masih berisi data tentang database, sehingga penyadap yang menemukan dapat membaca dan melakukan penyerangan yang sama pada kartu yang masih aktif sehingga dapat mengganggu privasi dari orang yang berhak.

3. Keamanan dari data media serta teknik komunikasi

   Keamanan RFID yang termasuk kelas ini adalah kelemahan dalam software yang digunakan untuk mengelola data pada kartu sehingga seseorang yang

tidak mempunyai hak akses dapat membuka isi dari kartu karena password yang ada di kartu dapat dibuka.

4. Keamanan dalam operasi

   Keamanan RFID yang termasuk dalam kelas ini adalah prosedur yang digunakan untuk mengatur dan mengelola tentang sistem keamanan dan juga termasuk prosedur setelah terjadi serangan.

ID Smart card sering disebut juga sebagai chip card atau integrated circuit (IC) card . Definisi chip card sendiri yaitu kategori umum yang mencakup smart card dan memory card. ID Smart card adalah plastic card yang mengandung memory chip dan microprocessor. Kartu ini bisa menambah, menghapus, mengubah informasi yang terkandung. Keunggulannya adalah ID smart card tidak perlu mengakses database di server karena sudah ada sebagian terkandung di kartu. Sedangkan memory card dipasangi memory silicon tanpa microprocessor. Fungsi dasar suatu ID smart card adalah untuk mengidentifikasi card holder ke sistem komputer. Card holder disini adalah pemilik kartu tersebut, identifikasi ini menyangkut otentifikasi organisasi yang membuat kartu tersebut dan card holder dan hak aksesnya. Beberapa hal yang harus dipertimbangkan dalam metode identifikasi kartu adalah :

1. Mengkonfirmasi identitas card holder (pemegang kartu) sebelum mengakses data.
2. Memberikan data untuk konfirmasi ke alat eksternal, sistem atau perorangan.

3. Menyediakan data ke sistem tanpa pengecekan orang yang menggunakan.

2.9. Borland C++ Builder

C++ Builder adalah merupakan suatu alat pengembang aplikasi (*development tools*) berbasis Microsoft Windows yang menerapkan konsep visualisasi. Dengan adanya dukungan visualisasi ini, C++ Builder menjadi mudah digunakan untuk membuat aplikasi-aplikasi secara tepat. Dengan C++ Builder, kita dapat melakukan desain, *testing*, *debugging*, maupun *deployment* aplikasi secara mudah.

Bahasa dasar yang digunakan dalam pemrograman C++ Builder adalah bahasa C++ yang telah terkenal dengan keampuhannya dalam pembuatan program. Tingkat keteraturan dan kemudahan dari bahasa C++ relatif lebih rendah jika dibandingkan dengan bahasa Objek *Pascal*, namun dengan mempelajari konsep-konsep yang terdapat di dalamnya, tentu C++ akan menjadi bahasa yang mudah dipahami dan dipelajari.