BAB 2

LANDASAN TEORI

2.1 *Database* dan *DBMS* (*Database Management System*)

2.1.1 Pengertian *Database*

Database adalah sekumpulan koleksi data yang berhubungan secara logikal, dan sebuah deskripsi dari data tersebut, didesain untuk menemukan keperluan informasi pada sebuah perusahaan (Conolly, p15). *Database* merupakan tempat penyimpanan data yang besar yang dapat digunakan secara bersamaan oleh banyak pengguna dan berisi deskripsi dari data itu sendiri selain data operasional milik perusahaan.

Menurut Hoffer, Prescott, dan McFadden, *database* adalah sekumpulan organisasi data yang berelasi secara logikal. *Database* dapat memiliki banyak ukuran dan tingkat kompleksitas.

2.1.2 Pengertian *DBMS* (*Database Management System*)

Menurut James A. Hall, *DBMS* adalah sebuah sistem perangkat lunak khusus yang diprogram untuk mengetahui elemen data mana yang bisa diakses (didapatkan otorisasinya) oleh pemakai.

Menurut Connoly, *DBMS* atau *Database Management System* merupakan sebuah perangkat lunak yang memungkinkan pengguna untuk mendefinisikan, membuat, mengambil data, dan mengontrol akses kepada *database* (Conolly, p16). *DBMS* merupakan sebuah perangkat lunak yang menginterasikan *database* dengan aplikasi program pada pengguna.

Biasanya, *DBMS* menyediakan fasilitas sebagai berikut :

a. *Data Definition Language* (*DDL*) memperbolehkan pengguna untuk mendeskripsikan *database*, misalnya merinci tipe dan batasan data yang akan disimpan dalam *database*.

b. *Data Manipulation Language* (*DML*) memperbolehkan pengguna untuk memanipulasi data, misalnya memasukkan data, menghapus data, dan mendapatkan data dari *database*.

c. Menyediakan akses terkontrol ke *database*, misalnya *security system*, *integrity system*, *concurrency control system*, *recovery control system*, *user-accesible catalog*.

2.1.3 Komponen *DBMS*

Ada 5 komponen utama pada *DBMS*, yaitu (Conolly, p18) :

1. Hardware (Perangkat Keras)

   DBMS dan aplikasi membutuhkan perangkat keras untuk dapat berjalan. Perangkat kerasnya dapat berupa satu *personal computer*, satu *mainframe*, maupun jaringan yang terdiri dari banyak komputer. Perangkat keras yang dibutuhkan bergantung dari permintaan dari organisasi dan *DBMS* yang digunakan.

2. Software (Perangkat Lunak)

   Komponen dari perangkat lunak terdiri dari perangkat lunak *DBMS* itu sendiri dan program aplikasi, bersama dengan sistem aplikasi, termasuk perangkat lunak jaringan jika *DBMS* digunakan melalui jaringan.

3. Data

Mungkin komponen yang terpenting pada *DBMS*, terutama dari sudut pandang pengguna, adalah data. Data berperan sebagai jembatan antara komponen mesin (*hardware* dan *software*) dan komponen manusia (prosedur dan manusia). Database berisi baik data, maupun *meta* data, yaitu data tentang data. Struktur dari *database* disebut skema.

4. Prosedur

Prosedur menunjuk pada instruksi dan aturan yang mempengaruhi desain dan penggunaan dari *database*. Para pengguna sistem dan para staf yang mengatur dokumen prosedur *database* yang dibutuhkan dan bagaimana cara menggunakan atau menjalankan sistem. Hal ini mungkin mengandung instruksi dari :

a. Bagaimana cara memasuki *DBMS*
b. Bagaimana menggunakan fasilitas *DBMS* atau program aplikasi tertentu
c. Bagaimana memulai dan mengakhiri *DBMS*
d. Bagaimana membuat salinan dari *database*
e. Bagaimana mengatasi kegagalan perangkat keras dan perangkat lunak. Hal ini termasuk prosedur tentang bagaimana mengidentifikasi komponen yang gagal, bagaimana memperbaikinya, dan mengikuti perbaikan dari kesalahan, bagaimana memulihkan *database*.
f. Bagaimana mengubah struktur dari *table*, mengorganisasi ulang *database* yang terdapat pada lebih dari satu tempat penyimpanan,

memperbaiki performa, atau menyimpan data ke penyimpanan kedua.

5. Manusia

Orang-orang yang berhubungan dengan sistem antara lain :

a. Database designers

Ada dua tipe dari *database designer*, yaitu :

i. *Logical database designer*, tugasnya berhubungan dengan mengidentifikasi data, relasi antar data, dan batasan pada data yang akan disimpan di *database.*

ii. *Physical database designer*, bertugas untuk memutuskan bagaimana desain *logical database* direalisasikan. Hal ini termasuk memetakan desain *logical database* kepada seperangkat *table* dan batasan integritas, memilih struktur penyimpanan yang spesifik dan metode akses data untuk mencapai performa yang baik, dan mendesain aturan keamanan yang dibutuhkan oleh data.

b. Application developers

Ketika *database* diimplementasikan, program aplikasi yang menyediakan fungsi yang dibutuhkan oleh pengguna harus diimplementasikan juga. Ini adalah tanggung jawab dari *application developers*. Biasanya, *application developers* bekerja dari spesifikasi yang diproduksi oleh *system analyst.* Setiap program mengandung kalimat yang meminta *DBMS* untuk

melakukan beberapa operasi pada *database*. Hal ini termasuk mengambil data, memasukkan, mengubah, dan menghapus data.

c. *End-users*

Para pengguna dapat diklasifikasikan berdasarkan bagaimana mereka menggunakan sistem, yaitu :

i. Pengguna dasar

Pengguna dasar adalah pengguna yang terlatih menggunakan *DBMS* secara awam. Mereka mengakses *database* melalui program aplikasi tertulis khusus yang diusahakan untuk membuat operasi sesederhana mungkin. Mereka menggunakan operasi *database* dengan memasukan perintah sederhana atau memilih pilihan dari *menu*. Hal ini berarti mereka tidak perlu mendalami topik khusus mengenai *database* atau *DBMS*.

ii. Pengguna berpengalaman

Pengguna berpengalaman biasanya sudah mengenal struktur dari *database* dan fasilitas yang ditawarkan oleh *DBMS*. Pengguna berpengalaman mungkin menggunakan bahasa *query* yang dengan tingkat tinggi seperti SQL untuk melakukan operasi yang dibutuhkan.

2.1.4 Keuntungan dan Kerugian *DBMS*

Keuntungan dari *DBMS* adalah (Connolly, p26) :

a. Pengaturan dari data yang berlebihan

b. Konsistensi data

c. Mendapat informasi yang lebih dari data yang berjumlah sama
d. Pembagian data
e. Memperbaiki integritas data
f. Memperbaiki keamanan
g. Pelaksanaan standar
h. Keseimbangan ekonomi
i. Keseimbangan dari permintaan yang berselisih
j. Memperbaiki pengaksesan dan tanggapan data
k. Meningkatkan produktifitas
l. Memperbaiki pemeliharaan melalui independensi data
m. Meningkatkan persetujuan
n. Memperbaiki *backup* dan layanan perbaikan

Kerugian dari *DBMS* adalah (Connolly, p29) :

a. Kompleks
b. Ukuran
c. Harga dari *DBMS*
d. Harga dari perangkat keras yang dibutuhkan
e. Harga pengkonversian
f. Performa
g. Dampak yang lebih besar pada kegagalan

2.2 *Entity Relationship Modeling*

2.2.1 Tipe Entitas

Tipe entitas adalah sekumpulan objek dengan properti yang sama, yang diidentifikasikan oleh perusahaan sebagai pemilik dari keberadaan

yang independen (Connolly, p343). Konsep dasar dari *Entity Relationship Model* (*ER Model*) adalah tipe entitas, yang merepresentasikan sekumpulan dari objek dalam dunia nyata dengan properti yang sama. Sebuah tipe entitas memiliki keberadaan yang independen dan dapat pula objek dengan keberadaan yang nyata atau objek dengan keberadaan abstrak. Desainer berbeda dapat mendidentifikasikan entitas yang berbeda pula karena tidak ada peraturan tentang definisi tipe entitas yang formal.

Menurut McLeod Jr., tipe entitas dapat berupa suatu elemen lingkungan, sumber daya, atau transaksi, yang begitu pentingnya bagi perusahaan sehingga didokumentasikan dengan data.

2.2.2 Tipe Relasi

Tipe *relationship* adalah seperangkat hubungan yang berarti antar tipe entitas (Connolly, p346). Tipe *relationship* adalah sekumpulan hubungan antara satu atau lebih tipe entitas yang berpartisipasi. Setiap tipe *relationship* diberikan nama yang menjelaskan fungsinya.

Menurut McLeod Jr., hubungan (*relationship*) adalah suatu asosiasi antara dua jenis entitas.

2.2.3 Atribut

Atribut adalah properti dari tipe entitas atau tipe *relationship* (Connolly, p350). Atribut memegang nilai yang menjelaskan setiap peristiwa entitas dan merepresentasikan bagian utama dari data yang disimpan dalam *database*.

2.2.4 *Keys*

Candidate key adalah sekumpulan minimal dari atribut yang mengidentifikasikan secara unik setiap peristiwa dalam sebuah tipe entitas (Connolly, p352). Ada dua jenis *candidate key*, yaitu :

a. *Primary key*

Primary key adalah *candidate key* yang dipilih untuk mengidentifikasikan secara unik setiap peristiwa sebuah tipe entitas.

b. *Composite key*

Composite key adalah *candidate key* yang memiliki atribut dua atau lebih.

2.2.5 Entiti Kuat dan Entity Lemah

Ada dua klasifikasi dari tipe enitas, yaitu (Connolly, p354):

a. *Strong entity type*

Strong entity type adalah sebuah tipe entitas yang tidak bergantung pada keberadaan tipe entitas yang lain.

b. *Weak entity type*

Weak entity type adalah sebuah tipe entitas yang bergantung pada keberadaan tipe entitas lain.

2.2.6 *Structural Constraints*

Ada beberapa *structural constraint*, yaitu (Connoly, p356) :

1. Derajat hubungan *one to one* (1 : 1)
2. Derajat hubungan *one to many* (1 : *)
3. Derajat hubungan *many to many* (* : *)

4. *Multiplicity* untuk relasi yang kompleks

Multiplicity untuk relasi kompleks adalah jumlah atau jarak dari beberapa kemungkinan dari sebuah tipe entitas adalah relasi ke-n ketika nilai (n-1) nya tetap.

5. Batasan *cardinality* dan partisipasi

Cardinality menerangkan tentang jumlah maksimum dari relasi yang mungkin timbul untuk entitas yang ikut bagian pada tipe relasi yang diberikan. Partisipasi menentukan apakah semua atau hanya beberapa entitas yang ikut bagian dalam sebuah relasi.

2.2.7 Persiapan *ERD*

Menurut McLeod Jr., ada tujuh langkah dalam mempersiapkan *ERD*, yaitu (McLeod Jr., p305) :

1. Mengidentifikasi entitas

Manajemen menentukan elemen lingkungan, sumber daya, dan transaksi mana yang akan dijelaskan dengan data.

2. Mengidentifikasi hubungan

Tiap entitas dihubungkan dengan entitas lain melalui suatu jenis tindakan.

3. Menyiapkan rancangan *ERD*

Simbol-simbol dibuat sketsanya sehingga jika mungkin, hubungan terbaca dari kiri ke kanan, atau dari atas ke bawah.

4. Memetakan elemen-elemen pada entitas

Elemen-elemen data yang mengidentifikasi dan menjelaskan tiap entitas data didaftarkan di sebelah entitasnya.

5. Membuat analisis data

Elemen-elemen data dipelajari untuk membuat struktur *database* menjadi efisien. Proses melaksanakan analisis data disebut normalisasi. Dan tugasnya adalah menyesuaikan data sehingga serupa dengan serangkaian bentuk-bentuk normal.

a. Bentuk normal pertama (1NF) : Hapuskan semua elemen yang berulang dalam suatu entitas

b. Bentuk normal kedua (2NF) : Pastikan bahwa atribut *descriptor* bergantung pada seluruh *composite key* untuk identifikasi.

c. Bentuk normal ketiga (3NF) : Pastikan bahwa nilai atribut tidak bergantung pada nilai atribut lain dalam entitas yang sama.

6. Menyiapkan *ERD* yang telah dimodifikasi

Hasil dari analisis data disatukan ke dalam satu *ERD* baru. Dengan cara ini, jenis-jenis entitas dan hubungannya diatur sehingga mereka memberikan dasar yang paling efisien untuk rancangan *database*.

7. Menelaah *ERD* bersama pemakai dan memperbaikinya

Spesialis informasi menelaah diagram tersebut bersama para eksekutif, manajer, dan non-manajer pada area pemakai dan memperbaikinya jika perlu.

2.3 *State Transition* Diagram

Diagram bagian juga memodelkan kedinamisan dari sebuah sistem. Menurut Jeffrey L. Whitten, *UML* (*Unified Modelling Language*) memiliki sebuah diagram untuk memodelkan sikap objek khusus yang kompleks (Diagram

statechart) dan sebuah diagram untuk menggambarkan perilaku dari sebuah *use case* atau sebuah metode , yaitu :

1. Diagram *statechart* digunakan untuk menggambarkan perilaku objek khusus yang dinamis. Diagram ini mengilustrasikan siklus hidup objek dalam berbagai kondisi yang dapat diasumsikan oleh objek dan *event-event* yang menyebabkan keadaan objek beralih dari suatu *state* ke *state* lain.
2. Diagram aktivitas secara grafis digunakan untuk menggambarkan rangkaian aliran aktivitas baik itu proses bisnis atau *use case*. Diagram ini juga dapat digunakan untuk menggambarkan aksi yang akan dilakukan saat sebuah operasi dieksekusi, dan hasil dari aksi tersebut.

STD dapat dianalogikan sebagai sebuah peta di mana setiap layarnya berfungsi sebagai sebuah kota. Berikut ini adalah simbol-simbol yang digunakan dalam *STD* :

a. Digunakan untuk tampilan layar

b. Digunakan untuk merepresentasikan alur kontrol. Arah panah mengindikasikan perintah di mana layar menjadi aktif.

c. Digunakan untuk menunjukan terminal awal dari alur kontrol.

d. Digunakan untuk menunjukan terminal akhir dari alur kontrol

2.4 Data *Flow* Diagram

2.4.1 Pengertian Data *Flow* Diagram

Data *Flow Diagram* (Diagram Arus Data) atau *DFD* adalah suatu gambaran grafis dari suatu sistem yang menggunakan sejumlah bentuk-bentuk simbol untuk menggambarkan bagaimana data mengalir melalui suatu proses yang saling berkaitan (McLeod Jr., p316). Walaupun nama diagram ini menekankan pada data, situasinya justru sebaliknya : penekanannya ada pada proses. *DFD* mungkin cara paling alamiah untuk mendokumentasikan proses.

2.4.2 Simbol-Simbol *DFD*

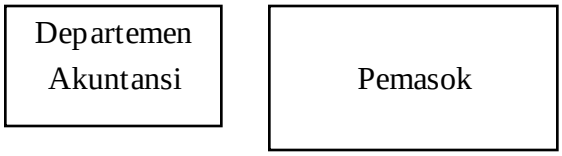
DFD terdiri dari empat simbol. Simbol-simbol itu digunakan untuk (McLeod Jr., p316) :

1. Elemen-elemen lingkungan yang berhubungan dengan sistem

   Elemen-elemen lingkungan berada di luar batas sistem. Elemen-elemen ini menyediakan bagi sistem *input* data dan menerima *output* data dari sistem. Pada *DFD*, tidak dibuat perbedaan antara data dan informasi, semua arus dipandang sebagai data.

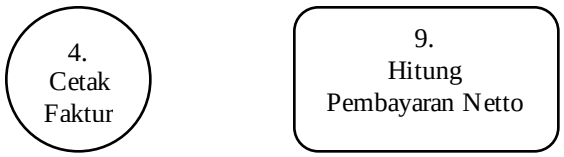
   Nama *terminator* digunakan untuk menggambarkan elemen-elemen lingkungan, yang menandai titik-titik berakhirnya sistem. Suatu *terminator* dapat berupa orang, organisasi, maupun sistem lain.

Terminator digambarkan sebagai suatu kotak atau segi empat dan tiap simbol *terminator* diberi label nama elemen lingkungan.



2. Proses

Proses adalah sesuatu yang mengibah *input* menjadi *output*. Tiap simbol proses diidentifikasikan oleh label. Proses dapat digambarkan dengan lingkaran, segi empat horisontal, atau segi empat tegak dengan sudur-sudut yang membulat.



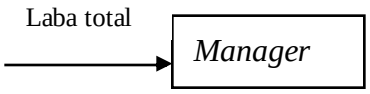
3. Arus Data

Arus data terdiri dari sekelompok elemen data yang berhubungan secara logis yang bergerak dari satu titik atau proses ke titik atau proses yang lain.

Arus data terdiri dari satu atau beberapa struktur data. Struktur data adalah sekelompok elemen data yang menggambarkan suatu hal atau transaksi tertentu. Arus data dapat bercabang (*diverge*) ketika data yang sama bergerak ke beberapa lokasi dalam sistem, dapat pula

memusat (*converge*) untuk menggambarkan beberapa arus data yang sama bergerak ke satu lokasi.

Arus data dapat digambarkan dengan tanda panah, panah tersebut dapat digambar sebagai garis lurus atau lengkung.



4. Penyimpanan data

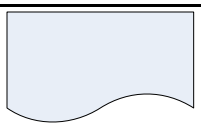
Jika data perlu dipertahankan karena suatu sebab, maka digunakan penyimpanan data, dalam istilah *DFD*, penyimpanan data (data storage) adalah suatu tempat penampungan data.

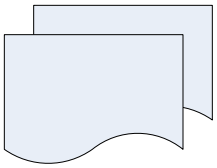
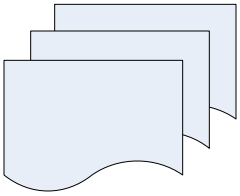
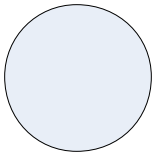
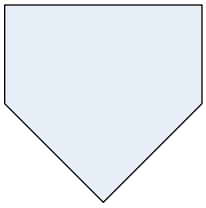
Penyimpanan data dapat digambarkan sebagai satu set garis paralel, segi empat terbuka, atau bentuk lonjong.

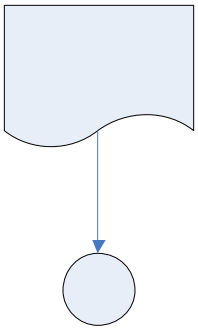
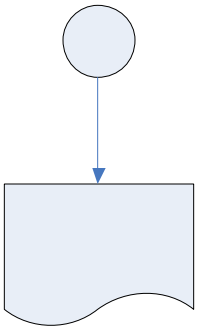
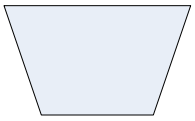
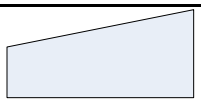
2.5 *Flowchart*

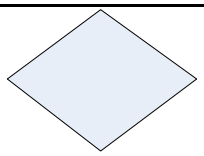
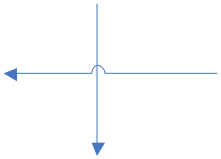
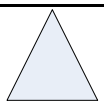
Simbol-simbol standar *flowchart* (Mulyadi, p60) dapat dilihat pada tabel 2.1

Tabel 2.1 Simbol-Simbol *Flowchart*

Simbol	Deskripsi
	**Dokumen** menggambarkan semua jenis dokumen, merupakan formulir yang

	digunakan untuk mencatat data terjadinya suatu transaksi.
	**Dokumen dan tembusannya** menggambarkan dokumen asli dan tembusannya. Nomor lembar dokumen dicantumkan di sudut kanan atas.
	**Banyak dokumen** menggambarkan berbagai jenis dokumen yang digabungkan bersama di dalam satu paket. Nama dokumen dituliskan dalam masing-masing simbol dan nomor lembar dokumen dicantumkan di sudut kanan atas simbol dokumen yang bersangkutan.
	**Penghubung pada halaman yang sama** memungkinkan aliran dokumen berakhir di suatu titik pada halaman tertentu dan kembali dimulai di titik lain pada halaman yang sama.
	**Penghubung pada halaman yang berbeda** menunjukkan ke mana dan bagaimana *flowchart* tersebut terkait satu dengan yang lainnya.

	**Akhir arus dokumen** yang mengarahkan pembaca ke simbol penghubung halaman yang bernomor sama seperti yang tercantum dalam simbol tersebut.
	**Awal arus dokumen** yang berasal dari simbol penghubung halaman yang bernomor sama seperti yang tercantum dalam simbol tersebut.
	**Manual *operation*** menggambarkan pekerjaan manual seperti : menerima order dari pembeli, mengisi formulir, membandingkan, memeriksa dan berbagai jenis kegiatan pencatatan yang lain.
	***On-line computer process*** menggambarkan pengolahan data dengan komputer secara *on-line*. Nama program ditulis di dalam symbol.
	***Keying (typing, verifying)*** menggambarkan pemaskkan data ke dalam computer melalui *on-line terminal.*

	**Keputusan** menggambarkan keputusan yang harus dibuat dalam proses pengolahan data. Keputusan yang dibuat ditulis dalam simbol.
	Garis alir (*flowline*) menggambarkan arah proses pengolahan data.
	**Persimpangan garis alir** akan ditunjukan dengan garis melengkung. Hal ini menunjukan garis alir tidak berhubungan atau terpisah.
	**Awal/akhir (terminal)** menggambarkan awal dan akhir suatu proses
	**Arsip permanen** Menggambarkan tempat penyimpanan dokumen yang tidak akan diproses lagi dalam sistem yang bersangkutan. Tiga jenis pengurutan arsip yaitu : A (menurut abjad), N (menurut nomor urut), T (menurut tanggal)

2.6 Normalisasi

2.6.1 Pengertian Normalisasi

Normalisasi adalah sebuah teknik untuk memproduksi seperangkat hubungan dengan properti yang diinginkan, memberikan kebutuhan data dari sebuah perusahaan (Connolly, p388). Tujuan dari

normalisasi adalah untuk mengidentifikasikan seperangkat relasi yang sesuai yang dapat mendukung kebutuhan data pada sebuah perusahaan.

Karakteristik dari seperangkat relasi tersebut diantaranya adalah :

a. Atribut dengan jumlah terkecil penting untuk mendukung kebutuhan data pada sebuah perusahaan
b. Atribut dengan hubungan logikal terdekat (dideskripsikan sebagai *functional dependency*) ditemukan di relasi yang sama
c. *Redundancy* minimal dengan setiap atribut direpresentasikan hanya sekali dengan pengecualian penting dari atribut yang membentuk semua atau sebagian dari *foreign key*, yang penting untuk menggabungkan relasi yang berhubungan.

2.6.2 Proses Normalisasi

Normalisasi merupakan teknik formal untuk menganalisis relasi berdasarkan primary key (atau *candidate key*) dan fungsional *dependency* (Connolly, p401). Teknik ini meliputi adanya batasan-batasan atau aturan-aturan yang digunakan untuk menguji relasi yang ada, dan menormalisasikan *database* kedalam beberapa tingkat. Ada tiga susunan normalisasi yang biasa disebut sebagai Unnormalized Form (UNF), *First Normal Form* (1NF), *Second Normal Form* (2NF), dan *Third Normal Form* (3NF).

1. *Unnormalized Form* (UNF)

Unnormalized form adalah sebuah tabel yang mengandung satu atau lebih grup yang berulang. *Unnormalized form* adalah bagian awal dari *First Normal Form*.

2. *First Normal Form* (1NF)

First Normal Form adalah sebuah relasi di mana persimpangan antara baris dan kolomnya mengandung satu dan hanya satu nilai.

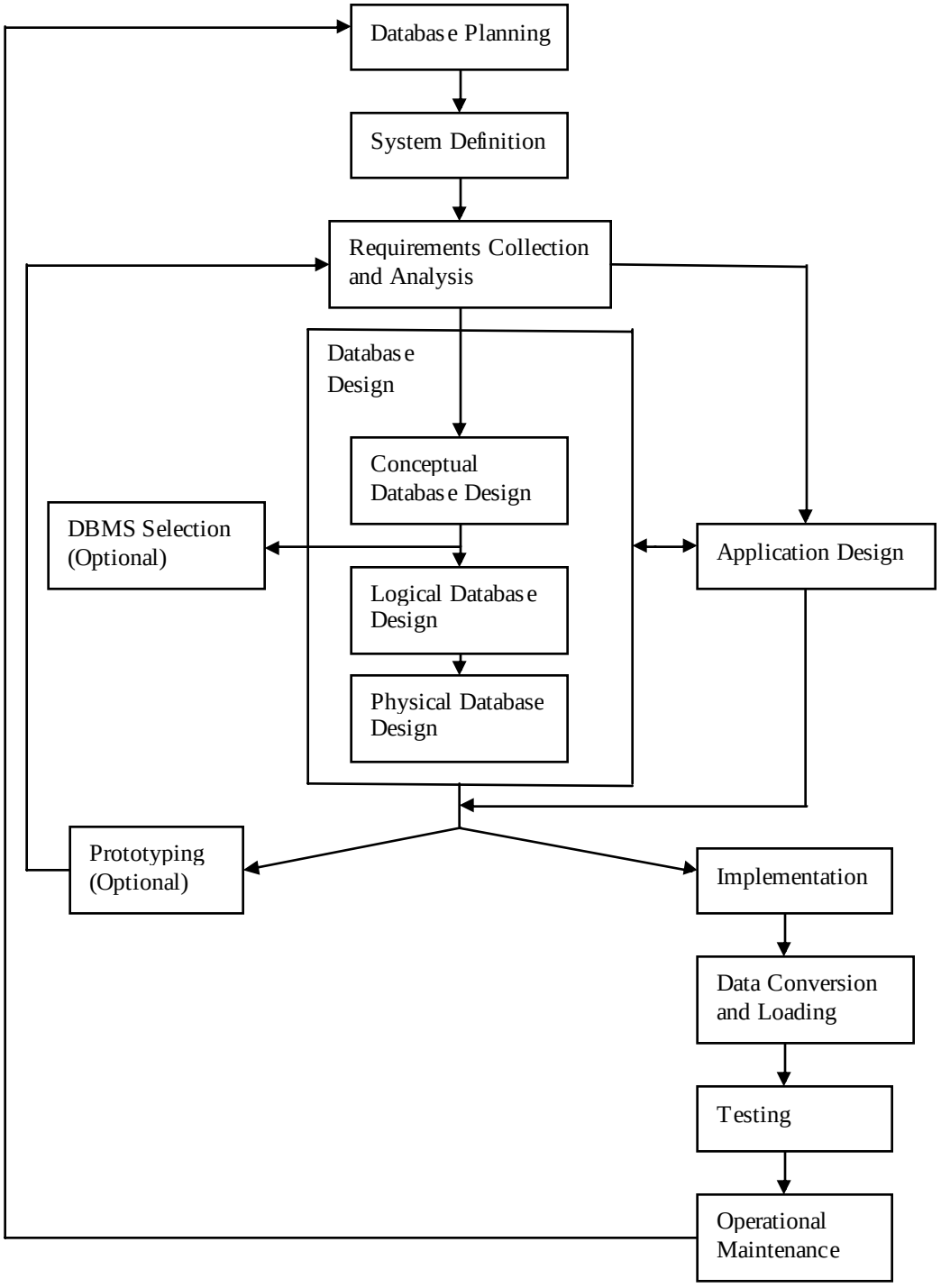
3. *Second Normal Form* (2NF)

Second Normal Form adalah relasi yang ada di *First Normal Form* dan setiap atribut yang *non primary key* menjadi sepenuhnya bergantung pada fungsi dari *primary key*.

4. *Third Normal Form* (3NF)

Third Relational Form adalah relasi yang ada pada *First* dan *Second Normal Form* dan di mana atribut yang *non primary key* beralih menjadi *dependent* pada *primary key*.

2.7 *Database Application Lifecycle*



Gambar 2.1 *Database Lifecycle*

Database lifecycle dibagi menjadi beberapa tahap (Connoly, p284) :

2.7.1 *Database Planning*

Database planning adalah manajemen aktivitas yang memperbolehkan tingkat dari *database system development lifecycle* untuk diwujudkan secara efisien dan efektif (Connoly, p285). Ada 3 kegiatan utama yang digunakan, yaitu :

a. Mengidentifikasi rencana dan tujuan perusahaan dengan tujuan utama untuk memastikan keperluan sistem informasi.
b. Mengevaluasi sistem informasi yang ada untuk mengetahui kekuatan dan kelemahannya.
c. Menentukan *IT* yang dapat memberikan keuntungan yang kompetitif.

2.7.2 System *Definition*

System Definition menjelaskan tentang lingkup dan batasan dari aplikasi *database* dan pandangan utama dari pengguna (Connoly, p286).

2.7.3 *Requirement Collection and Analysis*

Requirement Collection and Analysis merupakan proses dari pengumpulan informasi tentang bagian dari organisasi yang didukung oleh sistem *database,* dan menggunakan informasi untuk mengidentifikasikan kebutuhan untuk sistem baru (Connoly, p288).

2.7.4 *Database Design*

Database design adalah proses membuat desain yang dapat mendukung misi perusahaan dan misi objektif untuk sistem *database* yang dibutuhkan (Connoly, p291). Ada 3 fase dari *database design* yaitu :

a. *Conceptual Database Design*

Conceptual database design adalah proses pembuatan model dari data yang digunakan pada sebuah perusahaan, terpisah dari semua pertimbangan fisikal. *Conceptual database design* melalui 8 tahapan (Connoly, p440), yaitu:

1. Mengidentifikasi tipe *entity*

   Langkah pertama ntuk membuat data model konseptual adalah dengan menjelaskan objek utama yang menarik bagi user. Objek-objek ini adalah tipe *entity* untuk model yang ada. Salah satu metode untuk mengidentifikasi *entity* adalah dengan memeriksa spesifikasi kebutuhan *user*. Metode lain yaitu dengan mencari objek yang memiliki keberadaan pada kenyataan.

2. Mengidentifikasi tipe relasi

   Sesudah mengidentifikasi *entity*, maka langkah selanjutnya adalah dengan mengidentifikasi semua relasi yang ada antara *entity-entity* tersebut. Dapat digunakan metode seperti metode mengidentifikasi *entity*, yaitu dengan melihat spesifikasi kebutuhan user yang merupakan *noun*(kata benda).

3. Mengidentifikasi dan menghubungkan *attribute* dengan *entity* atau relasi

   Langkah berikutnya yaitu dengan mengidentifikasi dan menghubungkan atribut dengan *entity* atau relasi. Metode yang digunakan sama dengan yang digunakan untuk mengidentifikasi *entity*, yaitu dengan cara mencari *noun* dan *noun phrase* pada

spesifikasi kebutuhan perusahaan. Dapat juga dengan cara bertanya kepada perusahaan mengenai hubungan tersebut.

4. Menentukan domain *attribute*

   Tujuan dari langkah ini adalah untuk menjelaskan domain dari semua atribut yang ada pada model. Sebuah domain adalah kumpulan dari nilai yang dimiliki oleh atribut.

5. Menentukan *candidate, primary,* dan *alternate key*

   Langkah ini berhubungan dengan mengidentifikasi *candidate key* untuk *entity* dan kemudian memilih salah satunya untuk dijadikan *primary key. Candidate key* yang tidak terpilih menjadi *primary key* disebut sebagai *alternate key.*

6. Memeriksa *redundacy* model

   Pada langkah ini, terjadi pemeriksaan data model konseptual lokal dengan tujuan utama untuk mengidentifikasi apakah ada *redundancy* dan menghapuskan *redundancy* yang ada.

7. Memvalidasi konseptual model dengan transaksi *user*

   Pada langkah ini, telah didapat data model konseptual lokal yang merepresentasikan kebutuhan data pada perusahaan. Tujuan dari langkah ini adalah untuk memeriksa model dan memastikan apakah model tersebut mendukung transaksi yang dibutuhkan.

8. Meninjau kembali konseptual data model dengan *user*

   Sebelum menyelesaikan konseptual data model, kita perlu meninjau kembali dengan user. Data model konseptual mencakup *ER Diagram* dan dokumentasi yang dapat mendukung penjelasan

mengenai data model. Apabila terdapat anomali pada data model, harus dilakukan perubahan yang berarti, yang mungkin mengharuskan pengulangan dari langkah-langkah sebelumnya.

b. *Logical Database Design*

Logical database design adalah proses pembuatan model dari data yang digunakan pada perusahaan, berdasarkan data model tertentu, tetapi independen pada *DBMS* utama dan pertimbangan fisikal lainnya. *Logical database design* melalui 7 tahapan (Connoly, p440), yaitu:

1. Membuat relasi untuk logikal data model

   Pada tahap ini, didapatkan relasi untuk data model logikal untuk merepresentasikan *entity*, relasi, dan atribut. Komposisi pada setiap relasi digambarkan dengan menggunakan *Database Definition Language* (*DBDL*) untuk *relational database.* Relasi didapatkan dari struktur-struktur yang mungkin terjadi pada data model konseptual, yaitu :

   a. *Strong entity types*
   b. *Weak entity types*
   c. *One-to-many (1:*) binary relationship types*
   d. *One-to-one (1:1) binary relationship types*
   e. *One-to-one (1:1) recursive relationship types*
   f. *Superclass/subclass relationship types*
   g. *Many-to-many (*:*) binary relationship types*
   h. Tipe relationship yang kompleks
   i. Atribut yang *multi-valued*

2. Memvalidasi relasi dengan normalisasi

   Pada langkah ini, terjadi validasi grup-grup atribut pada setiap relasi dengan menggunakan aturan dari normalisasi. Tujuan dari normalisasi adalah untuk memastikan bahwa sekumpulan relasi memiliki atribut yang minimal namun mencukupi untuk mendukung kebutuhan data perusahaan.

3. Memvalidasi relasi dengan transaksi *user*

   Tujuan dari langkah ini adalah untuk memvalidasi data model logikal untuk memastikan bahwa model tersebut mendukung transaksi yang dibutuhkan, yang telah dirinci pada spesifikasi kebutuhan *user*.

4. Memeriksa *integrity constraints*

   Integrity constraint adalah *constraint* yang diharapkan untuk diberlakukan demi menjaga *database* agak tidak menjadi tidak selesai, tidak akurat, dan tidak konsisten.

5. Meninjau *logical* data model dengan *user*

   Data model logikal pada tahap ini seharusnya sudah lengkap dan didokumentasikan secara penuh. Namun, untuk memastikannya, *user* diminta untuk meninjau data model logikal dan memastikan bahwa data model yang dibuat telah merepresentasikan kebutuhan data dari perusahaan.

6. Menggabungkan *logical* data model ke dalam global data model

   Langkah ini hanya diperlukan untuk mendesain *database* dengan pandangan *user* yang banyak yang diatur dengan menggunakan

pendekatan integrasi *view*. Data model logikal lokal merepresentasikan satu atau banyak tapi tidak semua pandangan *user* pada *database* sementara data model logikal global merepresentasikan semua pandangan *user* pada *database*.

7. Memeriksa untuk pertumbuhan masa datang

   Tujuan dari langkah ini adalah untuk menentukan apakah ada perubahan yang penting pada masa depat yang diperkirakan dan untuk menilai apakah data model logikal tersebut dapat menyesuaikan perubahan yang terjadi.

c. *Physical Database Design*

Physical Database Design adalah proses memproduksi sebuah deskripsi dari implementasi *database* pada penyimpanan kedua; mendeskripsikan relasi dasar, organisasi file, dan indeks yang digunakan untuk mencapai akses yang efisien kepada data, dan batasan integritas dan ukuran keamanan. *Physical database design* melalui 7 tahapan (Connoly, p441), yaitu:

1. Penerjemahan *logical* data model dalam *DBMS*

   Aktivitas pertama untuk desain *database* fisikal mencakup penerjemahan dari relasi pada data model logikal kepada sebuah bentuk yang dapat diimplementasikan pada target relasional *DBMS*. Bagian pertama pada proses ini memerlukan perbandingan antara informasi yang telah didapatkan selama desain database logikal dan dokumentasi pada kamus data dengan informasi yang didapatkan selama pengumpulan kebutuhan dan tahap analisis dan dokumen

spesifikasi sistem. Bagian kedua dari proses menggunakan informasi tersebut untuk memproduksi desain dari relasi dasar.

2. Perancangan organisasi file dan indeks

   Tujuan dari langkah ini adalah untuk menetapkan organisasi file yang optimal untuk menyimpan relasi dasar dan indeks yang dibutuhkan untuk mencapai performa yang diinginkan. Aktivitas yang dilakukan pada tahap ini adalah:

   a. Menganalisis transaksi
   b. Memilih organisasi file
   c. Memilih indeks
   d. Mengestimasi kebutuhan *disk space*

3. Perancangan *user view*

   Tujuan dari langkah ini adalah untuk mendesain pandangan *user* yang diidentifikasikan selama pengumpulan kebutuhan dan tahap analisis pada *database system development lifecycle.*

4. Perancangan mekanisme keamanan

   Tujuan dari langkah ini adalah untuk mendesain mekanisme keamanan untuk *database* seperti yang telah dispesifikasi oleh *user* selama tahap pengumpulan dan kebutuhan pada *database system development lifecycle.*

5. Pertimbangan pengenalan pengawasan *redundancy*

   Tujuan dari langkah ini adalah untuk menetapkan apakah *redundancy* yang terkontrol dengan aturan normalisasi akan meningkatkan peforma dari sistem.

6. Pemantauan dan pengaturan sistem operasional

   Tujuan dari langkah ini adalah untuk memantau sistem operasional dan meningkatkan performa dari sistem untuk membetulkan desain yang tidak cocok atau mencerminkan perubahan kebutuhan.

2.7.5 *DBMS Selection*

DBMS selection adalah pemilihan *DBMS* yang cocok untuk mendukung sistem *database* (Connoly, p295).

2.7.6 *Application Design*

Application Design adalah desain dari tampilan pengguna dan program aplikasi yang menggunakan dan memproses *database* (Connoly, p299).

2.7.7 *Prototyping*

Prototyping adalah membuat model yang bekerja pada sebuah sistem *database* (Connoly, p304).

2.7.8 *Implementation*

Implementation adalah perwujudan fisik dari *database* dan desain aplikasi (Connoly, p304).

2.7.9 *Data Conversion and Loading*

Data conversion and loading adalah memindahkan data yang ada ke *database* baru dan mengkonversikan aplikasi yang ada untuk dijalankan di *database* baru (Connoly, p305).

2.7.10 *Testing*

Testing adalah proses menjalankan sistem *database* dengan tujuan mencari kesalahan yang ada (Connoly, p305).

2.7.11 *Operational Maintenance*

Operational maintenance adalah proses dari pengawasan dan penjagaan sistem *database* mengikuti instalasi (Conolly, p306).

2.8 Operasi Perdagangan

2.8.1 Siklus Perdagangan

Menurut Weygandt, pada proses perdagangan, pendapatan bersih (*net income*) didapatkan dari hasil perhitungan antara pengeluaran dan penerimaan. Penjualan barang merupakan sumber utama dalam penerimaan. Pembelian barang merupakan salah satu aspek pengeluaran dalam proses perdagangan.

Umumnya siklus dalam proses perdagangan melalui tiga tahap yang selalu berulang yaitu:

1. Pembelian barang

   Kas yang ada digunakan untuk membeli barang-barang kebutuhan perdagangan berupa barang yang diperdagangkan dan inventori lain untuk menunjang proses perdagangan.

2. Penjualan barang

   Tenaga penjualan menjual barang barang yang diperdagangkan untuk memperoleh penerimaan kas dari penjualan tersebut.

3. Penerimaan kas

Kas yang diterima dari hasil penjualan dikalkulasikan dengan kas yang digunakan dalam proses pembelian barang untuk mendapatkan jumlah yang diterima secara bersih.

2.8.2 Pembelian

2.8.2.1 Pengertian Pembelian

Menurut Mulyadi (2001, p299) pembelian adalah suatu usaha yang dilakukan untuk pengadaan barang yang diperlukan oleh perusahaan. Secara umum definisi dari pembelian adalah merupakan usaha pengadaan barang atau jasa dengan tujuan yang akan digunakan sendiri, untuk kepentingan proses produksi ataupun untuk dijual kembali, baik dengan atau tanpa proses.

2.8.2.2 Jenis-Jenis Pembelian

Jenis-jenis dari pembelian antara lain :

1. Jenis pembelian berdasarkan pemasok :
    a. Lokal : Pemasok berasal dari dalam negeri.
    b. Impor : Pemasok berasal dari luar negeri.
2. Jenis pembelian berdasarkan transaksi :
    a. Tunai : Jenis transaksi dimana pembayarannya dilakukan secara langsung pada saat barang diterima.
    b. Kredit : Jenis transaksi dimana pembayaran tidak dilakukan secara langsung pada saat barang diterima, tetapi dilakukan selang beberapa waktu setelah barang diterima sesuai perjanjian kedua belah pihak.

2.8.2.3 Fungsi yang Terkait dengan Pembelian

Menurut Mulyadi (2001, p299) fungsi yang terkait dalam sistem pembelian adalah :

a. Fungsi gudang

Bertanggung jawab untuk mengajukan permintaan pembelian sesuai posisi persediaan yang ada di gudang dan untuk menyimpan barang yang telah diterima oleh fungsi penerimaan.

b. Fungsi barang

Bertanggung jawab untuk memperoleh informasi mengenai harga barang, menentukan pemasok yang dipilih dalam pengadaan barang dan mengeluarkan pesanan pembelian kepada pemasok yang dipilih.

c. Fungsi penerimaan

Bertanggung jawab untuk melakukan pemeriksaan terhadap jenis, mutu, dan kuantitas barang yang diterima dari pemasok guna menentukan dapat atau tidaknya barang tersebut diterima oleh perusahaan.

d. Fungsi akuntansi

Fungsi pencatat hutang bertanggung jawab untuk mencatat transaksi pembelian ke dalam register bukti kas keluar dan untuk menyelenggarakan arsip dokumen sumber yang berfungsi sebagai catatan hutang.

Fungsi pencatat persediaan berfungsi untuk mencatat harga pokok persediaan yang dibeli ke dalam kartu persediaan.

2.8.2.4 Pencatatan Pembelian Barang

Menurut Weygandt, pembelian barang umumnya dicatat pada saat barang yang dibeli diterima dari penjualnya. Setiap transaksi pembelian seharusnya disertai dengan dokumen yang menujukan bukti dan fakta pembelian. Setiap pembelian akan menambah persediaan barang, namun mengurangi kas perusahaan.

Pencatatan fakta-fakta dalam pembelian umumnya berupa *invoice* pembelian yang diberikan oleh penjual barang. *Invoice* pembelian yang berikan tergantung pada jenis pembelian, jika kredit maka *invoice* pembelian berupa salinan dari *invoice* pembelian yang asli. Untuk jenis pembelian tunai *invoice* yang diterima berupa *invoice* pembelian asli.

2.8.2.5 Pengembalian Pembelian Barang

Umumnya pengembalian barang dapat dilakukan apabila terjadi kerusakan pada barang-barang yang dibeli ataupun kesalahan dalam pemberian barang dari pihak penjual barang. Hal ini menyebabkan pengurangan persediaan barang dan penambahan kas.

2.8.3 Penjualan

2.8.3.1 Pengertian Penjualan

Menurut Mulyadi (2001, p202) kegiatan penjualan terdiri dari transaksi penjualan barang atau jasa, baik secara kredit atau tunai. Penjualan dapat disimpulkan sebagai suatu aktivitas perusahaan yang utama dalam memperoleh pendapatan, baik untuk perusahaan besar maupun perusahaan kecil. Perusahaan merupakan sasaran akhir dari kegiatan pemasaran, karena pada bagian ini ada penetapan harga, dagakan perundingan dan perjanjian serah terima barang, maupun perjanjian cara pembayaran yang disepakati oleh kedua belah pihak, sehingga tercapai suatu titik kepuasan.

2.8.3.2 Jenis-Jenis Penjualan

Menurut Mulyadi (2001, p202), transaksi penjualan dibedakan menjadi :

1. Penjualan kredit

   Penjualan kredit adalah penjualan yang dilakukan dengan cara memenuhi order dari pelanggan dengan mengirimkan barang atau menyerahkan jasa, untuk jangka waktu tertentu perusahaan memiliki piutang kepada pelanggannya.

2. Penjualan tunai

   Penjualan yang dilakukan perusahaan dengan cara mewajibkan pembeli dengan melakukan pembayaran barang terlebih dahulu sebelum barang diserahkan kepada pembeli.

2.8.3.3 Fungsi yang Terkait dengan Penjualan Tunai

Menurut Mulyadi (2001, p462) fungsi-fungsi yang terkait dalam sistem penerimaan kas dari penjualan tunai adalah :

a. Fungsi penjualan

Bertanggung jawab untuk menerima *order* dari pembeli, mengisi faktur penjualan tunai, dan menyerahkan faktur tersebut kepada pembeli untuk kepentingan pembayaran harga barang ke fungsi kas.

b. Fungsi kas

Bertanggung jawab sebagai penerima kas dari pembeli.

c. Fungsi gudang

Bertanggung jawab untuk menyiapkan barang yang dipesan oleh pembeli, serta menyertakan barang tersebut ke fungsi pengiriman.

d. Fungsi pengiriman

Bertanggung jawab untuk membungkus barang dan menyerahkan barang yang telah dibayar harganya kepada pembeli.

e. Fungsi akuntansi

Bertanggung jawab sebagai pencatat transaksi penjualan dan penerimaan kas dan pembuat laporan penjualan.

2.8.3.4 Pencatatan Penjualan Barang

Menurut Weygandt, pencatatan penjualan dilakukan pada setiap transaksi penjualan yang terjadi untuk memberikan fakta-

fakta telah terjadi penjualan. Secara umum penjualan terjadi saat barang barang berpindah dari penjual ke pembeli. Saat penjualan terjadi, pihak penjual mengeluarkan dokumen penjualan dalam bentuk *invoice* penjualan.

2.8.3.5 Pengembalian Penjualan

Menurut Weygandt, jika terjadi pengembalian barang karena kerusakan barang, maka harus dilakukan penyesuaian inventori atas barang tersebut dan biayanya. Hal ini akan menyebabkan *contra revenue account* atas penjualan. Pengembalian barang menyebabkan berkurangnya kas.

2.8.4 Persediaan

2.8.4.1 Pengertian Persediaan

Menurut Mulyadi (2001, p553), sistem penjualan bertujuan untuk mencatat mutasi tiap jenis persediaan yang disimpan di gudang. Dalam perusahaan manufaktur, persediaan terdiri dari persediaan produk jadi, produk dalam proses, bahan baku, bahan pelengkap, bahan habis pakai pabrik, dan suku cadang. Dalam perusahaan dagang, persediaan hanya terdiri dari satu golongan, yaitu persediaan barang dagangan, yang merupakan barang yang dibeli untuk tujuan dijual kembali.

2.8.4.2 Jenis-Jenis Persediaan

Menurut Render (2001, p314) terdapat 4 jenis persediaan :

1. Persediaan bahan mentah

   Bahan mentah telah dibeli, namun belum diproses.

2. Persediaan barang dalam proses

   Barang telah mengalami beberapa perubahan, tetapi belum selesai.

3. Persediaan *MRO* (*Maintenance Repair Operation*)

   Persediaan yang dikhususkan untuk perlengkapan pemeliharaan atau perbaikan atau operasi.

4. Persediaan barang jadi

   Barang sudah selesai dan menunggu untuk dikirimkan, barang jadi dimasukkan ke dalam persediaan karena permintaan konsumen untuk jangka waktu tertentu mungkin tidak diketahui.

2.8.4.3 Fungsi yang Terkait dengan Persediaan

Menurut Mulyadi (2001, p579) fungsi-fungsi yang terkait dalam sistem persediaan adalah :

a. Panitia penghitungan fisik persediaan

Berfungsi untuk melaksanakan penghitungan fisik persediaan dan menyerahkan hasil penghitungan tersebut kepada Bagian Kartu Persediaan untuk digunakan sebagai dasar penyesuaian terhadap catatan persediaan dalam kartu persediaan.

b. Fungsi akuntansi

Bertanggung jawab untuk mencantumkan harga pokok satuan persediaan yang dihitung ke dalam daftar hasil penghitungan fisik, mengalikan kuantitas dan harga pokok per satuan yang tercantum dalam daftar hasil penghitungan fisik, melakukan penyesuaian

terhadap kartu persediaan berdasarkan data hasil penghitungan fisik persediaan, membuat bukti memorial uituk mencatat penyesuaian data persediaan dalam jurnal umum berdasarkan hasil perhitungan fisik persediaan.

c. Fungsi gudang

Bertanggung jawab untuk melakukan penyesuaian data kuantitas persediaan yang dicatat dalam kartu gudang berdasarkan hasil penghitungan fisik persediaan.

2.9 Teori Pendukung

2.9.1 SQL Server 2000

SQL (Structured Query Language) Server 2000 merupakan generasi lanjutan dari SQL Server 7.0 yang dikeluarkan pada tahun 1999. SQL Server 2000 ini memiliki tambahan berupa dukungan Extensible Markup Language (XML), tambahan Online Analytical Processing (OLA), kemampuan data *mining*, mendukung Windows 2000, integrasi dengan direktori aktif Windows 2000, dan banyak lagi penambahan performa, kegunaan, dan program.

SQL Server 2000 menyediakan dukungan untuk menyimpan, menggunakan, dan memperbaharui dokumen XML yang merupakan kebutuhan penting karena XML telah menjadi bahasa yang menjadi pilihan banyak sistem bisnis dan sebuah komponen arsitektur dasar dari Microsoft .Net. SQL Server 2000 merupakan *server enterprise* .Net pertama yang tersedia untuk implementasi umum dan menawarkan penyimpanan data dan manajemen komponen pada layanan .Net.

2.9.2 Microsoft .Net

Menyediakan alat dan bahasa yang penting untuk membangun aplikasi menggunakan komponen arsitektur .Net. Arsitektur program .Net didukung oleh Visual Studio.Net termasuk VB.Net, Active Server Pages+, ActiveX Data Objects+, dan C# (termasuk C++). Kekurangan dari Visual Studio.Net adalah Visual InterDev. Microsoft Visual Studio sendiri terintegrasi dengan lingkungan pengembangan antar bahasa dengan integrasi dari aplikasi web melalui semua teknologi Microsoft, jadi peranan dari Visual InterDev sudah tidak signifikan untuk *web programming*.