BAB II

LANDASAN TEORI

2.1 Teori-Teori Dasar *Database*

2.1.1 Pengertian Sistem

Menurut Connoly dan Begg (2002, p270), " sistem adalah suatu cara untuk mengumpulkan, mengatur, mengendalikan, dan menyebarkan informasi ke seluruh organisasi ".

Menurut James A. 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 ".

2.1.2 Pengertian Data

Menurut James A. O'Brien (2003, p13), " data adalah fakta-fakta atau observasi yang mentah, biasanya mengenai kejadian atau transaksi bisnis ".

Menurut Donald R. Cooper (2002, p72), " data adalah sebagai fakta-fakta yang diberikan kepada peneliti dari lingkungan studinya ".

Menurut Whitten, Bentley, Dittman (2002, p475), " data adalah sumber yang mesti dikontrol dan ditangani ".

2.1.3 Pengertian Basis Data

Sebelum terdapat sistem basis data, digunakan sistem berbasis *file* yang disebut sebagai *File Based System*.

2.1.3.1 *File Based System* (Sistem Berbasis *File*)

Menurut Connoly dan Begg (2002, p7), " *File-Based System* adalah suatu kumpulan dari program-program aplikasi yang menyediakan kegunaan tertentu kepada *users* seperti pembuatan laporan-laporan ". Setiap program mendefinisikan dan mengelola data-datanya sendiri. Namun penggunaan *File Based System* ini memiliki keterbatasan seperti :

1. Terjadi pemisahan dan isolasi data.
2. Terjadi duplikasi data.
3. Terjadi ketergantungan data kepada fungsi.
4. Dapat menyebabkan bentuk *file* yang tidak cocok untuk suatu fungsi dengan fungsi lainnya (tidak kompatibel).
5. Sulit mengembangkan fungsi-fungsi program aplikasinya

Untuk mengatasi keterbatasan itu, digunakanlah pendekatan dengan menggunakan basis data dan DBMS (*Database Management System)*.

2.1.3.2 Basis Data

Menurut Connoly dan Begg (2002, p14), " Basis data didefinisikan sebagai sekumpulan data yang saling terhubung secara logikal, yang dirancang dalam rangka memenuhi kebutuhan informasi dalam suatu organisasi ".

Menurut James A. O'Brien (2003, p145), “ Basis data adalah sebuah kumpulan yang terintegrasi dari elemen data yang terhubung secara logikal ”. Elemen data mendeskripsikan entiti-entiti dan hubungan antara entiti-entiti.

Menurut Turban (2003, p16), “ Basis data merupakan kumpulan *file* atau *record* yang terorganisir yang menyimpan data beserta hubungan diantara data tersebut ”.

Menurut Gary W. Hansen, James W. Hansen (1996, p12), ” Basis data adalah kumpulan dari item-item data yang saling terhubung yang dapat diproses oleh satu atau lebih sistem aplikasi ”.

Tujuan utama pengelolaan data dalam basis data adalah agar kita dapat memperoleh data yang kita cari 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 pemakaian (*sharability*)

Dalam penggunaannya, basis data memiliki beberapa keuntungan yaitu:

1. Mengurangi kesalahan yang disebabkan oleh faktor manusia. Tugas mekanis lebih baik dilaksanakan oleh mesin.
2. Komputer dapat mengambil dan mengubah data lebih cepat dari manusia.
3. Akurat dan informasi terbaru selalu tersedia setiap saat.
4. Menghemat ruangan karena tidak perlu menyediakan ruangan penyimpanan kertas *file* yang sangat banyak.

2.1.2.3 Komponen Basis Data

Pada saat ini sistem basis data sudah bisa dikembangkan dengan baik pada mesin-mesin komputer mikro sampai dengan komputer *mainframe*. Tujuan dari sistem tersebut secara keseluruhan adalah untuk melakukan perawatan informasi dan menyediakannya kapan saja diperlukan oleh pengguna. Komponen-komponen penting pada sistem basis data yaitu:

a. Perangkat keras (*Hardware*)

Untuk manajemen basis data hanya dibutuhkan mesin standar, namun yang harus diperhatikan adalah kapasitas penyimpanan karena pada basis data akan membutuhkan kapasitas yang besar.

b. Perangkat lunak (*Software*)

Piranti lunak untuk sistem basis data disebut dengan DBMS (*Database Management System*). DBMS memungkinkan pengguna untuk membentuk *file* (*create*), penambahan data (*insert*), penghapusan (*delete*), dll.

c. Data

Data dalam basis data dapat merupakan data yang *single user* (hanya satu pengguna yang beroperasi terhadap basis data) atau *multi user*, dimana satu atau lebih *user* beroperasi secara bersama-sama ke dalam basis data. Sehingga data dalam basis data terutama untuk sistem yang besar, harus terintegrasi (*integrated*) dan dapat dipakai bersama (*shared*).

Pengertian terintegrasi adalah suatu basis data dapat dipandang sebagai kumpulan file-file yang saling berhubungan satu sama lainnya dan dimana sebagian atau seluruh redundansi yang dapat dihilangkan. Sedangkan pengertian penggunaan bersama (s*hared*) adalah setiap bagian data yang ada di dalam basis data dapat digunakan secara bersama-sama oleh lebih dari satu pengguna untuk penggunaan yang mungkin berbeda. Dengan pengertian bahwa setiap pengguna mungkin hanya berkepentingan pada subset data tertentu saja.

d. Prosedur

Prosedur merupakan instruksi dan aturan yang mengatur rancangan dan penggunaan basis data. *User* dari sistem dan stafnya mengatur dokumen prosedur tentang bagaimana caranya menjalankan sistem. Prosedur bisa terdiri dari intruksi tentang bagaimana caranya untuk *log on* kedalam DBMS, menggunakan fasilitas DBMS dan program aplikasinya, memulai dan mengakhiri menggunakan DBMS, membuat salinan (*back up*) dari basis data, mengatasi kerusakan perangkat keras dan perangkat lunak, mengganti struktur tabel dan meningkatkan kinerja

e. Pengguna (*User*)

Pengguna yang terlibat dalam komponen DBMS yaitu :

a. Administrasi Basis Data (DBA)

   Bertanggung jawab untuk mengatur manajemen sumber daya data yang meliputi perencanaan basis data, pengembangan dan pemeliharaan standarnya, aturan dan prosedur, dan rancangan basis data konseptual atau logikal.

b. Programmer

   Programmer adalah seorang atau sekelompok orang yang menjadi tenaga ahli komputer yang berfungsi untuk mengembangkan program-program aplikasi yang diperlukan dalam DBMS.

c. Pengguna akhir

   Yang termasuk dalam kategori pengguna akhir adalah pemilik sistem, para manajer, operator, dan sebagainya yang terlibat langsung dalam pengguna basis data.

2.1.4 *Database Management System* (DBMS)

Definisi *Database Management System* (DBMS) menurut Connoly dan Begg (2002, p16) adalah suatu sistem perangkat lunak yang memungkinkan *user* untuk mendefinisikan (*define*), membuat (*create*), memelihara (*maintain*) basis data, dan menyediakan kendali dalam mengakses basis data.

2.1.4.1 Komponen-komponen dalam lingkungan DBMS

Menurut Connolly (2002, p18) DBMS memiliki 5 komponen, yaitu:

a. Perangkat keras (*Hardware*)

   DBMS membutuhkan perangkat keras untuk menjalankannya.

   Contohnya : *single personal computer, single mainframe*

b. Perangkat lunak (*Software*)

   Karena seluruh kendali DBMS akan dilakukan oleh program aplikasi maka DBMS harus dapat dihubungkan ke program-program aplikasinya.

c. Data

   Data yang digunakan oleh suatu organisasi harus didesain sedemikian rupa sehingga mudah untuk digunakan.

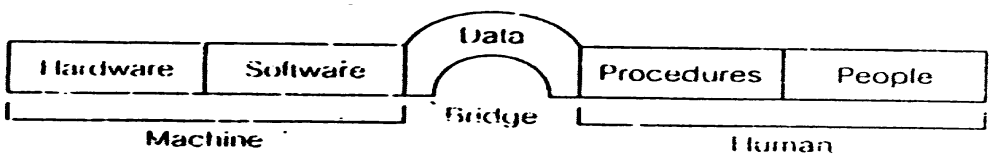
d. Prosedur (*procedures*)

   Perintah-perintah yang harus dilakukan untuk menjalankan DBMS ini.

e. Faktor manusia (*People*)

   Ini adalah masalah keterkaitan perilaku orang yang menggunakan sistem (*user*) dengan sistem DBMS yang sudah didesain sebelumnya.

   Manusia yang terlibat dengan sistem, termasuk dalamnya adalah : *database administrator,* perancang *database*, pengembang aplikasi dan pemakai akhir. Komponen-komponen dalam lingkungan DBMS dapat digambarkan sebagai berikut :



Gambar 2.1 Komponen-komponen dalam lingkungan DBMS
(Connoly dan Begg (2002, p18))

2.1.4.2 Fasilitas-fasilitas DBMS

Menurut Connoly dan Begg (2002, p16), Fasilitas-fasilitas dalam DBMS adalah sebagai berikut ini :

a. Mendefinisikan basis data, biasanya menggunakan suatu *Data Definition Language* (DDL) yang berguna untuk memberikan fasilitas kepada *user* untuk mendeklarasikan tipe data, struktur dan isi data yang disimpan ke dalam basis data tersebut.

b. Memperbolehkan *user* untuk menambah data, mengubah data, menghapus data, dan menemukan data. Biasanya dengan menggunakan suatu *Data Manipulation Language* (DML). Biasanya ada suatu fasilitas untuk melayani pengaksesan data yang disebut sebagai *Query Language*. Bahasa *query* yang paling diakui adalah *Structured Query Language* (SQL), yang secara *de facto* merupakan standar bagi DBMS.

c. Selain itu terdapat juga fungsi-fungsi untuk pengendalian basis data. Sebagai contoh :

   1. Sistem keamanan (*security system*)

      Untuk mencegah *user* yang tidak berwenang untuk mengakses file-file yang bukan haknya.

2. Sistem kontrol (*concurrency control system*)

   Untuk mengijinkan data dapat dipakai bersama-sama oleh *user-user* lainnya.

3. Sistem kontrol pengembalian (*recovery control system*)

   Untuk memperbaiki data-data jika sebelumnya terjadi kerusakan-kerusakan pada perangkat keras atau perangkat lunak.

4. Katalog yang dapat diakses *user* (*user accessible catalog*)

   Suatu catatan dari deskripsi data-data dalam *database*.

2.1.4.3 *Data Definition Language* (DDL)

Definisi dari *Data Definition Language* menurut Connoly dan Begg (2002, p40) adalah " suatu bahasa yang memperbolehkan *Database Administrator* (DBA) atau *user* untuk mendefinisikan entiti, atribut, dan *relationship* yang dibutuhkan oleh suatu aplikasi serta menambahkan fungsi-fungsi khusus tertentu untuk mempermudah atau meningkatkan keamanan datanya ".

2.1.4.4 *Data Manipulation Langua*ge (DML)

Data Manipulation Language menurut Connoly dan Begg (2002, p41) adalah " suatu bahasa yang melakukan operasi-operasi standar pada data yang ada di dalam basis data ". Pengoperasian data yang akan dimanipulasi biasanya meliputi :

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

2.1.4.5 Keuntungan dan kekurangan dari DBMS

Keuntungan dari penggunaan DBMS antara lain :

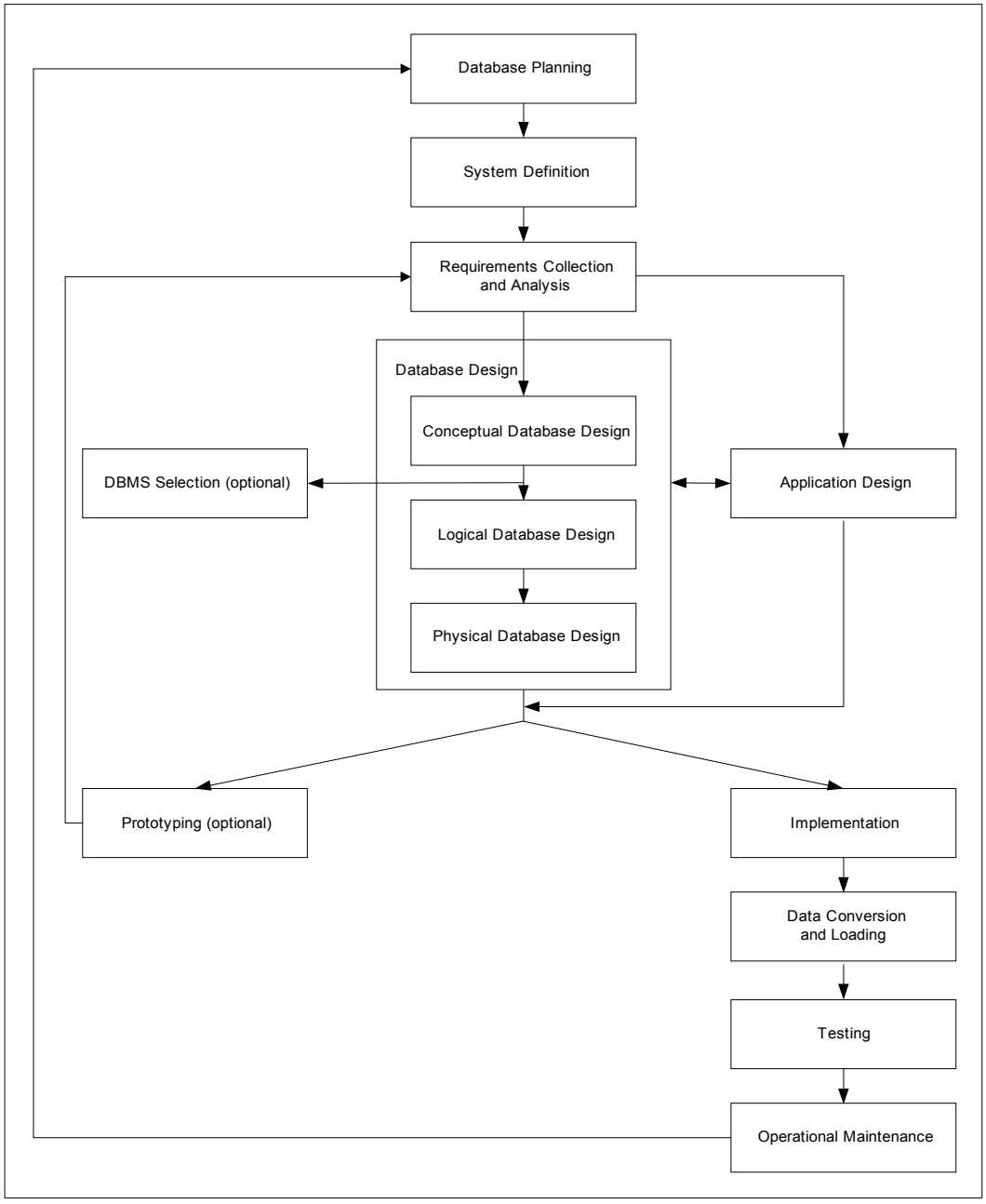
1. Dapat menghindari data-data sama yang disimpan berulang kali (*control of data redundancy*).
2. Data menjadi lebih konsisten.
3. Bisa didapat informasi yang lebih banyak dari kumpulan data-data yang sama, dengan permasalahan tertentu.
4. Data-data yang sama dapat digunakan bersama-sama oleh *user-user* yang berbeda-beda (*shared data*).
5. Meningkatkan keterkaitan antar data.
6. Meningkatkan keamanan.
7. Membuat suatu standar.
8. Meningkatkan efisiensi kebutuhan.
9. Kebutuhan-kebutuhan yang berbeda-beda dapat dipenuhi dengan mudah.
10. Meningkatkan kemampuan akses data dan kecepatan prosesnya.
11. Meningkatkan produktivitas.
12. Memudahkan perawatan data.
13. Meningkatkan keamanan pada data yang dipakai bersama-sama.
14. Mempermudah fungsi *back-up* dan *recovery*.

Selain keuntungan juga terdapat kekurangan dari DBMS dibanding *File Based System* :

1. Proses didalamnya lebih rumit.
2. Mempunyai kapasitas yang lebih besar jika dibandingkan oleh masing-masing *file* pada *file based system*.
3. Harganya lebih mahal.
4. Membutuhkan biaya tambahan untuk perangkat keras jika DBMS yang digunakan menuntut penggunaan jenis perangkat keras tertentu.
5. Membutuhkan proses-proses tambahan jika ingin mengubah format-format data dari suatu jenis DBMS ke DBMS jenis lain.
6. Kehandalan (kecepatan proses), jika suatu DBMS didesain untuk penggunaan yang sangat umum maka kecepatan prosesnya menjadi lambat.

2.1.5 Database Application Lifecycle

Menurut Connoly dan Begg (2002,p272), suatu basis data dianalisis dan dirancang berdasarkan siklus hidup seperti tergambar berikut:



Gambar 2.2 Siklus Hidup Aplikasi Basis Data

(Connoly dan Begg (2002, p272))

2.1.5.1 *Database Planning*

Database planning adalah aktivitas manajemen yang memungkinkan tingkatan aplikasi basis data untuk lebih dipahami seefisien dan seefektif mungkin. *Database planning* harus terintegrasi dengan IS (*Information System*) strategi secara keseluruhan dalam organisasi. Ada 2 isu utama yang terlibat dalam memformulasikan sebuah IS strategi, yaitu:

1. Mengidentifikasikan rencana perusahaan dan tujuan dari IS yang dibutuhkan. Tujuannya untuk mengevaluasi terhadap IS yang ada untuk menentukan kekuatan dan kelemahan.
2. Penafsiran kesempatan IT yang mungkin dapat menghasilkan keuntungan yang kompetitif.

Sebuah langkah awal yang penting dalam *database planning* adalah menentukan dengan jelas *mission statement* untuk proyek basis data. Selain itu juga *database planning* harus meliputi pengembangan standar bagaimana data akan dikumpulkan, bagaimana data dispesifikasikan, dokumentasi penting apa yang dibutuhkan, dan juga bagaimana desain dan implementasi harus diproses.

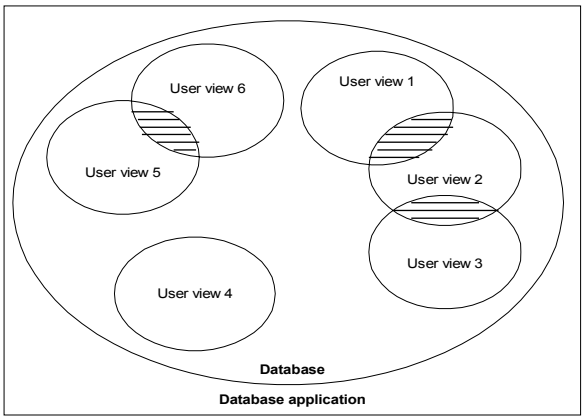
2.1.5.2 *System Definition*

Sistem definisi menggambarkan ruang lingkup aplikasi basis data dan *user views*. Sebelum diadakan percobaan desain aplikasi basis data, kita perlu mengidentifikasi batasan-batasan sistem (*the boundaries of the system*) yang sudah kita telusuri dan bagaimana sistem tersebut bertatap muka dengan bagian-bagian lain dalam organisasi sistem informasi.

User Views

User views didefinisikan sebagai kebutuhan aplikasi basis data dari tugas-tugas yang ada (seperti manajer atau *supervisor*) atau area aplikasi perusahaan (seperti marketing, personalia, *stock control*).

Sebuah aplikasi data mungkin memiliki satu atau lebih *user views*. Untuk lebih jelasnya dapat dilihat dari diagram dibawah ini:



Gambar 2.3 Aplikasi basis data dengan *user views*

Diagram diatas menjelaskan tentang aplikasi basis data dengan *multiple user view*s: *user views* (1,2,3) dan (5,6) memiliki kebutuhan untuk saling melengkapi (area yang di arsir), sedangkan *user views* 4 memiliki kebutuhan sendiri yang jelas.

2.1.5.3 *Requirement Collection* dan *Analysis*

Requirement collection dan *analysis* adalah proses untuk mengumpulkan dan menganalisa informasi tentang bagian dari suatu organisasi yang didukung oleh aplikasi basis data dan menggunakan informasi tersebut. Untuk mengidentifikasi kebutuhan *user* untuk system yang baru.

Ada beberapa macam *fact-finding techniques:*

a. *Examining documentation*

b. *Interviewing*

c. *Observing the enterprise in operation*

d. *Research*

e. *Questionnaire*

Informasi yang didapat menggunakan *fact-finding technique* dianalisa untuk mengidentifikasi kebutuhan. Ada macam-macam cara untuk membuat *requirement specification* lebih terstruktur:

a. *Structur Analysis dan Design* (SAD)

b. *Data Flow Diagram* (DFD)

c. *Hierarchical Input Process Output* (HIPO) *Charts*.

Salah satu aktifitas yang penting dalam tahap ini adalah dalam menangani lebih dari 1 *user view* dalam aplikasi basis data. Ada 3 pendekatan utama dalam menangani kebutuhan dari aplikasi basis data dengan *multiple user view*, yaitu:

1. *Centralized Approach*

   Adalah kebutuhan dari tiap *user view* disatukan menjadi satu set kumpulan kebutuhan dari aplikasi basis data.

2. *View Integration Approach*

   Adalah kebutuhan dari tiap *user view* digunakan untuk membangun model data yang terpisah yang menggambarkan *user view* tersebut hasil dari data

model tersebut disatukan nanti pada saat tahap menggunakan local data model *database design*.

3. Kombinasi keduanya.

2.1.5.4 *Database Design*

Database design adalah proses menciptakan desain untuk basis data yang akan mendukung operasi dan tujuan perusahaan.

Menurut C.J Date (2000,p10), ”*A Database is Collection of Persistent data is used by the application system of some given enterprise*” dimana artinya database merupakan kumpulan dari data yang hampir tidak mengalami perubahan dan digunakan oleh aplikasi sistem pada beberapa perusahaan.

Adapun tujuan utama dari konsep basis data adalah meminimumkan pengulangan dan mencapai independensi data. Independensi data adalah kemampuan untuk membuat perubahan dalam struktur data tanpa membuat perubahan pada program yang memproses data. Independensi data dicapai dengan menempatkan spesifikasi dalam tabel dan kamus yang terpisah secara fisik dari program.

2.1.4.5.1 *Conceptual Database Design*

Menurut Connoly dan Begg (2002, p.419) “ perancangan basis data konseptual adalah proses membangun sebuah model informasi yang digunakan dalam sebuah perusahaan, bebas dari segala pertimbangan fisik “. Pada tahap ini melibatkan pembuatan dari konseptual data model dari perusahaan. Data model ini dibuat dengan menggunakan informasi yang di

dokumentasikan dari spesifikasi kebutuhan *user*. Rancangan basis data konseptual ini bebas atau tidak terikat dengan detai-detail seperti target perangkat lunak DBMS, program aplikasi, bahasa pemrograman, *platform* perangkat keras atau dari perangkat fisik lainnya. Melalui tahapan rancangan basis data konseptual ini, model diuji dan divalidasi agar sesuai dengan kebutuhan *user*. langkah-langkahnya sebagai berikut:

1. Identifikasi tipe *entity*

   Menurut Connoly dan Begg (2002,p331) " tipe *entity* adalah sekumpulan objek yang diindentifikasi oleh sebuah perusahaan atau perorangan yang mempunyai sifat-sifat yang sama dan mempunyai keberadaan yang independen ". *Entitas occurance* adalah sebuah objek unik yang teridentifikasi dalam *entity type* (Connoly dan Begg,2002,p333).

   Menurut Connolly, entiti dapat diklasifikasikan sebagai berikut:

   a. *Strong / Parent / Owner / Dominant entity type*

      Adalah sebuah entitas yang tidak tergantung pada entitas lain karena entitas tersebut diidentifikasikan dengan menggunakan *primary key.*

   b. *Weak / Child / Subordinate / Dependent entity type*

      Adalah sebuah entitas yang bergantung pada entitas lain karena entitas tersebut tidak diidentifikasikan dengan menggunakan *primary key*.

2. Identifikasi tipe *relationship*

   Menurut Connoly dan Begg (2002, p334) " tipe *relationship* adalah sekumpulan hubungan antara satu atau lebih entitas ". *Relationship occurance*

adalah sebuah hubungan unik antara satu atau lebih entitas yang teridentifikasi dalam *relationship type*. Relasi mempunyai batasan (*constraint*) utama yang harus merefleksikan larangan dalam hubungan seperti di 'dunia nyata' yang disebut *multiplicity*.

Menurut Connolly dan Begg, *multiplicity* sesungguhnya terdiri dari dua jenis, yaitu

a. Cardinality

Adalah penggambaran jumlah maksimum dari kemungkinan batasan sebuah entitas yang ikut berpartisipasi dalam sebuah relasi.

Multiplicity dengan jenis *cardinality* dari *binary relationship* dapat dibagi menjadi :

1. *One to One* (1:1) *relationship*
2. *One to Many* (1:*) *relationship*
3. *Many to Many* (*:*) *relationship*

b. Participation

Adalah penggambaran yang menentukan apakah semua atau hanya beberapa *entity occurance* (anggota entitas) yang ikut berpartisipasi dalam sebuah hubungan.

3. Identifikasi dan asosiasikan atribut dengan *entity-entity* dan *relationship* yang telah kita pilih untuk digambarkan dalam *database*. Atribut adalah sebuah *property entity* atau sebuah tipe *relationship*.

Attribute adalah sifat dari sebuah *entity* atau sebuah *relationship type*. *Attribute* menyimpan nilai dari setiap *entity occurrence* dan mewakili bagian utama dari data yang disimpan dalam basis data.

Simple attribute adalah suatu *attribute* yang terdiri atas komponen tunggal dengan keberadaan yang tidak terikat. Atribut ini tidak dapat dibagi lagi menjadi komponen yang lebih kecil.

Composite attribute adalah *attribute* yang terdiri atas banyak komponen, tiap-tiap komponen dengan keberadaan yang tidak terikat. *Attribute* tertentu dapat dibagi lagi menjadi komponen yang lebih kecil dengan keberadaan masing-masing yang tidak terikat.

Singlevalued a*ttribute* adalah *attribute* yang menampung nilai tunggal untuk tiap-tiap kejadian dari suatu *entity type*. Sebagian besar *attribute* adalah bernilai tunggal.

Multivalued a*ttribute* adalah *attribute* yang menampung banyak nilai untuk setiap kejadian dari suatu *entity type*.

Derived attribute adalah *attribute* yang menggantikan sebuah nilai yang diturunkan dari nilai sebuah *attribute* yang berhubungan, tidak perlu pada jenis *entity* yang sama.

4. Menentukan atribut domain

Domain Attribute adalah satuan nilai-nilai untuk satu atau beberapa *attribute*. Setiap *attribute* yang dihubungkan dengan sejumlah nilai disebut *domain*. *Domain* mendefinisikan nilai-nilai yang dimiliki sebuah *attribute* dan sama dengan konsep *domain* pada model relasional.

5. Menentukan kandidat dan atribut-atribut *primary key*

   Menentukan kandidat *key* untuk setiap tipe *entity* dan jika ada lebih dari satu kandidat *key* untuk memilih satu untuk dijadikan *primary key*.

6. Pertimbangan menggunakan konsep *modeling* yang menarik

   Untuk mempertimbangkan penggunaan konsep *enchaned modeling,* seperti spesialisasi, generalisasi, agregasi, dan komposisi. Jika kita memilih pendekatan generalisasi, kita mencoba untuk menyoroti perbedaan antara *entity-entity* dengan menjelaskan satu atau lebih *subclasses* dari sebuah *entity superclass*.

   Subclasses adalah sebuah *sub grouping* yang jelas dari kejadian dari sebuah tipe *entity* yang perlu digambarkan dalam sebuah model data.

   Superclass adalah sebuah tipe *entity* yang termasuk satu atau lebih *sub grouping* yang jelas dari kejadian-kejadiannya yang perlu direpresentasikan dalam sebuah model data.

7. Periksa model untuk *redundancy*

   Pada langkah ini kita memeriksa model data konseptual lokal dengan objektif spesifik untuk mengidentifikasi apakah ada terjadi *redundancy* dan menghapus yang sudah ada.

   Dua aktivitas dalam langkah ini antara lain:

   1. Memeriksa kembali *one-to-one* (1:1) *relationship*

      Dalam mengidentifikasikan *entity-entity* kita mungkin telah mengidentifikasikan dua *entity* yang merepresentasikan objek yang sama di perusahaan.

- Menghapus *relationship* yang *redundancy*

  Sebuah *relationship* adalah *redundant* bila informasi yang sama dapat diperoleh melalui *relationship* lain. Kita mencoba untuk mengembangkan sebuah model data minimal dan *relationship* yang *redundant* tidak diperlukan data tersebut harus di hapus.

8. Validasikan konseptual model lokal yang bertentangan dengan transaksi *user*

   Untuk meyakinkan bahwa model konseptual lokal mendukung transaksi yang dibutuhkan oleh tampilan. Kita sekarang memiliki model data konseptual lokal yang merepresentasikan tampilan spesifik dari perusahaan. Menggunakan model ini kita mencoba untuk menampilkan operasi-operasi secara manual. Kita memeriksa 2 pendekatan yang mungkin untuk meyakinkan, yaitu:

   1) Model data konseptual lokal mendukung transaksi yang diperlukan untuk menggambarkan transaksi.
   2) Menggunakan jalur transaksi (*Transaction Pathway*).

9. Mengulang model data konseptual lokal dengan *user*.

   Untuk me*review* model data konseptual lokal dengan *user* untuk meyakinkan bahwa model adalah representasi yang nyata dari tampilan. Sebelum melengkapi *step* satu, kita me*review* model data konseptual lokal *user*. Model data konseptual meliputi diagram ER dan dokumentasi yang mendukung yang menggambarkan model data.

2.1.5.4.2 Logical Database Design

Menurut Connoly dan Begg (2002, p441) " perancangan basis data logikal adalah proses membangun suatu model informasi yang digunakan dalam suatu perusahaan berdasarkan suatu model data khusus, tapi tidak tergantung kepada suatu DBMS tertentu dengan pertimbangan fisik lain ". Konseptual data model yang dihasilkan dari tahap sebelumnya diubah ke dalam *logical data model*. *Logical data model* dibuat berdasarkan target data model untuk basis data (sebagai contoh, *relational data model*).

Melalui tahap rancangan basis data logikal ini, model dites dan divalidasikan sesuai dengan kebutuhan *user*. Teknik normalisasi digunakan untuk melakukan tes terhadap kebenaran logikal data model. Normalisasi memastikan bahwa tabel yang dihasilkan dari data model tidak menampilkan redudansi data yang akan mengakibatkan kerancuan atau keanehan (*anomaly*) *update* pada saat diimplementasikan.

Menurut Connoly dan Begg (2002, p441) langkah-langkah dalam membuat rancangan basis data logikal adalah :

1. Membangun dan memvalidasikan model data logikal untuk setiap *view*
   a. Menghilangkan fitur-fitur yang tidak sesuai dengan model relasional.

      Tujuan dilakukannya langkah ini untuk menyaring model data konseptual lokal guna untuk menghilangkan fitur-fitur yang tidak sesuai dengan model relasional, dengan cara sebagai berikut :

      - Hilangkan tipe *relationship binary many-to-many* (* : *)

- Hilangkan tipe *relationship recursive many-to-many* (* : *)
- Hilangkan tipe *relationship* yang kompleks
- Hilangkan *attributes multivalued*

b. Membuat *relations* untuk model data logikal lokal

Tujuan dilakukannya langkah ini untuk membuat *relations* dalam model data logikal lokal guna untuk menunjukkan *entities*, *relationships*, dan *attributes* yang telah diidentifikasi.

c. Memvalidasikan *relations* menggunakan normalisasi

Tujuan dilakukannya langkah ini untuk memvalidasikan *relations* dalam model data logikal lokal dengan menggunakan teknik normalisasi guna memeriksa pengelompokan *attributes* dalam setiap *relation* yang telah dibuat dalam langkah 2.

Proses normalisasi melibatkan beberapa langkah, yaitu :

1. *First Normal Form* (1NF), menghilangkan *repeating group*
2. *Second Normal Form* (2NF), menghilangkan *partial dependencies* pada *primary key*
3. *Third Normal Form* (3NF), menghilangkan *transitive dependencies* pada *primary key*
4. *Boyce-Codd Normal Form* (BCNF), menghilangkan anomali-anomali yang masih tersisa dalam *functional dependencies*

d. Memvalidasikan *relations* pada transaksi-transaksi *user*

Tujuan dilakukannya langkah ini untuk memastikan bahwa *relations* dalam model data logikal lokal mendukung transaksi-transaksi yang diperlukan dalam penggambaran (*view*).

e. Mendefinisikan *integrity constraints*

Tujuan dilakukannya langkah ini untuk mendefinisikan *constraints* integritas yang ada dalam gambaran (*view*). Dalam melakukan langkah ini, kita juga perlu mempertimbangkan 5 tipe *constraints* integritas, yaitu:

- *Required data* (data yang diperlukan)
- *Attribute domain constraints* (pembatas-pembatas domain atribut)
- *Entity integrity*
- *Referential integrity*
- *Enterprise contraints*

f. Meninjau ulang model data logikal lokal dengan *user*

Tujuan dilakukannya langkah ini untuk memastikan model data logikal lokal dan dokumentasi pendukung menggambarkan model yang merupakan representasi nyata dari gambaran (*view*) tersebut.

2. Buat dan validasikan model data logikal global

g. Menggabungkan model-model data logikal lokal ke dalam model data global

Tujuan dilakukannya langkah ini untuk menggabungkan model-model data logikal lokal individual ke dalam suatu model data logikal global tunggal perusahaan.

Hal-hal yang perlu dilakukan :

a. Meninjau ulang nama dan isi dari tiap *entity /* relasi dan *candidate key*
b. Meninjau ulang nama dan isi dari *relationships foreign keys*
c. Menggabungkan *entities* / relasi-relasi dari model data lokal
d. Memasukkan (tanpa menggabungkan) *entities* / *relations* yang unik ke dalam setiap model data lokal
e. Menggabungkan *relationships* / *foreign keys* dari model data lokal
f. Memasukkan (tanpa menggabungkan) *relationships* / *foreign keys* yang unik ke dalam setiap model data lokal
g. Mengecek apakah ada *entity* / relasi dan *relationship* / *foreign ke*y yang tertinggal
h. Mengecek *foreign keys*
i. Mengecek *integrity constraints*
j. Menggambarkan diagram ER / relasi global
k. Meng-*update* dokumentasi

h. Memvalidasikan model data logikal global

Tujuan dilakukannya langkah ini adalah untuk memvalidasikan *relations* yang dibuat dari model data logikal lokal menggunakan

teknik normalisasi dan untuk memastikan bahwa *relations* tersebut mendukung transaksi-transaksi yang diperlukan, jika diperlukan.

i. Mengecek pertumbuhan masa depan

Tujuan dilakukannya langkah ini adalah untuk menentukan apakah ada perubahan signifikan yang mungkin terjadi di masa depan dan untuk menaksir apakah model data logikal lokal dapat menolong segala perubahan tersebut.

j. Meninjau ulang model data logikal global dengan *users*

Tujuan dilakukannya langkah ini adalah untuk memastikan model data logikal global merupakan representasi nyata perusahaan.

2.1.5.4.3 Physical Database Design

Adalah proses penjabaran dan penerapan dari basis data pada tempat penyimpanan kedua. Ini menjelaskan tentang pembuatan basis data, organisasi *file,* dan *index* yang digunakan untuk mencapai akses efisien data dan tindakan pengamanan. *Physical database design* merupakan tahap ketiga dan terakhir dari proses perancangan basis data. Dimana perancang memutuskan bagaimana basis data tersebut diimplementasikan secara garis besar, tujuan utama dari *physical database design* adalah untuk mendeskripsikan bagaimana perancang bermaksud untuk mengimplementasikan secara fisik dari *logical database design*. Akan tetapi dalam pengembangan *physical database design* hal pertama yang harus ditegaskan adalah sasaran dari sistem basis data. Oleh karena itu, *physical database design* disesuaikan untuk sebuah sistem DBMS khusus.

Aktivitas dari *physical database design* adalah menterjemahkan model data logikal global ke dalam sasaran DBMS, dimana dengan membuat *base relation* dengan menggunakan *Database Design Language* (DBDL) yang tujuannya untuk memutuskan bagaimana representasi relasi utama yang diidentifikasi dalam model data logikal global ke dalam DBMS, merancang *constraint* perusahaan untuk kegiatan transaksi dan menentukan kebutuhan sumber daya sistem dan organisasi *file*.

2.1.5.5 Implementasi

Pada tahap ini akan dilakukan realisasi fisik dari basis data dan rancangan aplikasi. Implementasi basis data dicapai dengan menggunakan *Data Definition Language* (DDL) dari DBMS yang sudah dipilih atau *Graphical User Interface* (GUI). Aplikasi program diimplementasikan menggunakan *third or forth generation language* (3GL atau 4GL). Bagian dari aplikasi program dan transaksi basis data, yang diimplementasikan menggunakan *Data Manipulation Language* (DML), kemungkinan sudah ada dalam *host* bahasa pemrograman seperti Visual Basic, Delphi, C, C++, Java, COBOL, Fortran, Ada atau Pascal.

2.1.5.6 Data Konversi dan *Loading*

Pada tahap ini akan dilakukan transfer data yang sudah ada sebelumnya ke dalam basis data yang baru dan mengkonversi aplikasi lama untuk dijalankan dalam basis data yang baru. Tahap ini hanya dilakukan ketika sistem basis data yang lama akan digantikan dengan yang baru.

2.1.5.7 *Testing*

Pada tahap ini dilakukan proses untuk menjalankan aplikasi program dengan tujuan untuk mencari *error* yang mungkin terjadi. *User* yang telah menggunakan aplikasi basis data yang baru harus terlibat dalam proses *testing* ini. Situasi yang ideal untuk melakukan *testing* sistem ialah dengan melakukan tes basis data di sistem *hardware* yang berbeda. Setelah *testing* selesai dilakukan, sistem aplikasi sudah siap untuk digunakan.

2.1.5.8 *Operational Maintenance*

Pada tahap ini akan dilakukan proses memonitor dan memelihara sistem baru yang telah di-*install*. Aktivitas yang dilakukan dalam tahap ini berupa :

- Mengawasi kinerja dari sistem
- Memelihara dan melakukan *upgrade* aplikasi basis data ketika diperlukan.

2.1.6 *Data Flow Diagram* (DFD)

Data Flow Diagram (DFD) merupakan salah satu alat bantu yang paling penting bagi seorang analis sistem. Penggunaan DFD sebagai alat bantu dipopulerkan oleh Tom De Marco (1978) dan Gane & Sarson (1979) dengan menggunakan pendekatan metode analisis sistem terstruktur.

DFD menggambarkan aliran ataupun arus data dalam sebuah sistem, yang merupakan masukan maupun keluaran yang diperlukan dari sistem yang sedang dirancang. Dalam DFD terdapat beberapa tingkatan yaitu :

1. Diagram Konteks

   Diagram konteks merupakan level tertinggi dari DFD yang menggambarkan keseluruhan *input* ke atau *output* dari sistem, dan hanya terdiri dari satu proses serta *external entity* yang terlibat. Diagram konteks memberikan gambaran keseluruhan sistem.

2. Diagram Nol

   Diagram nol menggambarkan proses-proses penting yang terdapat pada suatu sistem. Pada diagram nol diperlihatkan *data store* yang digunakan. Setiap proses yang digambarkan diberi nomor yang terdiri dari satu digit. Proses yang tidak dirinci lagi pada level selanjutnya (*functional primitive*) diberi tambahan tanda '*' pada akhir nomor proses.

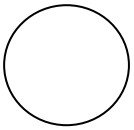
3. Diagram Rinci

   Diagram rinci merupakan rincian dari diagram nol atau diagram level di atasnya. Pada diagram rinci, nomor proses menunjukkan tingkat rincinya.

DFD terdiri dari sejumlah simbol-simbol yang mewakili komponen-komponen sistem. Komponen-komponen tersebut adalah proses sistem, data yang digunakan dalam proses, *eksternal* entiti yang berinteraksi dengan sistem, dan aliran informasi dalam sistem. Secara umum simbol-simbol tersebut dapat digambarkan sebagai berikut :

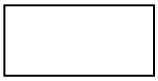
1. Proses

Menggambarkan apa yang dilakukan oleh sistem dan berfungsi mentransformasikan satu atau beberapa data masukan menjadi satu atau beberapa keluaran sesuai dengan spesifikasi yang diinginkan.



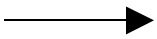
2. *Eksternal Entity*

   Merupakan entitas yang berada di luar sistem yang berfungsi untuk memberikan data kepada sistem (*source*) atau yang menerima informasi dari sistem.



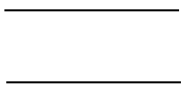
3. Aliran Data

   Menggambarkan aliran data dari suatu entiti ke entiti lainnya. Digambarkan dengan menggunakan garis lurus atau melengkung dengan anak panah yang menggambarkan arah aliran data.



4. *Data Store*

   Menggambarkan tempat penyimpanan atau penampungan data. *Data store* dapat berupa suatu *file* atau basis data di sistem komputer, arsip atau catatan manual, tabel acuan manual, agenda / buku, dan suatu kotak tempat data di meja seseorang.

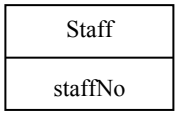


2.1.7 ***Entity Relationship Diagram*** **(ERD)**

Diagram yang digunakan untuk menggambarkan hubungan antara entitas dalam suatu sistem. Komponen-komponen dalam ERD yaitu :

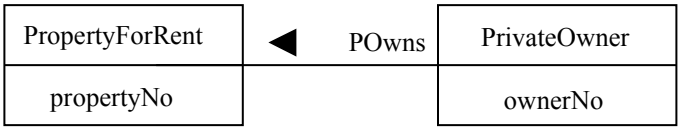
1. *Entity* adalah objek yang dapat diidentifikasikan dalam lingkungan pemakai, sesuatu yang penting bagi pemakai dalam konteks sistem yang akan dibuat. *Entity* disimbolkan dengan persegi panjang

   Contoh :

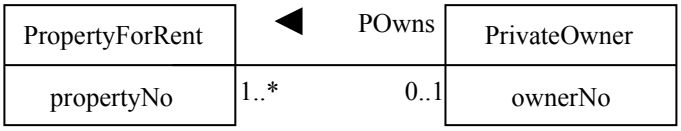
2. *Relationship* adalah asosiasi antara satu atau beberapa entitas. *Relationship* disimbolkan garis.

   Contoh :

3. *Cardinality* merupakan hubungan antara *entity* satu dengan *entity* lainnya.

   Contoh :

   Jenis-jenis *cardinality* yaitu :

   a. *One to One*

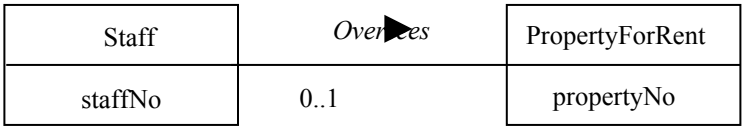
      Contoh : Hubungan antara *Staff* dan *Branch,* yang memiliki asosiasi *manages* dimana tiap *relationship* menggambarkan hubungan antara satu *Staff* dan satu *branch.*

Dari diagram di atas maka arti dari 1..1 adalah tiap cabang dikelola oleh seorang *staff*. Sedangkan 0..1 adalah tiap *staff* mengelola 1 cabang atau tidak mengelola satu pun cabang.

b. *One to Many / Many to One*

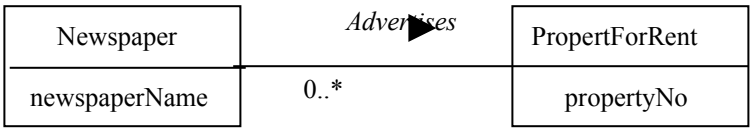
Contoh : hubungan antara *Staff* dan *PropertyForRent* yang memiliki asosiasi *Oversees*



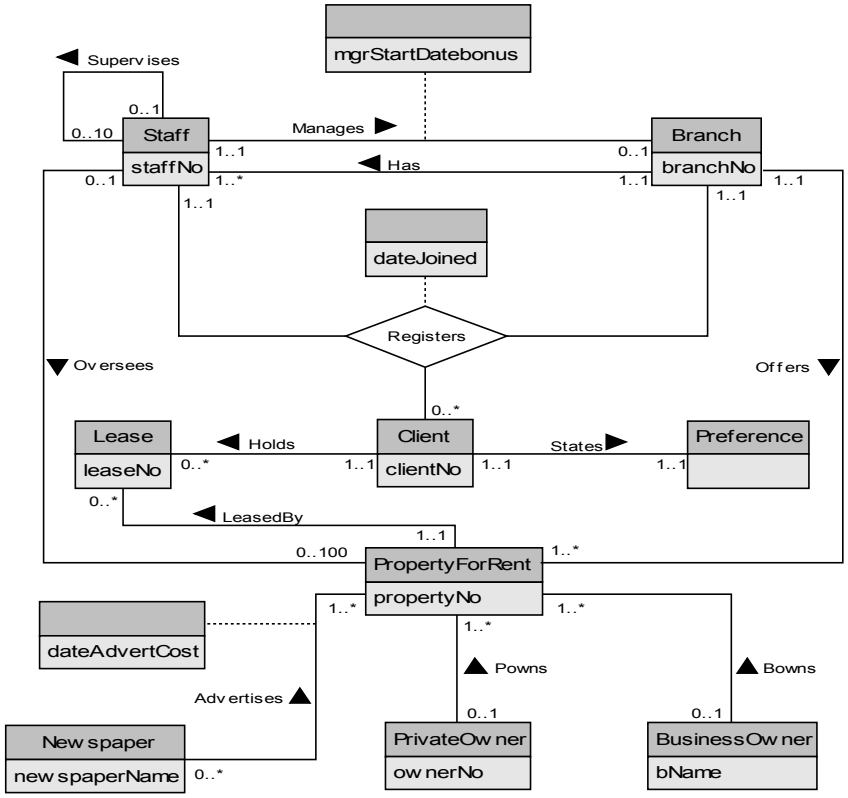
Dari diagram di atas maka arti dari 0..1 adalah tiap *PropertyForRent* diawasi oleh satu *Staff* atau sama sekali tidak diawasi oleh satu orang *Staff* pun. Sedangkan arti dari 0..* adalah tiap *Staff* mengawasi paling sedikit nol dan dapat lebih dari satu *PropertyForRent.*

c. *Many to Many*

Contoh : hubungan antara *Newspaper* dan *PropertyForRent*yang memiliki asosiasi *Advertises*



Dari diagram di atas maka arti dari 0..* adalah tiap *PropertyForRent* diiklankan dalam paling sedikit 0 atau lebih dari satu Koran. Sedangkan arti dari 1..* adalah tiap koran mengiklankan paling sedikit 1 atau lebih dari satu *PropertyForRent.*



Gambar 2.4 ***Entity Relationship Diagram***

(Connolly dan Begg, 2002, p332)

2.1.8 Normalisasi

Suatu desain basis data harus memenuhi kondisi untuk tidak mengandung anomali, yaitu suatu kejanggalan dari suatu penempatan atribut dari suatu obyek data. Untuk membedakan satu *record* dengan yang lainnya maka perlu dipilih atribut atau kombinasi atribut sebagai *primary key*. Syarat *primary key* adalah harus unik, jumlah kombinasi atribut minimum, dan tidak boleh mengandung nilai kosong (*null*).

Tujuan dari normalisasi antara lain :

a. Menghilangkan kumpulan relasi dari *insertion, update,* dan *delete dependency* yang tidak diharapkan.
b. Mengurangi kebutuhan restrukturisasi kumpulan relasi.
c. Membuat model relasional lebih informatif.

Langkah-langkah normalisasi :

a. Normalisasi Pertama (1NF)

   Suatu data dikatakan *un-normalized*, jika didalamnya mengandung kelompok berulang (*repeating group*), sehingga untuk membentuk normalisasi pertama (1NF) *repeating group* harus dihilangkan. Untuk menjadi 1stNF maka *group* yang berulang dihilangkan dengan mengisi pada bagian yang kosong dengan yang seharusnya pada suatu bentuk *record*.

b. Normalisasi Kedua (2NF)

   Dapat dihasilkan dengan melihat apakah ada atribut bukan *primary key* yang merupakan fungsi dari sebagian *primary key* (*partial dependence*). Dalam normalisasi kedua (2NF) setiap atribut yang tergantung parsial ini harus dipisahkan dengan mengikut sertakan determinannya. Bentuk normal diperoleh bila setiap atribut bukan *primary key* dari suatu tabel sepenuhnya merupakan fungsi (*fungsional dependence*) dari *primary key* tersebut.

c. Normalisasi Ketiga (3NF)

   Pengujian terhadap 3NF dilakukan dengan cara melihat apakah terdapat atribut bukan *key* tergantung fungsional terhadap atribut bukan *key* yang

lain (disebut ketergantungan transitif atau *transitive dependence*). Dengan cara yang sama, maka setiap ketergantungan transitif dipisahkan. 3NF sudah cukup bagus dalam arti bahwa anomali yang dikandungnya sudah sedemikian minimum (hampir tidak ada).

d. Normalisasi Keempat *Boyce-Codd Normal Form* (4 NF / BCNF)

Suatu relasi dikatakan BCNF bila di dalamnya berisi atribut yang berfungsi sebagai *candidate key* sehingga salah satu dari *candidate key* tersebut menjadi *primary key*.

2.2 Teori-teori Pendukung

2.2.2 SQL Server 2000

SQL Server 2000 adalah sebuah RDBMS (*Relational Data Base Management System*) yang bersifat *open source* dan berlevel tinggi. Dikembangkan oleh sebuah firma konsultan di Swedia yang dinamakan TcX. Mereka menciptakan sebuah sistem basis data yang sangat cepat dan fleksibel yang berdasarkan sebuah DBMS (*Database Management System*) yang bernama SQL Server 2000.

Selain itu SQL Server 2000 juga *portable*, ia telah dapat diaplikasikan atau di *port*kan ke hampir setiap *platform.* Ini berarti kita tidak harus mengganti *platform* utama kita untuk menggunakan SQL Server 2000. Dan jika kita ingin pindah ke *platform* baru pun, ada kemungkinan *port* SQL Server 2000 bisa digunakan di *platform* yang baru.

SQL Server 2000 juga sangat murah. Untuk sebuah SQL Server 2000 yang tidak terlisensi dan *full version,* harganya adalah 0 atau gratis. Untuk

melisensikan kopi anda hanya perlu mengeluarkan $200. Ini merupakan sebuah harga yang sangat murah, mengingat apa yang bisa kita dapatkan dari uang tersebut. Sistem basis data yang menyediakan hampir setengah fitur dari fitur yang dimiliki SQL Server 2000 dapat menghabiskan uang sepuluh ribuan dolar US. SQL Server 2000 dapat melakukan lebih baik dari apa yang dilakukan basis data lain dan dengan harga yang lebih murah. (Maskakowski, Mark, 2000, p10).

2.2.3 Perpustakaan

2.2.3.1 Definisi Perpustakaan

Pada awal 1980-an, pengertian perpustakaan ialah sebuah ruangan, bagian sebuah gedung, ataupun gedung itu sendiri yang digunakan untuk menyimpan buku dan terbitan lainnya yang disimpan menurut tata susunan tertentu untuk digunakan pembaca, bukan untuk dijual. (Basuki, 1991, p3).

2.2.6.2 Konsep Perpustakaan

Dalam pengertian buku dan terbitan lainnya termasuk didalamnya semua bahan cetak (buku, majalah, laporan, pamflet, *prosiding* manuskrip (naskah), lembaran musik), berbagai karya media *audio-visual* seperti film, slaid (*slide*), kaset, piringan hitam, bentuk mikro seperti mikro film, mikrofis, dan mikro buram (*microque*) (Basuki, 1991, p3).

Menurut *International Federation of Library Associations and Institutions* (IFLA) " definisi perpustakaan adalah sebagai kumpulan materi

tercetak dan media noncetak dan atau sumber informasi dalam komputer yang disusun secara sistematis untuk digunakan pemakai (Basuki, 1991, p4) “.

Kesimpulan dari dua definisi diatas adalah secara umum perpustakaan sebagai tempat untuk menyimpan koleksi buku dan bahan pustaka lainnya yang disimpan dengan tata aturan tertentu untuk digunakan pembaca.

2.2.6.3 Peranan Perpustakaan Dalam Masyarakat

Beberapa fungsi perpustakaan didalam masyarakat yaitu :

1. Sebagai sarana simpan karya manusia

   Dalam kaitannya dengan fungsi simpan, perpustakaan bertugas menyimpan khazanah budaya hasil masyarakat, khususnya karya cetak seperti buku, majalah, dan sejenisnya serta karya rekaman seperti kaset, CD-ROM, piringan hitam, dan sejenisnya.

2. Fungsi informasi

   Koleksi perpustakaan menyimpan banyak informasi yang dibutuhkan bagi pembacanya dan juga menyediakan jasa yang membantu keberhasilan sebuah penelitian. Penelitian ini mencakup arti luas karena dapat dimulai dari penelitian sedehana hingga pada penelitian yang rumit dan kompleks.

3. Fungsi pendidikan

   Perpustakaan merupakan sarana pendidikan nonformal dan informal, artinya perpustakaan merupakan tempat belajar di luar bangku sekolah maupun juga tempat belajar dalam lingkungan pendidikan sekolah.

4. Fungsi kultural

   Perpustakaan merupakan tempat untuk mendidik dan mengembangkan apresiasi budaya masyarakat. Pendidikan ini dapat dilakukan dengan cara menyelenggarakan pameran, ceramah, pertunjukan kesenian, pemutaran film bahkan bercerita untuk anak-anak. Dengan cara demikian masyarakat dididik mengenal budayanya.

5. Fungsi rekreasi

   Masyarakat dapat menikmati rekreasi kultural dengan cara membaca dan bacaan ini disediakan oleh perpustakaan.