

BAB 2

LANDASAN TEORI

2.1 Teori-Teori Basis Data

2.1.1 Data

Menurut Whitten, Bentley, dan Dittman (2004, p23), data adalah fakta mentah mengenai orang, tempat, kejadian, dan hal-hal penting dalam organisasi. Tiap fakta, dengan sendirinya, secara relatif tidak ada artinya.

Menurut Turban (2003, p17), data adalah suatu fakta atau deksripsi dasar dari sesuatu, kejadian, aktivitas, dan transaksi yang diperoleh, disimpan, direkam, diklasifikasikan, tetapi belum memberikan manfaat khusus bagi penggunanya.

Menurut Stephens dan Plew (2004), Data adalah segala informasi yang berhubungan dengan organisasi yang harus disimpan untuk berbagai keperluan sesuai dengan persyaratan organisasi.

Jadi dapat disimpulkan bahwa data adalah suatu fakta dari suatu, kejadian, aktivitas, dan transaksi yang diciptakan, disimpan, diklasifikasikan oleh sistem informasi tetapi belum memberikan manfaat khusus bagi penggunanya.

2.1.2 Definisi *Database*

Menurut Atzeni, Ceri, Paraboschi, dan Torlone *Database* (2005, p2) adalah kumpulan data yang digunakan untuk menggambarkan informasi penting menjadi sistem informasi.

Menurut Kroenke *Database* (2007, p11) adalah menggambarkan kumpulan dari table-table yang terintegrasi. *Table-table* yang terintegrasi adalah *table* yang menyimpan data dan hubungan-hubungan antar data.

Menurut Hoffer, Prescott, dan Mc Fadden *database* adalah kumpulan-kumpulan data yang terorganisir yang terhubung secara logis. *Database* dapat berbeda-beda bentuk dan kompleksitas.

Menurut Connolly dan Begg (2010, p65), pengertian *database* adalah kumpulan data yang terhubung secara logis yang dipakai bersama dan deskripsi dari data ini dirancang untuk memenuhi kebutuhan informasi sebuah organisasi.

Menurut Stephens dan Plew (2004), *Database* adalah mekanisme yang digunakan untuk menyimpan informasi atau data. Dengan *database*, user dapat menyimpan data pada tempat penyimpanan yang terorganisasi.

Menurut O'Brien (2003, p145) *database* adalah sebuah kumpulan yang terintegrasi dari elemen data yang terhubung secara logikal.

Elemen data mendeskripsikan entiti-entiti dan hubungan antara entiti-entiti.

Jadi *database* adalah suatu sistem penyimpanan data yang tersusun atas sekumpulan data yang secara logika saling terkait yang dirancang untuk memenuhi kebutuhan informasi perusahaan. Model *database* relasional adalah sistem yang banyak digunakan karena struktur logikalnya yang sederhana. Pada model relasional seluruh data disusun secara logikal dalam relasi-relasi atau tabel. Setiap relasi terdiri dari baris, dan kolom dari relasi yang diberi nama tertentu disebut atribut. Sedangkan baris dari relasi disebut *tuple* dan setiap *tuple* (baris) memiliki satu nilai untuk setiap atribut.

Database yang tabel-tabelnya saling terhubung dikatakan memiliki relasi. Karena tidak ada relasi yang memiliki dua *tuple* yang sama, maka setiap baris dapat didefinisikan secara unik dengan menggunakan *primary key*. Munculnya sebuah atribut dalam beberapa relasi dapat mempresentasikan hubungan antara *tuple* dari relasi tersebut (2010).

Pemakai *database* dapat berupa orang atau program aplikasi. Orang biasanya menggunakan *database* dari terminal dan mengambil data dan informasi dengan menggunakan *query language*. *Query* adalah permintaan informasi dari *database*, dan *query language* adalah bahasa khusus yang *user-friendly* yang memungkinkan komputer menjawab *query*.

Pendekatan basis data adalah memisahkan struktur data dari program aplikasi dan menyimpannya dalam *database*. *Database*

merupakan sistem penyimpanan *record* terkomputerisasi yang bertujuan untuk pemeliharaan informasi dan tersedia pada saat dibutuhkan. Dalam menganalisa kebutuhan informasi suatu organisasi, kita berusaha menentukan *entity*, *attribute*, dan *relation*.

Teknologi basis data memperbolehkan sekumpulan data dengan berbagai tipe (teks, angka, gambar, suara, dan lain-lain) disimpan dalam komputer dan digunakan secara efisien tanpa adanya duplikasi oleh aplikasi yang berhubungan.

2.1.3 Database Management System (DBMS)

2.1.3.1 Definisi DBMS

Menurut Connolly dan Begg (2010 , p66), pengertian DBMS adalah sebuah sistem piranti lunak yang memungkinkan user untuk mendefinisikan, membuat, menjaga, dan mengontrol akses ke dalam basis data.

2.1.3.2 Tujuan DBMS

Tujuan utama pengolahan data dalam basis data adalah agar dapat memperoleh data yang dicari dengan mudah dan cepat. Pemanfaatan basis data dilakukan untuk memenuhi sejumlah tujuan seperti berikut ini :

1. Kecepatan dan kemudahan (*speed*)
2. Efisiensi ruang penyimpanan (*space*)
3. Keakuratan (*accuracy*)
4. Ketersediaan (*availability*)
5. Kelengkapan (*completeness*)
6. Keamanan (*security*)
7. Kebersamaan pemakai (*sharebility*)

2.1.3.3 Komponen – Komponen DBMS

Menurut Connolly dan Begg (2010, p68), *Database Management System* (DBMS) memiliki 5 komponen penting, yaitu:

1. *Hardware* (Perangkat Keras)

Dalam menjalankan aplikasi dan DBMS diperlukan perangkat keras. Perangkat keras dapat berupa *single personal computer*, *single mainframe*, sampai jaringan komputer. Perangkat keras yang digunakan bergantung pada persyaratan dari organisasi dan DBMS yang digunakan.

2. *Software* (Perangkat Lunak)

Komponen perangkat lunak meliputi DBMS *software* dan program aplikasi beserta Sistem Operasi, termasuk perangkat lunak tentang jaringan bila DBMS digunakan dalam jaringan seperti LAN (*Local Area Network*).

3. Data

Data merupakan komponen terpenting dari DBMS dan juga merupakan komponen penghubung antara komponen mesin (*Hardware* dan *Software*) dan komponen *human* (*Procedures* dan *People*).

4. Prosedur

Prosedur merupakan panduan dan instruksi dalam membuat desain dan menggunakan basis data. Penggunaan dari sistem dan staf dalam mengelola basis data membutuhkan prosedur dalam menjalankan sistem dan mengelola basis data itu sendiri. Prosedur di dalam basis data dapat berupa: *login* di dalam basis data, penggunaan sebagian fasilitas DBMS, cara menjalankan dan memberhentikan DBMS, membuat salinan *backup database*, memeriksa *hardware* dan *software* yang sedang berjalan, mengubah struktur basis data, meningkatkan kinerja atau membuat arsip data pada media penyimpanan sekunder.

5. Manusia

Komponen terakhir yaitu manusia sendiri yang terlibat dalam sistem tersebut. Komponen ini meliputi *data and database administrator*, *database designers*, *application developers*, dan *end-users*.

2.1.3.4 Keuntungan dan Kerugian DBMS

Menurut Connolly dan Begg (2010, p77), keuntungan DBMS adalah sebagai berikut:

1. Mengontrol redudansi data
2. Mendapat informasi yang lebih dari jumlah data yang sama
3. Peningkatan integritas data
4. Peningkatan produktifitas
5. Peningkatan keamanan serta layanan backup dan recovery

Menurut Connolly dan Begg (2010, p80), kerugian DBMS adalah sebagai berikut :

1. Kompleksitas
2. Ukuran
3. Biaya dari DBMS
4. Biaya tambahan perangkat keras
5. Biaya proses konversi
6. Performa
7. Pengaruh kegagalan yang lebih tinggi

2.1.4 Database Language

2.1.4.1 Data Definition Language (DDL)

Menurut Connolly dan Begg (2010, p92), pengertian *Data Definition Language* adalah suatu bahasa yang memperbolehkan *Database Administrator* (DBA) atau pengguna untuk mendeskripsikan dan memberi nama suatu entitas, atribut, dan relasi data yang

dibutuhkan untuk aplikasi, bersama dengan integritas data yang diasosiasikan dan batasan (*constraint*) keamanan data.

Perintah dalam bahasa tersebut secara umum antara lain:

1. CREATE, digunakan untuk membuat suatu objek basis data yang baru.
2. ALTER, digunakan untuk mengubah atribut-atribut dari objek basis data yang sudah terdapat pada basis data.
3. DROP, digunakan untuk menghapus objek tertentu.

2.1.4.2 Data Manipulation Language (DML)

Menurut Connolly dan Begg (2010, p92), pengertian *Data Manipulation Language* adalah suatu bahasa yang menyediakan seperangkat operasi untuk mendukung manipulasi data yang berada pada basis data.

Pengoperasian data yang akan dimanipulasi biasanya meliputi :

1. Penambahan data baru ke dalam basis data.
2. Modifikasi data yang disimpan ke dalam basis data.
3. Pengembalian data yang terdapat di dalam basis data.
4. Penghapusan data dari basis data.

DML dibagi menjadi 2 jenis yaitu *Procedural* dan *Non-procedural*. Menurut Connolly dan Begg (2010, p), pengertian *Procedural DML* adalah suatu bahasa yang memperbolehkan pengguna untuk mendeskripsikan ke sistem data apa yang dibutuhkan

dan bagaimana mendapatkan data tersebut secara tepat, sedangkan *Non-procedural DML* adalah sebuah bahasa yang mengizinkan pengguna untuk menentukan data apa yang dibutuhkan tanpa memperhatikan bagaimana data diperoleh.

Perintah-perintah dalam DML antara lain :

1. SELECT, digunakan untuk melakukan *query*.
2. INSERT, digunakan untuk memasukan data ke tabel.
3. UPDATE, digunakan untuk memperbaharui data pada tabel.
4. DELETE, digunakan untuk menghapus data dari tabel.
5. FROM, digunakan untuk menentukan tabel yang ingin digunakan selama proses pengekseskuan *query*.
6. WHERE, digunakan untuk melakukan filtrasi data pada table yang dilakukan *query* berdasarkan kondisi tertentu.

2.1.4.3 Fungsi Agregat dan Kontrol Akses

Fungsi-fungsi agregat yang dimiliki SQL antara lain :

1. COUNT, digunakan untuk menghitung jumlah *record* / *row* yang berhasil diambil dalam suatu proses *query*.
2. SUM, digunakan untuk menghitung jumlah nilai keseluruhan dalam suatu kolom.
3. AVG, digunakan untuk menghitung nilai rata-rata dalam suatu

kolom.

4. *MIN*, digunakan untuk menampilkan nilai terkecil dalam suatu

kolom.

5. *MAX*, digunakan untuk menampilkan nilai terbesar dalam suatu

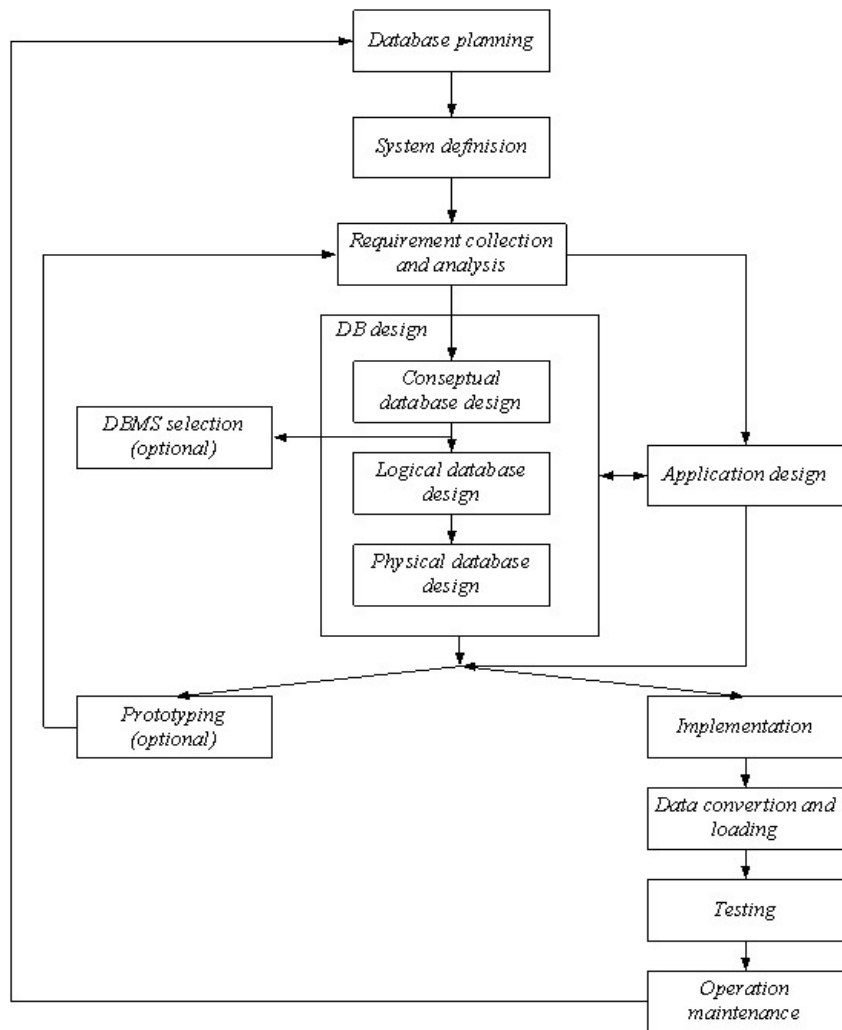
kolom.

Kontrol akses digunakan untuk memberi/mencabut hak akses dari/untuk pengguna basis data. Perintah dalam kontrol akses :

1. *GRANT*, digunakan untuk memberikan hak akses kepada pengguna basis data.
2. *REVOKE*, digunakan untuk mencabut hak akses kepada pengguna basis data.

2.1.5 Database Lifecycle

Menurut Connolly dan Begg (2010, p313), sebuah sistem *database* merupakan komponen dasar sistem informasi organisasi yang lebih besar sehingga siklus hidup aplikasi database berhubungan dengan siklus hidup system informasi. Tahapan-tahapan siklus hidup aplikasi adalah sebagaimana terlihat pada gambar berikut :



Gambar 2.1 Database Lifecycle

(Sumber : Connolly, 2010, p314)

2.1.5.1 Definisi Sistem

Menurut Connolly dan Begg (2010, p316), sistem adalah menggambarkan lingkup dan batasan-batasan dari aplikasi basis data dan *user view* yang utama. Sebelum mencoba merancang suatu aplikasi basis data diperlukan untuk mengenali batasan sistem dan bagaimana antarmuka dengan bagian sistem informasi lainnya dalam

organisasi. Hal penting yang harus diperhatikan adalah batasan pemakai dan aplikasi mendatang. Mengidentifikasi *user view* sangat penting dalam mengembangkan aplikasi basis data agar dapat memastikan tidak ada pemakai utama yang terlupakan ketika mengembangkan keperluan untuk aplikasi baru.

Menurut O'Brien (2003, p8) sistem adalah kumpulan elemen yang saling terhubung atau berinteraksi membentuk suatu kesatuan atau sekumpulan komponen yang saling terhubung dan bekerja sama untuk mencapai sasaran dengan menerima *input* dan menghasilkan *output* dalam sebuah proses transformasi yang terorganisir.

Dari pendapat-pendapat di atas dapat disimpulkan sistem adalah kumpulan unsur-unsur yang berhubungan untuk melaksanakan kegiatan-kegiatan perusahaan dalam mencapai suatu tujuan tertentu.

2.1.5.2 Pengumpulan dan Analisis Kebutuhan

Menurut Connolly dan Begg (2010, p316), pengumpulan dan analisis kebutuhan adalah proses dari analisis dan pengumpulan informasi tentang bagian organisasi yang didukung oleh sistem aplikasi basis data dan menggunakan informasi ini untuk mengenali kebutuhan-kebutuhan untuk sistem baru. Pengumpulan dan analisis kebutuhan adalah tahapan persiapan merancang basis data. Jumlah data yang dikumpulkan tergantung pada masalah alamiah dan kebijakan suatu perusahaan. Lebih banyak mempelajari lebih cepat membimbing ke analisis permasalahan. Lebih sedikit berpikir dapat mengakibatkan membuang waktu dan uang secara sia-sia karena

bekerja pada solusi yang salah ke masalah yang salah. Beberapa teknik atau cara untuk mendapatkan informasi adalah dengan teknik *Fact Findng. Fact Finding* adalah teknik yang digunakan untuk mengidentifikasi kebutuhan.

2.1.5.3 Metodologi Perancangan Basis Data

Menurut Connolly dan Begg (2010, p466), metodologi perancangan basis data adalah suatu pendekatan terstruktur yang menggunakan prosedur, teknik, alat-alat, dan bantuan dokumentasi untuk mendukung dan memfasilitasi proses perancangan. Menurut Connolly dan Begg (2010, p467) proses perancangan terdiri dari tiga bagian, yaitu:

1. Perancangan Basis Data Konseptual

Perancangan basis data konseptual adalah proses membangun suatu model informasi yang digunakan suatu perusahaan, yang berdiri sendiri terhadap semua pertimbangan fisik.

2. Perancangan Basis Data Logikal

Perancangan basis data logikal adalah proses membangun model informasi yang digunakan dalam suatu perusahaan berdasarkan pada spesifik data model, tetapi berdiri sendiri terhadap semua fakta-fakta DBMS dan pertimbangan fisik lainnya.

3. Perancangan Basis Data Fisikal

Perancangan basis data fisikal adalah proses menghasilkan satu deskripsi mengenai implementasi basis data pada media

penyimpanan sekunder; dia menggambarkan dasar relasi, *file* organisasi, dan *indeks-indeks* yang digunakan untuk mencapai efisiensi akses terhadap data, dan semua integritas *constraint* dan pengukuran keamanan

2.1.5.4 Seleksi DBMS

Menurut Connolly dan Begg (2010, p325), pengertian seleksi DBMS adalah menyeleksi DBMS yang tepat untuk mendukung aplikasi basis data. Seleksi DBMS dilakukan antara tahapan perancangan database logikal dan perancangan database fisikal. Tujuannya untuk kecukupan sekarang dan kebutuhan masa mendatang pada perusahaan, membuat keseimbangan biaya termasuk pembelian produk DBMS, piranti lunak untuk mendukung aplikasi basis data, biaya yang berhubungan dengan perubahan dan pelatihan pegawai.

2.1.5.5 Perancangan Aplikasi

Menurut Connolly dan Begg (2010, p329), pengertian perancangan aplikasi adalah merancang antarmuka pemakai dan program aplikasi, yang akan memproses basis data. Perancangan basis data dan aplikasi merupakan aktivitas yang dilakukan secara bersamaan pada *database application life cycle*.

2.1.5.6 Prototyping

Menurut Connolly dan Begg (2010, p333), pengertian *prototyping* adalah membuat model kerja dari aplikasi basis data. Tujuannya adalah untuk memungkinkan pemakai menggunakan *prototype* untuk

mengidentifikasi fitur-fitur sistem berjalan dengan baik atau tidak, dan bila memungkinkan untuk menyarankan peningkatan atau bahkan penambahan fitur-fitur baru ke dalam sistem database.

Ada dua macam strategi *prototyping* yang digunakan sekarang :

1. *Prototyping* Kebutuhan (*Requirement Prototyping*)

Menggunakan suatu *prototype* untuk menetapkan kebutuhan dari tujuan aplikasi basis data dan ketika kebutuhan sudah terpenuhi, *prototype* tidak digunakan lagi atau dibuang.

2. *Prototyping* Evolusioner (*Evolutionary Prototyping*)

Prototype Evolusioner digunakan dengan tujuan yang sama. Perbedaan yang penting adalah bahwa *prototype* tidak dibuang tetapi dengan mengembangkan lebih lanjut menjadi aplikasi basis data yang dikerjakan.

2.1.5.7 Implementasi

Menurut Connolly dan Begg (2010, p333), pengertian implementasi adalah realisasi fisik suatu basis data dan perancangan aplikasi. Implementasi basis data dapat dicapai menggunakan *Data Definition Language* (DDL) dari DBMS yang dipilih atau *Graphical User Interface* (GUI). Pernyataan DDL digunakan untuk menciptakan struktur-struktur basis data dan *file-file* basis data yang kosong. Semua spesifikasi *user view* juga diimplementasikan pada tahap ini.

2.1.5.8 Data Conversion And Loading

Menurut Connolly dan Begg (2010, p334), pengertian *data conversion and loading* adalah mentransfer semua data yang telah ada ke dalam basis data yang baru dan mengkonversi semua aplikasi yang ada untuk dijalankan pada basis data yang baru. Tahap ini hanya dibutuhkan ketika sistem basis data yang baru menggantikan sistem basis data yang lama. Pada masa sekarang, umumnya DBMS memiliki kegunaan untuk memasukkan *file* ke dalam basis data baru tujuannya adalah untuk memungkinkan pengembang untuk mengkonversi dan menggunakan aplikasi program lama untuk digunakan oleh sistem baru.

2.1.5.9 Pengujian

Menurut Connolly dan Begg (2010, p334), pengertian pengujian adalah proses menjalankan program aplikasi dengan maksud untuk mencari kesalahan. Sebelum digunakan, aplikasi basis data yang baru dikembangkan harus diuji secara menyeluruh. Untuk mencapainya harus hati-hati dalam menggunakan perencanaan strategi uji dan menggunakan data asli untuk semua proses pengujian.

Pengguna-pengguna suatu sistem yang baru seharusnya dilibatkan dalam proses pengujian. Situasi yang ideal untuk pengujian suatu sistem adalah dengan menguji basis data pada sistem *hardware* yang berbeda, tetapi sering kali ini tidak tersedia. Jika data sesungguhnya digunakan, sangat penting sekali untuk memiliki *backup* untuk menangkap kesalahan yang terjadi Setelah pengujian selesai , *system* aplikasi siap digunakan dan diserahkan ke pemakai.

2.1.5.10 Operasional dan Pemeliharaan

Menurut Connolly dan Begg (2010, p335), pengertian operasional dan pemeliharaan adalah proses memonitor dan memelihara sistem yang telah di-*install*.

2.2 Pengertian Normalisasi

Normalisasi merupakan sebuah teknik dalam perancangan logikal sebuah basis data, teknik pengelompokkan atribut dari suatu relasi sehingga membentuk struktur relasi yang baik (tanpa redundansi). Menurut Connolly dan Begg (2010, p416), pengertian normalisasi adalah teknik untuk menghasilkan sejumlah relasi *table* dengan *properties* yang diinginkan, sesuai dengan kebutuhan data dari perusahaan.

Dengan kata lain normalisasi merupakan proses mengubah suatu relasi yang memiliki masalah tertentu ke dalam dua relasi atau lebih yang tidak memiliki masalah tersebut. Masalah yang dimaksud itu sering disebut dengan istilah anomali.

2.2.1 Data Redundancy and Update Anomaly

Anomali adalah efek samping yang tidak diharapkan (misalnya menyebabkan *inconsistency* (tidak konsisten) data atau membuat suatu data menjadi hilang saat data lain dihapus) yang muncul dalam suatu proses perancangan basis data. Suatu tujuan desain *database relational* yang utama adalah menggolongkan atribut ke dalam hubungan-hubungan untuk memperkecil data *redundancy* dan dengan

demikian mengurangi tempat penyimpanan file yang diperlukan oleh hubungan-hubungan dasar yang diimplementasikan. Hubungan-hubungan yang memiliki data redundan mungkin memiliki masalah yang disebut *update anomalies*, yang diklasifikasikan sebagai *insertion, deletion, atau modification anomalies*

2.2.2 *Functional Dependency*

Functional Dependency (ketergantungan fungsional) menguraikan hubungan antara atribut-atribut dalam sebuah relasi. Sebagai contoh, jika A dan B adalah relasi R, B adalah secara fungsional bergantung kepada A ($A \rightarrow B$), jika setiap nilai dari A diasosiasikan dengan tepat satu nilai dari B. (A dan B masing-masing boleh dari satu atau lebih atribut).

2.2.3 **Bentuk Normal**

Normalisasi sering dieksekusi sebagai langkah-langkah yang berangkai/berseri. Bentuk normal adalah suatu aturan yang dikenakan pada relasi-relasi dalam basis data dan harus dipenuhi oleh relasi-relasi tersebut pada tingkatan normalisasi. Suatu relasi dikatakan berada dalam bentuk normal tertentu jika memenuhi kondisi-kondisi tertentu.

Beberapa tingkatan yang biasa digunakan pada normalisasi adalah:

1. UNF

Sebelum membahas bentuk normal yang pertama, kita mendefinisikan normal *form* awal yaitu *Unnormalized Form* (UNF). UNF adalah sebuah tabel yang berisi satu atau lebih

kelompok data yang berulang.

2. Bentuk normal pertama (1NF)

Bentuk normal pertama adalah hubungan dimana persimpangan dari setiap baris dan kolom berisi satu dan hanya satu nilai. Atau dengan kata lain, pada 1NF kita menghilangkan repetisi dan data yang merupakan hasil kalkulasi.

3. Bentuk normal kedua (2NF)

Bentuk normal kedua didefinisikan berdasarkan ketergantungan fungsional penuh (*Full Functional Dependency*). *Full Functional Dependency* menandai bahwa jika A dan B adalah atribut dari sebuah relasi, B adalah penuh secara fungsional tergantung pada A jika B adalah secara fungsional tergantung pada A, tetapi tidak pada semua subset dari A.

Sedangkan 2NF adalah sebuah relasi antara bentuk normal pertama, dan setiap atribut bukan *primary key* adalah penuh secara fungsional bergantung pada *primary key*. Atau dengan kata lain, pada 2NF kita menghilangkan ketergantungan *partial*.

4. Bentuk normal ketiga (3NF)

Bentuk normal ketiga didefinisikan berdasarkan ketergantungan transitif (*Transitive Dependency*).

Transitive Dependency adalah sebuah kondisi dimana A, B, dan C adalah atribut-atribut dari relasi seperti jika $A \rightarrow B$ dan $B \rightarrow C$, maka C secara transitif bergantung pada A melalui B. (Dengan

ketentuan bahwa A tidak secara fungsional bergantung pada B atau C). Sedangkan 3NF adalah sebuah relasi antara bentuk dan bentuk kedua, dan dimana tidak ada atribut yang bukan *primary key* secara transitif bergantung pada *primary key*.

5. Bentuk normal *Boyce-Codd* (BCNF)

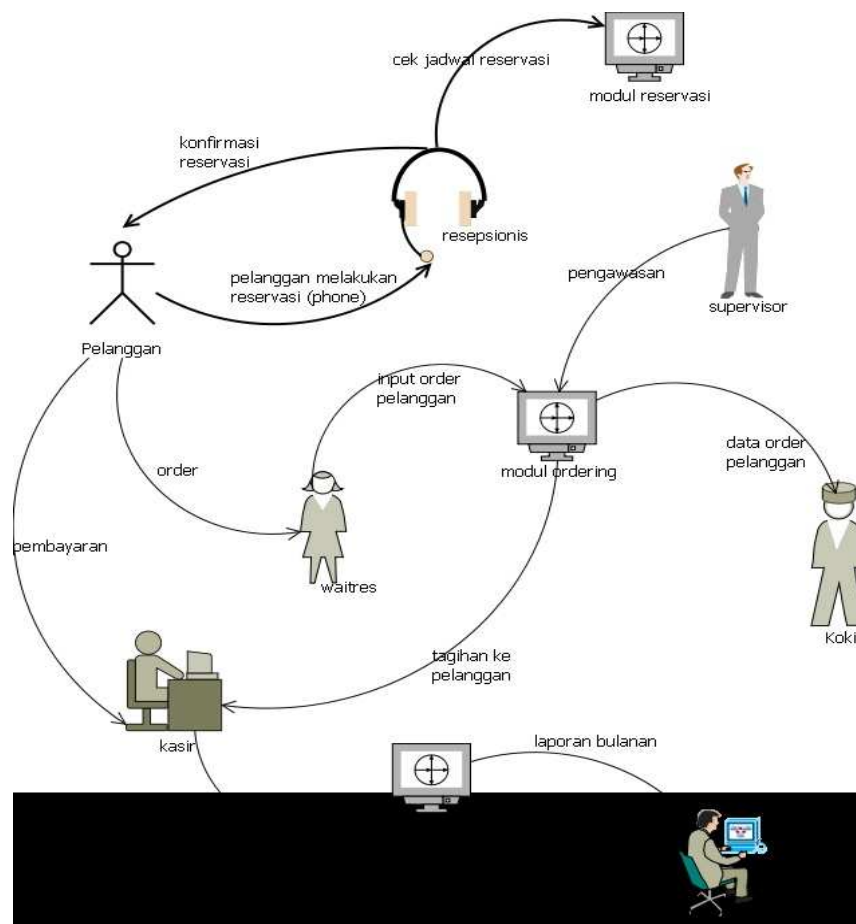
Menurut Connolly dan Begg (2010, p398) suatu relasi disebut memenuhi bentuk normal Boyce-Codd jika dan hanya jika semua penentu(determinan) adalah *candidate key*. BCNF merupakan bentuk normal sebagai perbaikan terhadap 3NF karena bentuk normal ketiga berkemungkinan masih memiliki anomali sehingga perlu dinormalisasi lebih jauh. Suatu relasi yang memenuhi BCNF selalu memenuhi 3NF, tetapi tidak untuk sebaliknya.

Bentuk normal pertama hingga ketiga merupakan bentuk normal yang umum dipakai. Artinya, bahwa pada kebanyakan relasi bila ketiga bentuk normal tersebut telah dipenuhi maka persoalan anomali tidak akan muncul lagi. Bentuk normal *Boyce-Codd* merupakan revisi terhadap bentuk normal ketiga. Bentuk normal keempat (4NF) dan kelima (5NF) hanya dipakai pada kasus-kasus khusus, yakni pada relasi yang mengandung banyak ketergantungan nilai.

2.3 Rich Picture

Rich picture adalah penggambaran sistem atau situasi dengan menggunakan gambar-gambar. Gambaran keseluruhan dari orang, objek, proses, struktur, dan masalah pada keseluruhan proses bisnis yang ada di perusahaan

Rich picture digunakan untuk menggambarkan keseluruhan proses bisnis secara jelas dengan gambar dan hubungan antar gambar tersebut dengan penjelasan singkat agar orang yang melihat dapat dengan mudah untuk mengerti dan memahami maksud dari gambar tersebut



Gambar 2.2 Contoh Rich picture

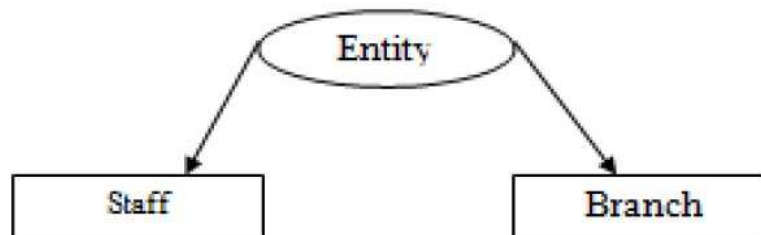
2.4 *Entity Relationship Modelling*

Menurut Connoly dan Begg (2010, p371), Salah satu aspek yang sulit dalam perancangan *database* adalah kenyataan bahwa perancang, *programmer*, dan pemakai akhir cenderung melihat data dengan cara yang berbeda. Untuk memastikan pemahaman secara alamiah dari data dan bagaimana data digunakan oleh perusahaan dibutuhkan sebuah bentuk komunikasi yang non-teknis dan bebas dari kebingungan.

2.4.1 *Entity Type*

Menurut Stephens dan Plew (2004), *Entity* adalah objek bisnis yang merepresentasikan kumpulan atau kategori data.

Menurut Connoly dan Begg (2010, p372), *Entity Type* adalah kumpulan objek-objek yang berproperti sama, dimana properti tersebut diidentifikasi memiliki keberadaan yang bebas.



Gambar 2.3 Representasi Diagram dari Tipe *entity*

(Sumber : Connoly, 2010, p374)

2.4.2 *Attribute*

Menurut Stephens dan Plew (2004), Atribut adalah informasi sub-group dalam *entity*.

Menurut Connolly dan Begg (2010, p379), atribut adalah sifat dari sebuah *entity* atau sebuah tipe *relationship*. Atribut menyimpan nilai dari setiap *entity occurrence* dan mewakili bagian utama dari data yang disimpan dalam basis data.

Macam-macam atribut:

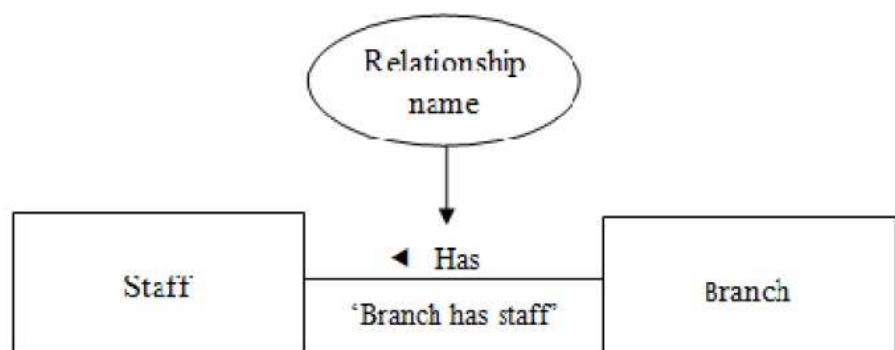
1. *Simple Attribute* yaitu atribut yang terdiri dari suatu komponen tunggal dengan keberadaan yang independent dan tidak dapat dibagi menjadi lebih kecil lagi. Dikenal juga dengan nama *atomic attribute*.
2. *Composite attribute* yaitu atribut yang terdiri dari beberapa komponen, dimana masing-masing komponen memiliki keberadaan yang independen. Misalkan atribut *address* dapat terdiri dari *street, city, postcode*.
3. *Single-Valued Attribute*, yaitu atribut yang mempunyai nilai tunggal untuk setiap kejadian. Misalnya entitas *Branch* memiliki satu nilai untuk atribut *BranchNo* pada setiap kejadian.
4. *Multi-valued attribute*, yaitu atribut yang mempunyai beberapa nilai untuk setiap kejadian. Misal 1 karyawan memiliki lebih dari 1 no.telp.

5. *Derived-attribute* yaitu atribut yang memiliki nilai yang dihasilkan dari satu atau beberapa atribut2 lainnya, dan tidak harus berasal dari satu entitas.

2.4.3 *Relationship Type*

Menurut Connolly dan Begg (2010, p376), *Relationship Type* adalah sekumpulan hubungan antara satu atau lebih tipe-tipe *entity*. Derajat dari *relationship* adalah jumlah dari partisipasi (*participating*) tipe *entity* dalam sebuah tipe *relationship* tertentu. Sebuah *relationship* berderajat dua disebut *binary*; *relationship* berderajat tiga disebut sebagai *ternary*; dan *relationship* berderajat empat disebut sebagai *quarternar*.

Relationship digambarkan dengan sebuah garis yang menghubungkan *entity* yang saling berhubungan. Garis tersebut diberi nama sesuai dengan nama hubungannya dan diberi tanda panah satu arah disamping nama hubungannya.



Gambar 2.4 Representasi Diagram dari *Relationship*

(Sumber : Connolly, 2010, p376)

2.4.4 Kunci (*Key*)

Menurut Connolly dan Begg (2010, p150), kunci relasi sangat dibutuhkan untuk mengidentifikasi satu atau lebih atribut yang memiliki nilai unik setiap tuple *dalam relasi*. *Macam-macam kunci relasi :*

1. Kunci Sederhana (*Simple Key*)

Kunci Sederhana adalah suatu kunci yang dibentuk oleh satu atribut.

2. Kunci Komposit (*Composite Key*)

Kunci Komposit adalah kunci yang disusun berdasarkan lebih dari satu atribut.

3. Kunci Kandidat (*Candidate Key*)

Kunci Kandidat adalah suatu atribut atau satu set minimal atribut yang mengidentifikasi secara unik suatu kejadian spesifik dari *entity*.

4. Kunci Primer (*Primary Key*)

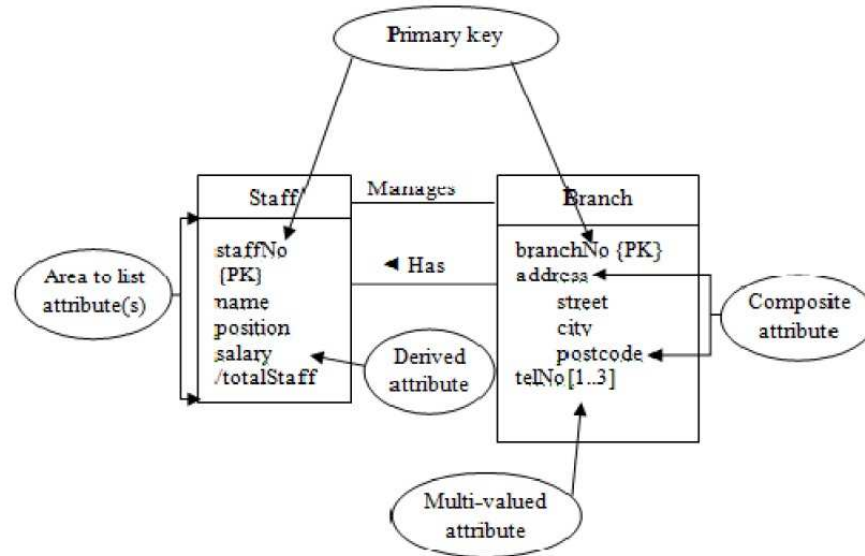
Kunci Primer adalah satu atribut atau satu set minimal atribut yang tidak hanya mengidentifikasi secara unik suatu kejadian spesifik, tapi juga dapat mewakili setiap kejadian dari suatu *entity*.

5. Kunci Alternatif (*Alternative Key*)

Kunci Alternatif adalah kunci kandidat yang tidak terpakai sebagai kunci primer.

6. Kunci Tamu (*Foreign key*)

Kunci Tamu adalah satu atribut yang melengkapi satu hubungan (*relationship*) yang menunjukkan ke induknya.



Gambar 2.5 Representasi Diagram *Entity Employee* dan Cabang Beserta Atribut dan *Primary Key*
(Sumber : Connolly, 2010, p382)

2.5 Diagram Aliran Data (DFD)

DFD adalah suatu teknik analisa struktur dimana *system analyst* bisa meletakkan semua representasi grafis dari proses data yang melalui organisasi (Kendall, 2005, p192). DFD mampu menggambarkan sistem informasi secara logikal maupun fisikal. DFD memiliki empat jenis simbol, yaitu : aliran data (*data flow*), penyimpanan data (*data store*), proses (*process*), sumber (*source*).

Menurut Whitten, et.al (2004, p334), "*Data Flow Diagram (DFD) is a tool that depicts the flow of data through a system and the work or processing performed by that system*" , dapat diartikan sebagai *DFD* adalah sebuah alat yang menggambarkan aliran data sampai sebuah sistem selesai dan kerja atau proses dan dilakukan dalam sistem tersebut.

Empat Komponen *DFD*, antara lain :

1. *External agent*

Menurut Whitten, et.al (2004 ,p 363), "*External agents define a person, an organization unit, another system or another organization that lies outside to scope of the project but that interacts with the system being studied*", dapat diartikan sebagai *external agent* mendefinisikan orang, sebuah unit organisasi, sistem lain atau organisasi lain yang berada di luar sistem proyek tetapi yang mempengaruhi kerja sistem.

2. *Process*

Menurut Whitten, et.al (2004, 347), proses adalah penyelenggaraan kerja atau jawaban, datangnya aliran data atau kondisinya. Menurut Whitten, et.al (2004, 347) ada beberapa bentuk proses diantaranya :

- a. Bentuk *Gane dan Sarson*
- b. Bentuk *DeMarco/Yourdon*

c. Bentuk *SSADM/IDEFO*

3. *Data stores*

Menurut Whitten, et.al (2004, p366) *Data stores* adalah tempat penyimpanan data.

2.6 Jenis-jenis DFD

Jenis-jenis DFD antara lain:

1. Level 0 (Diagram konteks) Level ini terdapat sebuah proses yang berada di posisi pusat.
2. Level 1 (Diagram Nol) Level ini merupakan sebuah proses yang terdapat di level nol yang dipisahkan menjadi beberapa proses lainnya.
3. Level 2 (Diagram Rinci) Pada level ini merupakan diagram yang merincikan diagram dari level 1
 - Tanda ‘*’ digunakan hanya jika proses tersebut tidak bisa dirincikan lagi.

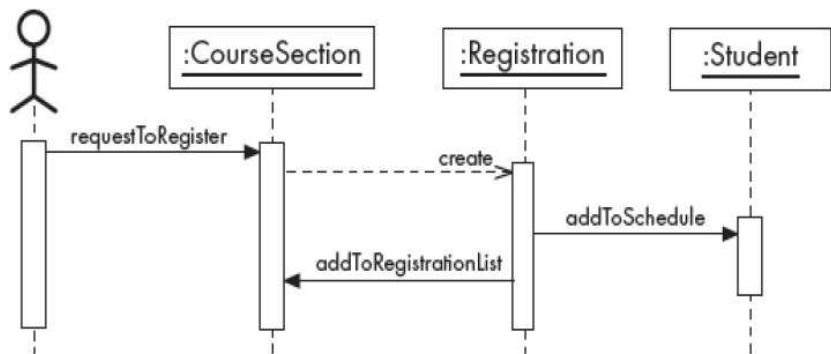
Contoh : 2.0*, artinya proses level rendah tidak bisa dirincikan lagi.

- Penomoran yang dilakukan berdasarkan urutan proses.

2.7 Sequence Diagram

Sequence diagram menggambarkan bagaimana objek berinteraksi satu sama lain melalui pesan dalam *use case* atau suatu operasi. *Sequence* diagram menggambarkan bagaimana pesan dikirim dan diterima antara objek dan dalam satu urutan.

Sebuah *sequence* diagram memiliki aktor beserta dengan pesan dan fungsi dimana pesan tersebut dikerjakan dengan berurutan seperti gambar berikut :



Gambar 2.6 Contoh *Sequence Diagram*

2.8 Flowchart

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urutan-urutan prosedur dari suatu program. Flowchart menolong analis dan programmer untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian.

Flowchart biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut.

2.8.1 Pedoman-Pedoman dalam membuat *flowchart*

Bila seorang analis dan programmer akan membuat *flowchart*, ada beberapa petunjuk yang harus diperhatikan, seperti :

1. *Flowchart* digambarkan dari halaman atas ke bawah dan dari kiri ke kanan.
2. Aktivitas yang digambarkan harus didefinisikan secara hati-hati dan definisi ini harus dapat dimengerti oleh pembacanya.
3. Kapan aktivitas dimulai dan berakhir harus ditentukan secara jelas.
4. Setiap langkah dari aktivitas harus diuraikan dengan menggunakan deskripsi kata kerja, misalkan menghitung pajak penjualan.
5. Setiap langkah dari aktivitas harus berada pada urutan yang benar.
6. Lingkup dan *range* dari aktifitas yang sedang digambarkan harus ditelusuri dengan hati-hati. Percabangan-percabangan yang memotong aktivitas yang sedang digambarkan tidak perlu digambarkan pada *flowchart* yang sama. Simbol konektor harus digunakan dan percabangannya diletakan pada halaman yang terpisah atau hilangkan seluruhnya bila percabangannya tidak berkaitan dengan sistem.
7. Gunakan simbol-simbol *flowchart* yang standar.

2.8.2 Jenis-jenis *flowchart*

Flowchart terbagi atas lima jenis, yaitu :

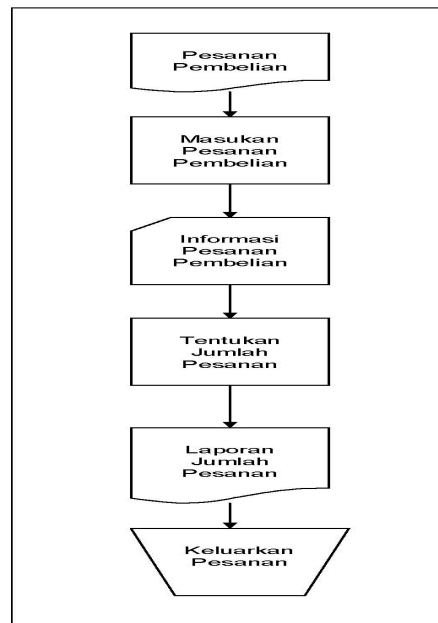
- *Flowchart* Sistem (*System Flowchart*)
- *Flowchart Paperwork* / *Flowchart* Dokumen (*Document Flowchart*)
- *Flowchart* Skematik (*Schematic Flowchart*)
- *Flowchart* Program (*Program Flowchart*)
- *Flowchart* Proses (*Process Flowchart*)

2.8.3 Flowchart sistem

Flowchart Sistem merupakan bagan yang menunjukkan alur kerja atau apa yang sedang dikerjakan di dalam sistem secara keseluruhan dan menjelaskan urutan dari prosedur-prosedur yang ada di dalam sistem. Dengan kata lain, *flowchart* ini merupakan deskripsi secara grafik dari urutan prosedur-prosedur yang terkombinasi yang membentuk suatu sistem.

Flowchart Sistem terdiri dari data yang mengalir melalui sistem dan proses yang mentransformasikan data itu. Data dan proses dalam flowchart sistem dapat digambarkan secara *online* (dihubungkan langsung dengan komputer) atau *offline* (tidak dihubungkan langsung dengan komputer, misalnya mesin tik, cash register atau kalkulator).

Contoh sederhana untuk flowchart sistem:



Gambar 2.7 Contoh *Flowchart* Sistem

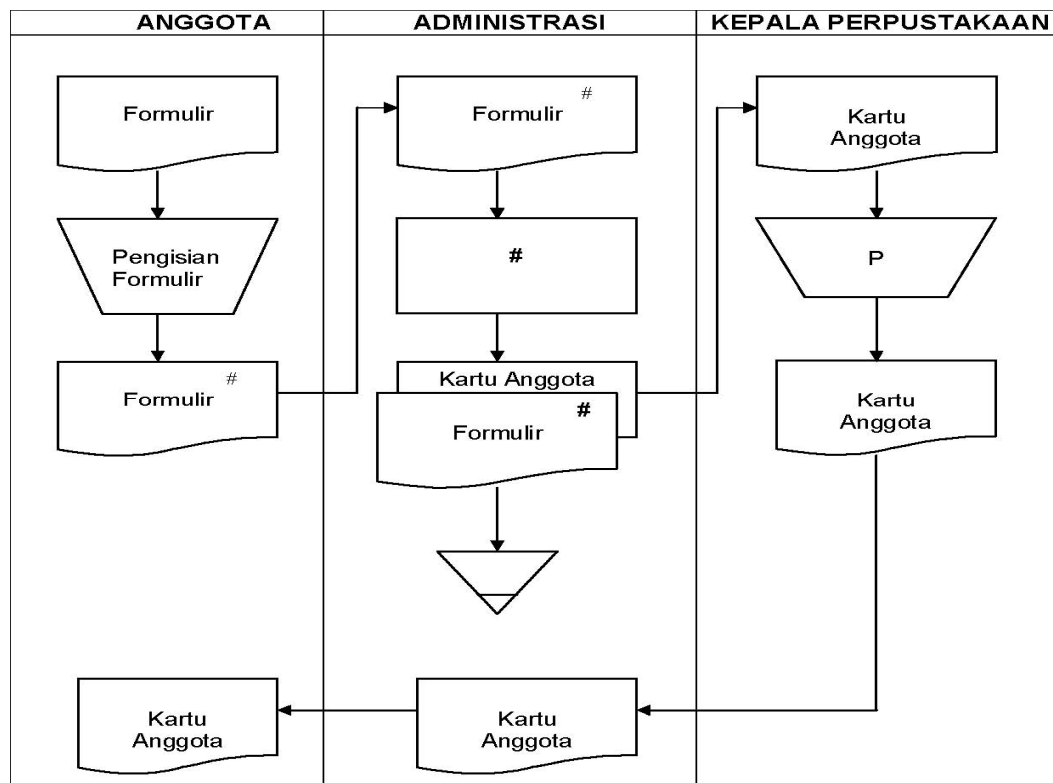
2.8.4 *Flowchart paperwork / flowchart dokumen*

Flowchart Paperwork menelusuri alur dari data yang ditulis melalui sistem. *Flowchart Paperwork* sering disebut juga dengan *Flowchart Dokumen*.

Kegunaan utamanya adalah untuk menelusuri alur *form* dan laporan sistem dari satu bagian ke bagian lain baik bagaimana alur *form* dan laporan diproses, dicatat dan disimpan.

contoh *flowchart* mengenai alur pembuatan kartu anggota untuk suatu perpustakaan.

FLOW DOKUMEN SISTEM BARU CALON ANGGOTA PERPUSTAKAAN



Gambar 2.8 Contoh *Flowchart Paperwork*

KETERANGAN :

: Masukkan data calon anggota ke dalam komputer (proses pengisian data)

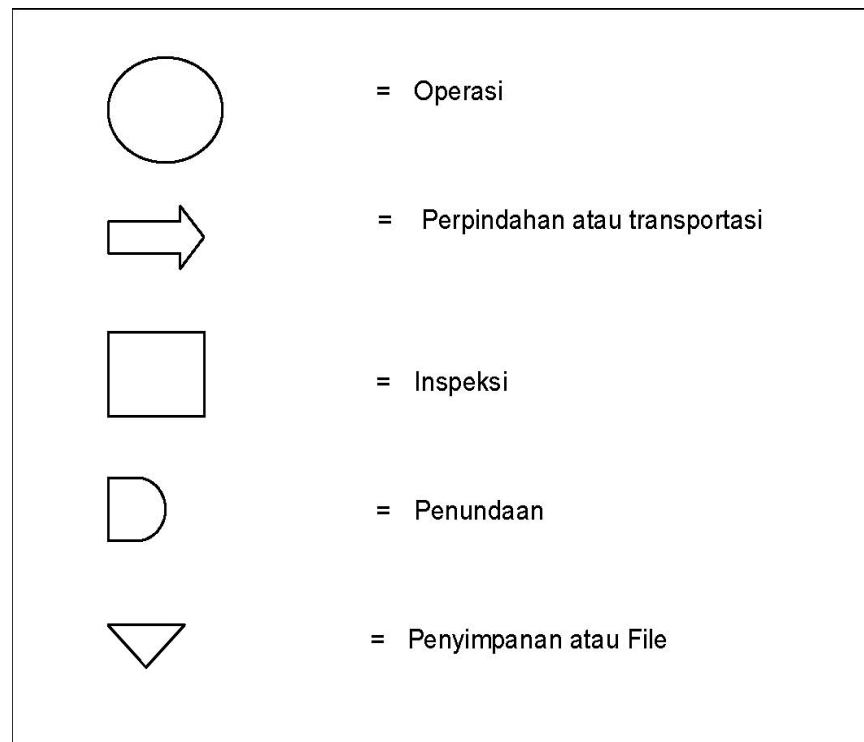
P : Tanda tangan dan validasi data

2.8.5 Flowchart Proses

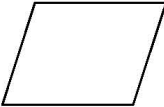
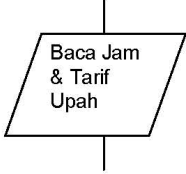

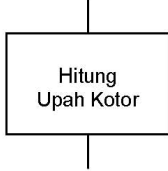

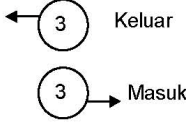

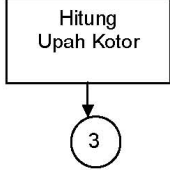
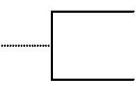
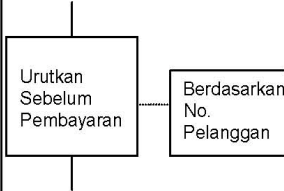
Flowchart Proses merupakan teknik penggambaran rekayasa industrial yang memecah dan menganalisis langkah-langkah selanjutnya dalam suatu prosedur atau sistem.

2.8.6 Simbol-simbol *Flowchart*

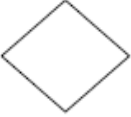
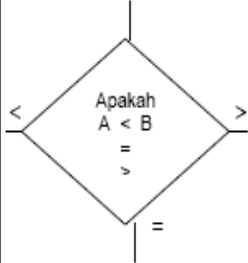



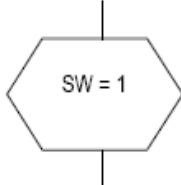

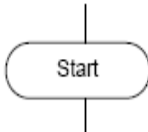

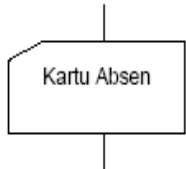
Simbol-simbol flowchart yang biasanya dipakai adalah simbol-simbol *flowchart* standar yang dikeluarkan oleh ANSI dan ISO.



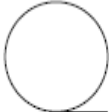
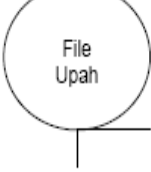

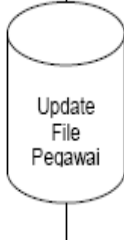
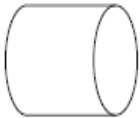
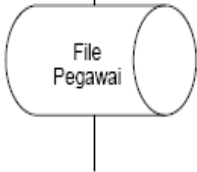



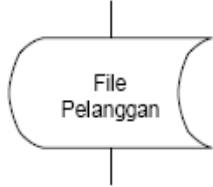





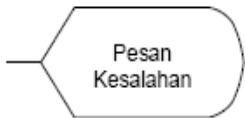

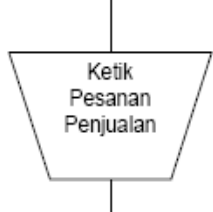
Gambar 2.9 Simbol *Flowchart* Proses



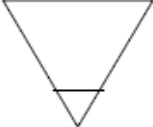
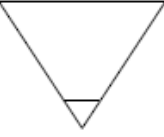
SIMBOL	ARTI	CONTOH
<p>Input / Output</p> 	<p>Merepresentasikan Input data atau Output data yang diproses atau Informasi.</p>	
<p>Proses</p> 	<p>Mempresentasikan operasi</p>	
<p>Penghubung</p> 	<p>Keluar ke atau masuk dari bagian lain flowchart khususnya halaman yang sama</p>	
<p>Anak Panah</p> 	<p>Merepresentasikan alur kerja</p>	
<p>Penjelasan</p> 	<p>Digunakan untuk komentar tambahan</p>	

Gambar 2.10 Simbol *Flowchart* Standar

SIMBOL	ARTI	CONTOH
<p>Keputusan</p> 	<p>Keputusan dalam program</p>	
<p>Predefined Process</p> 	<p>Rincian operasi berada di tempat lain</p>	
<p>Preparation</p> 	<p>Pemberian harga awal</p>	
<p>Terminal Points</p> 	<p>Awal / akhir flowchart</p>	
<p>Punched card</p> 	<p>Input / output yang menggunakan kartu berlubang</p>	

SIMBOL	ARTI	CONTOH
<p>Dokumen</p> 	<p>I/O dalam format yang dicetak</p>	
<p>Magnetic Tape</p> 	<p>I/O yang menggunakan pita magnetik</p>	
<p>Magnetic Disk</p> 	<p>I/O yang menggunakan disk magnetik</p>	
<p>Magnetic Drum</p> 	<p>I/O yang menggunakan drum magnetik</p>	

SIMBOL	ARTI	CONTOH
<p>On-line Storage</p> 	<p>I/O yang menggunakan penyimpanan akses langsung</p>	
<p>Punched Tape</p> 	<p>I/O yang menggunakan pita kertas berlubang</p>	
<p>Manual Input</p> 	<p>Input yang dimasukkan secara manual dari keyboard</p>	
<p>Display</p> 	<p>Output yang ditampilkan pada terminal</p>	
<p>Manual Operation</p> 	<p>Operasi Manual</p>	

SIMBOL	ARTI	CONTOH
<p>Communication Link</p> 	<p>Transmisi data melalui channel komunikasi, seperti telepon</p>	<p>Komputer  Terminal</p>
<p>Off-line Storage</p> 	<p>Penyimpanan yang tidak dapat diakses oleh komputer secara langsung</p>	

Gambar 2.11 Simbol *Flowchart* tambahan

2.9 Web

Menurut Williams & Sawyer (2004, p6) tentang *World Wide Web* (*www*), biasa disebut dengan *web*, adalah sistem yang saling berhubungan pada komputer internet(disebut juga server) yang mana mendukung dokumen dengan format spesial dalam bentuk *multimedia*.

Menurut Hongjun Lu (2008), *Web* menyediakan lingkungan yang *user-friendly* untuk penghasil informasi pada *web*. *Web* merupakan *CGI script* yang dapat diprogram yang dapat ditambahkan pada *web server* yang sudah diinstal.

2.10 *PHP*

PHP singkatan dari *PHP Hypertext Preprocessor* yang digunakan sebagai bahasa script *server-side* dalam pengembangan Web yang disisipkan dalam dokumen *HTML*. Penggunaan *PHP* memungkinkan web dapat dibuat dinamis sehingga pemeliharaan situs web tersebut menjadi lebih mudah dan efisien. *PHP* merupakan *software Open-Source* yang disebar dan dilisensikan secara gratis serta dapat di-*download* secara bebas dari situs resminya (www.php.net). *PHP* ditulis dengan menggunakan bahasa C.

Saat ini *PHP* amat populer dan menggantikan *Perl* yang sebelumnya juga populer sebagai bahasa *scripting web*. *PHP* telah menjadi modul *Apache* terpopuler (menurut www.securityspace.com), melebihi *FrontPage* dan *Mod Perl*. Dan menurut hasil survei www.netcraft.co.uk, *PHP* terus meningkat penggunaannya dan telah digunakan pada jutaan domain dan jutaan alamat *IP*. *PHP* telah digunakan oleh berbagai situs populer baik luar negeri maupun situs dalam negeri.

2.11 *Database yang Dapat Didukung PHP*

Salah satu fitur yang dapat diandalkan oleh *PHP* adalah dukungannya terhadap banyak basisdata. Berikut basis data yang dapat didukung oleh *PHP*

:

- *Adabas D*
- *dBase*
- *Direct MS-SQL*
- *Empress*
- *FilePro (read only)*

- *Front Base*
- *Hyperwave*
- *IBM DB2*
- *Informix*
- *Ingres*
- *Interbase*
- *MSQL*
- *MySQL*
- *ODBC*
- *Oracle (OCI7 dan OCI8)*
- *Ovrimos*
- *PostgrSQL*
- *Solid*
- *SYbase*
- *Unix DBM*
- *Velocis*

2.12 Sintaks Dasar PHP

Sintaks Program *PHP* ditulis dalam apitan tanda khusus *PHP*.

Ada empat macam pasangan *tag PHP* yang dapat digunakan untuk menandai *blok script PHP* :

1. `<?php ... ?>`
2. `<script language="PHP"> ... </script>`
3. `<? ... ?>`
4. `<% ... %>`

Cara pertama dan kedua merupakan cara yang paling umum digunakan sekalipun cara ketiga tampak lebih praktis karena cara ketiga tidak selalu diaktifkan pada konfigurasi *file* php ini yang terdapat pada direktori *c:\apache\php*. Cara keempat juga dimungkinkan sebagai kemudahan bagi Anda yang sudah terbiasa dengan ASP (*Active Server Pages*). Namun, bila itu tidak dikenal, maka harus dilakukan pengaktifan pada *file* konfigurasi *PHP* ini.

2.13 *SQL Server*

SQL Server adalah sebuah sistem manajemen basis data relasional yang biasa dengan *RDBMS* (*Relational Database Management System*). *SQL Server* menangani hubungan antara *Client* dengan *Server* yang berupa pengolahan basis data relasional dengan menggunakan *Transact-SQL* (*T-SQL*) sebagai bahasa untuk mengirim permintaan / perintah antara *Client* dan *Server*.

Server adalah suatu objek yang berfungsi untuk menyediakan *Service* terhadap data yang ada, misalnya : analisa, pencarian, dan update data. *Client* adalah suatu bentuk objek dalam bentuk program yang memiliki *User Interface* untuk berkomunikasi atau mengakses data dari *server*.

Untuk mengakses *SQL Server* yang berfungsi sebagai *Server Database* maka terdapat 4 metode akses yang umum digunakan, yaitu

:

- *ADO* (*ActiveX Data Objects*)

- *ODBC (Open Database Connectivity)*
- *OLEDB (Object Linking and Embedding Database)*
- *JDBC (Java Database Connectivity)*

2.14 *MySQL*

MySQL merupakan bahasa pemrograman *open-source* paling populer dan paling banyak digunakan di lingkungan *Linux*. Kepopuleran ini karena ditunjang oleh performansi *query* dari database-nya yang jarang bermasalah. *MySQL (My Structure Query Language)* adalah sebuah program pembuat database yang bersifat *open-source*, artinya siapa saja dapat menggunakannya secara bebas.

MySQL sebenarnya produk yang berjalan pada *platform Linux*. Karena sifatnya yang *open-source*, *MySQL* dapat berjalan pada semua *platform* baik *Windows* maupun *Linux*. Selain itu, *MySQL* juga merupakan program pengakses *database* yang bersifat jaringan sehingga dapat digunakan untuk aplikasi *multiuser* (banyak pengguna). Saat ini *database MySQL* telah digunakan hampir oleh semua pemrogram *database*, terlebih dalam pemrograman *web*.

Kelebihan lain dari *MySQL* adalah penggunaan bahasa *query* yang dimiliki *SQL (Structured Query Language)*. *SQL* adalah suatu bahasa permintaan yang terstruktur dan telah terstandarisasi untuk semua program pengakses database seperti *Oracle, PostgreSQL, SQL Server*, dan lain-lain.

Menurut Arlita, Ismail, dan Putro (2010), *MySQL* adalah *Relational Database Management System (RDMS)* yang didistribusikan secara gratis di

bawah *General Public License* (GPL). *MySQL* merupakan turunan dari salah satu konsep utama dalam *database*, yaitu *Structured Query Language* (SQL).

Sebagai sebuah program penghasil *database*, *MySQL* tidak dapat berjalan sendiri tanpa adanya sebuah aplikasi lain (*interface*). *MySQL* dapat didukung oleh hampir semua program aplikasi baik yang open-source seperti *PHP* maupun yang tidak, yang ada pada *platform Windows* seperti *Visual Basic*, *Delphi*, dan lainnya.

2.15 Interaksi Manusia dan Komputer

Menurut Sneiderman (2010, p32), ada lima faktor manusia terukur yang dapat dijadikan sebagai pusat evaluasi, yaitu :

1. Waktu belajar Waktu yang dibutuhkan oleh *user* untuk mempelajari cara relevan dalam mengerjakan tugas dengan lancar.
2. Kecepatan kinerja Waktu yang diperlukan untuk mengerjakan suatu tugas yang diberikan.
3. Tingkat kesalahan Berapa banyak kesalahan yang dilakukan oleh *user* dan kesalahan- kesalahan seperti apa yang bisa terjadi saat *user* mengerjakan tugas tersebut.
4. Daya ingat Kemampuan *user* mempertahankan pengetahuannya setelah jangka waktu tertentu.
5. Kepuasan subjektif Kepuasan *user* terhadap berbagai aspek dari sistem.

Menurut Shneiderman (2010, p88), terdapat delapan aturan emas dalam desain antarmuka yaitu:

1. Berusaha untuk konsisten

Aturan ini merupakan aturan yang paling sering dilanggar, karena terdapat banyak bentuk konsistensi. Konsisten dalam hal urutan aksi, istilah yang digunakan, menu, *layout*, penggunaan warna, tata letak, kapitalisasi, *font*, dan sebagainya.

2. Menyediakan kebutuhan *universal*

Dengan memahami kebutuhan *user* yang bermacam-macam dan membuat desain fleksibel yang mendukung perubahan dalam konten.

3. Memberikan umpan balik yang informatif

Untuk setiap aksi *user*, harus ada sistem umpan balik. Untuk aksi kecil yang sering dilakukan, respon dapat dibuat dengan sederhana. Sedangkan untuk aksi yang besar dan jarang dilakukan, respon harus dibuat lebih tegas dan jelas.

4. Desain dialog untuk menghasilkan penutupan atau keadaan akhir

Urutan aksi hendaknya disusun ke dalam kelompok kategori awal, tengah, dan akhir. Umpan balik yang informatif dapat memberikan kepuasan pencapaian, rasa lega, sinyal untuk mempersiapkan diri memasuki kelompok kategori aksi selanjutnya.

5. Penawaran pencegahan dan penanganan kesalahan sederhana

Usahakan dalam mendesain suatu sistem, diarahkan agar *user* tidak membuat kesalahan yang serius. Misalnya menyediakan pilihan menu, tidak mengizinkan karakter alfabet pada kotak entri numerik. Jika *user*

melakukan kesalahan, sistem harus dapat mendeteksi kesalahan dan menawarkan instruksi yang sederhana, konstruktif, dan spesifik untuk perbaikan. Contoh, *user* tidak perlu mengetik ulang seluruh perintah, melainkan hanya memperbaiki bagian yang salah saja.

6. Mengizinkan pembalikan aksi yang mudah

Pada suatu sistem aplikasi harus terdapat pembalikan aksi. Fitur ini dapat memperkecil kesalahan, selama *user* tahu bahwa aksi dapat dibatalkan. Pembalikan aksi dapat berupa tindakan tunggal, tugas data *entry*, atau serangkaian aksi seperti *entri* nama dan alamat.

7. Mendukung pusat kendali internal

User yang sudah terbiasa dengan suatu aplikasi, biasanya ingin memiliki kendali atas antarmuka dan tanggapan dari aksinya. Aksi antarmuka yang tidak umum, urutan entri data yang membosankan, kesulitan dalam memperoleh informasi yang dibutuhkan, serta ketidakmampuan menghasilkan aksi yang diinginkan dapat menimbulkan keresahan dan ketidakpuasan pada *user*.

8. Mengurangi beban ingatan jangka pendek

Manusia mempunyai keterbatasan dalam memproses informasi dengan waktu yang singkat. Oleh karena itu diperlukan tampilan yang sederhana, pengurangan jendela-gerak frekuensi, pemberian waktu pelatihan yang cukup untuk kode-kode, hafalan dan rangkaian aksi. Jika diperlukan, akses *online* untuk sintaks, singkatan, kode, dan informasi yang terkait harus disediakan.

2.16 Web Database

Web database merupakan aplikasi penggunaan *web* sebagai *platform* yang menghubungkan pengguna dengan antarmuka satu atau lebih basis data.

Menurut Abdullat (2004), Penggabungan teknologi Internet dan aplikasi *database* yang membuat komputasi lingkungan baru yang dapat digambarkan sebagai kaya dan kompleks.

Keuntungan dari penggunaan *web database* adalah sebagai berikut (Connolly dan Begg, 2010, p1036-p1038) :

- Kesederhanaan (*Simplicity*)

Dalam bentuk asli, HTML sebagai *markup language* sangat mudah untuk di pelajari oleh *developer* maupun penggunaakhir

- Tidak bergantung pada *platform* (*Platform independence*)

Salah satu alasan dibuatnya aplikasi basis data berbasis *web* adalah karena pada umumnya *web-client* (*browser*) tidak bergantung pada *platform* sehingga jika suatu aplikasi akan di jalankan pada sistem operasi yang berbeda tidak memerlukan modifikasi.

- Grafis antarmuka pengguna (*Graphical User Interface / GUI*)

Menyederhanakan dan meningkatkan pengaksesan basisdata, tetapi GUI memerlukan program tambahan yang akan menyebabkan ketergantungan pada *platform*. *Web browser* menyediakan GUI yang mudah digunakan untuk mengakses banyak hal. Memiliki GUI yang baik akan mengurangi biaya pelatihan untuk pengguna akhir

- Standarisasi (*Standardization*)

HTML standar secara *de facto* dimana seluruh *web browser* berada, memungkinkan sebuah dokumen HTML pada satu mesin dibaca oleh pengguna pada mesin lainnya di bagian lain dunia melalui koneksi di *internet* dan *web browser*.

- Dukungan lintas *platform* (*Cross-platform Support*)

Web browser ada secara *virtual* untuk setiap tipe dari *platform* komputer memungkinkan pengguna pada sebagian besar jenis komputer mengakses basis data dari manapun diseluruh bagian dunia.

- Memungkinkan Akses jaringan yang transparan (*Transparent Network Access*)

Akses jaringan terlihat transparan bagi pengguna, kecuali untuk spesifikasi URL, ditangani sepenuhnya oleh *web browser* dan *web server*. Dukungan *built-in* untuk jaringan akan menyederhanakan akses basis data dan mengeliminasi kebutuhan perangkat lunak jaringan dan kompleksitas dari *platform* yang berbeda untuk saling berkomunikasi.

- Penyebaran bisa diukur (*Scalable Deployment*)

Solusi berbasis *web* mampu membuat arsitektur *three-tier* yang menyediakan dasar untuk *scalability*. Dengan menyimpan aplikasi di *server* terpisah, maka *web* mengeliminasi waktu dan biaya yang berhubungan dengan aplikasi penyebaran. Hal ini akan menyederhanakan penanganan *upgrade* dan *Administrasi* pengaturan dari banyak *platform* yang melalui banyak kantor.

- Inovasi (*Innovation*)

Web memungkinkan organisasi untuk menyediakan jasa barudan mencapai konsumen baru melalui aplikasi yang dapat diakses secara global.

Selain memiliki kelebihan-kelebihan yang telah dijelaskan diatas, *web database* juga tidak lepas dari beberapa kelemahan sebagai berikut (Conolly dan Begg, 2010, p1038-p1040):

- Keandalan (*Reliability*)

Ketika sebuah perintah diberikan melalui *internet*, tidak dapat dipastikan perintah tersebut pasti akan dikirim.

- Keamanan (*Security*)

Autentifikasi pengguna dan transmisi data sangat kritis karena banyak pengguna yang tidak dikenal bisa mengakses data.

- Biaya (*Cost*)

Biaya perawatan bisa menjadi semakin mahal bersamaan dengan meningkatnya permintaan dari pengguna.

- Skalabilitas (*Scalability*)

Aplikasi *web* bisa menghadapi *load* data yang sangat banyak yang datang secara di terduga. Hal ini memerlukan pembangunan performa yang kuat dari arsitektur *server* yang sangat sulit diukur.

- Fungsi terbatas dari HTML (*Limited Functionality of HTML*)

Beberapa aplikasi basis data interaktif ada yang tidak terkonversi secara

mudah ke aplikasi berbasis *web* selama masih menyediakan kemudahan yang sama bagi pengguna.

- *Statelessness*

Statelessness dari lingkungan *web* membuat pengaturan dari koneksi basis data dan transaksi pengguna menjadi sulit dan membutuhkan aplikasi untuk merawat informasi tambahan.

- *Bandwidth*

Kegiatan di dalam *internet* memerlukan *bandwidth* bahkan untuk tugas yang sangat sederhana sekalipun sehingga masalah *bandwidth* menjadi penting.

- Kinerja (*Performance*)

Banyak bagian dari *web database* yang kompleks berpusat pada interpretasi bahasa yang akhirnya membuatnya lebih lambat dari pada basis data tradisional.

- Perangkat pembangunan yang belum siap (*Imaturity of development tools*)

Pembangun aplikasi basis data untuk *web* secara cepat bisa mengidentifikasi ketidaksiapan perangkat pembangunan yang ada. Teknologi *web* masih belum matang dan pembangunan lingkungan yang lebih baik dibutuhkan untuk meringankan beban dari *programmer* aplikasi.

2.17 Teori Khusus

2.17.1 Penjualan

Menurut Kotler (2006, p457) penjualan merupakan sebuah proses dimana kebutuhan pembeli dan kebutuhan penjual dipenuhi, melalui antar pertukaran informasi dan kepentingan, jadi konsep penjualan adalah cara untuk mempengaruhi konsumen untuk membeli produk yang ditawarkan.

Ciri-ciri penjualan menurut (Force one, 2006) adalah :

1. Push atau mendorong atau penyebaran.
2. Ditargetkan kepada pedagang atau *salesman*.
3. Lebih mengandalakan harga dan distribusi.
4. Berdampak jangka pendek dan menengah 1-6 bulan.
5. Berkepentingan menabuh jumlah pelanggan terdaftar.
2. Rasio pelanggan aktif atau inti bertambah
3. Frekwensi transaksi atau *repeat order*
4. Unit *volume* karton transaksi naik atau bulan produktivitas *outlet*

2.17.2 Distribusi

Sistem distribusi adalah pengaturan penyaluran barang dan jasa dari produsen ke konsumen.

Sistem distribusi adalah Seluruh *set-up* yang terdiri dari prosedur, metode, peralatan, dan fasilitas, dirancang dan saling berhubungan untuk memfasilitasi dan memantau arus barang atau jasa dari sumber ke pengguna akhir.

Jenis-jenis Sistem Distribusi, antara lain :

1. Sistem Distribusi Jalan Pendek atau Langsung yaitu sistem distribusi yang tidak menggunakan saluran distribusi
2. Sistem Distribusi Jalan Panjang atau Tidak Langsung yaitu sistem distribusi yang menggunakan saluran distribusi dalam kegiatan distribusinya biasanya melalui agen.

2.17.3 Persediaan

Menurut Nasution (2003, p103), persediaan adalah sumber daya yang menganggur yang menunggu proses lebih lanjut. Proses lebih lanjut tersebut berupa kegiatan produksi pada sistem manufaktur, kegiatan pemasaran pada sistem distribusi ataupun kegiatan konsumsi pangan pada sistem rumah tangga.

Menurut Freddy Rangkuti (2004, p1), persediaan adalah sebagai berikut. "Persediaan merupakan bahan-bahan, bagian yang disediakan, dan bahan-bahan dalam proses yang terdapat dalam perusahaan untuk proses produksi, serta barang-barang jadi atau produk yang disediakan untuk memenuhi permintaan dari konsumen atau pelanggan setiap waktu."

Jadi dapat disimpulkan bahwa persediaan adalah bahan-bahan, bagian yang disediakan, dan bahan-bahan dalam proses yang terdapat dalam perusahaan untuk proses produksi, serta barang-barang jadi atau produk yang disediakan untuk memenuhi permintaan dari konsumen atau pelanggan setiap waktu yang disimpan dan dirawat menurut aturan tertentu dalam tempat persediaan agar selalu dalam keadaan siap pakai dan dicatat dalam bentuk buku perusahaan.

2.17.4 Fungsi dan Tujuan Persediaan

Fungsi persediaan menurut Freddy Rangkuti (2004, p15) adalah sebagai berikut:

1. Fungsi *Batch Stock* atau *Lot Size Inventory*

Penyimpanan persediaan dalam jumlah besar dengan pertimbangan adanya potongan harga pada harga pembelian, efisiensi produksi karena proses produksi yang lama, dan adanya penghematan di biaya angkutan.

2. Fungsi *Decoupling*

Merupakan fungsi perusahaan untuk mengadakan persediaan *decouple*, dengan mengadakan pengelompokan operasional secara terpisah-pisah.

3. Fungsi Antisipasi

Merupakan penyimpanan persediaan bahan yang fungsinya untuk penyelamatan jika sampai terjadi keterlambatan datangnya pesanan bahan dari pemasok atau leveransir. Tujuan utama adalah untuk menjaga proses konversi agar tetap berjalan dengan lancar.