BAB 2

LANDASAN TEORI

2.1 Konsep Dasar Basis data

Saat ini basis data merupakan suatu aplikasi yang tidak terpisahkan dari kehidupan kita sehari-hari. Contohnya : Pada saat kita melakukan pembelian di supermarket atau pada saat kita mengakses internet. Secara tidak sadar kita telah menggunakan suatu aplikasi yang dinamakan basis data. Sebelum adanya aplikasi basis data seperti DBMS, penyimpanan data masih menggunakan penyimpanan di dalam suatu file.

Menurut Connoly (2002, p7), sistem berbasiskan file adalah kumpulan program aplikasi yang memberikan servis kepada user untuk membuat suatu laporan. Setiap program mendefinisikan dan mengatur datanya masing-masing. Kemudian akhirnya ditemui banyaknya keterbatasan dari pendekatan menggunakan file, antara lain :

1. Data yang terpisah dan terisolasi
2. Data yang terduplikasi
3. Ketergantungan data
4. Format data yang tidak kompatibel
5. Query yang tetap
6. Tidak dapat mengantipasi perkembangan program aplikasi

Kemudian akhirnya digunakanlah pendekatan dengan menggunakan basis data dan DBMS.

2.1.1 Pengertian Data dan Basis data

Menurut Connolly (2002, p14) "*Database is a shared collection of logically related data, and a description of this data, designed to meet the information needs of an organization*", yang artinya basis data adalah suatu koleksi data yang saling berhubungan yang dirancang dan dipakai secara bersama untuk memenuhi kebutuhan informasi dari suatu organisasi.

Menurut Fathansyah (1999, p2), Basis Data terdiri atas dua kata yaitu Basis dan Data, Basis diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan data adalah representasi fakta dari dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, peristiwa, konsep keadaan, dan sebagainya yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi atau kombinasinya.

Menurut Kronke (2002, p15) "*Database is a self describing collection of integrated records* ", yang artinya basis data adalah suatu koleksi data yang menggambarkan integrasi antara record yang satu dengan record yang lainnya. *A self Describing* disini menggambarkan struktur data yang saling terintegrasi dalam suatu tempat yang dikenal sebagai kamus data atau metadata.

Dari definisi di atas dapat disimpulkan bahwa basis data adalah suatu koleksi data yang saling berhubungan dan menggambarkan integrasi antara record yang satu dengan record yang lainnya yang dirancang dan dipakai untuk memenuhi kebutuhan informasi dari suatu organisasi.

2.1.2 Sistem Basis Data

Menurut C. J Date (2000, p5), Sistem basis data pada dasarnya ialah sebuah sistem yang menyimpan record-record secara terkomputerisasi yang bertujuan untuk memelihara informasi dan memungkinkan pengguna untuk mengakses, mendapatkan dan merubah informasi tersebut sesuai dengan kebutuhan.

Menurut Fathansyah (1999, p2), Sistem basis data merupakan sistem yang terdiri atas kumpulan file/tabel yang saling berhubungan dalam sebuah basis data di sebuah sistem komputer dan sekumpulan program (*DBMS*) yang memungkinkan beberapa pemakai atau program lain untuk mengakses dan memanipulasi file-file (tabel-tabel) tersebut.

2.1.3 Komponen Basis Data

Di bawah ini adalah komponen-komponen basis data:

- Perangkat keras (*Hardware*)

  Perangkat keras yang biasanya terdapat dalam basis data adalah: komputer, memori sekunder yang *on-line* (hardisk), memory sekunder yang *off-line* (tape atau *removable disk*) untuk keperluan back up data, dan media/perangkat komunikasi (untuk sistem jaringan).

- Sistem operasi

  Merupakan program yang mengaktifkan/memfungsikan sistem komputer, mengendalikan seluruh sumber daya dalam komputer dan melakukan operasi-operasi dasar dalam komputer.

- Basis Data

  Sebuah basis data dapat memiliki beberapa basis data. Setiap basis data dapat berisi/momiliki sejumlah objek basis data (seperti file/tabel, indeks, dll). Disamping berisi/menyimpan data, setiap basis data juga mengandung/ menyimpan definisi struktur baik untuk basis data maupun objek-objeknya secara detail.

- Sistem Manajemen Basis Data (*Database Management System/DBMS*)

  DBMS merupakan sebuah paket perangkat lunak yang komplek, yang digunakan untuk memanipulasi basis data, menentukan bagaimana data diorganisasi, disimpan, diubah, dan diambil kembali, juga menerapkan mekanisme pengamanan data, pemakaian data secara bersama, pemeriksaan keakuratan/konsistensi data, dsb.

- Pemakai (user)

  Adapun beberapa user yang memakai sistem basis data ini terdiri dari Programmer, user mahir, user umum, user khusus.

2.1.4 Bahasa Basis Data (*Database Language*)

DBMS merupakan perantara bagi pemakai dengan basis data dalam *disk*. Cara berinteraksi/berkomunikasi antara pemakai dengan basis data tersebut diatur dalam suatu bahasa khusus yang ditetapkan oleh perusahaan pembuat *DBMS*. Bahasa itu dapat kita sebut sebagai bahasa basis data yang terdiri atas sejumlah perintah (*statement*) yang diformulasikan dan diberikan oleh user dan dapat dikenali / diproses oleh *DBMS* untuk melakukan suatu aksi/pekerjaan tertentu. Contoh-contoh bahasa basis data adalah *SQL*, *DBase*, *QUEL*, dsb.

Menurut Fathansyah, bahasa basis data biasanya dapat dibagi ke dalam dua bentuk yaitu :

1. *Data Definition Language* (*DDL*)

   Struktur/skema basis data yang menggambarkan/mewakili desain basis data secara keseluruhan dispesifikasikan dengan bahasa khusus yang disebut *Data Definition Language* (*DDL*). Dengan bahasa inilah kita dapat membuat tabel baru, membuat indeks, mengubah tabel, menentukan struktur penyimpanan tabel, dsb. Hasil dari kompilasi perintah *DDL* adalah kumpulan tabel yang disimpan dalam file khusus yang disebut kamus data (*data dictionary*). Contoh perintah itu adalah: *CREATE, ALTER, DROP*.

2. *Data Manipulation Language* (*DML*)

   Merupakan bentuk bahasa basis data yang berguna untuk melakukan manipulasi dan pengambilan data (akses) pada suatu basis data oleh user/pemakai. Dengan *DML* kita dapat melakukan:

   - Pengambilan data yang tersimpan di dalam basis data,
   - Penyisipan/penambahan data baru ke dalam basis data,
   - Penghapusan data dari suatu basis data,
   - Pengubahan data di dalam basis data.

   Contoh bahasa perintah *DML*, yaitu: *SELECT, INSERT, UPDATE, DELETE*.

2.1.5 Sistem Manajemen Basis Data (*DBMS*)

Menurut Connoly (2002, p16) Sistem Manajemen Basis Data (*DBMS*) adalah sebuah *software* yang menangani segala akses ke basis data yang dilakukan oleh user/pengguna. Fasilitas yang terdapat pada *DBMS* antara lain adalah menambah dan menghapus file atau tabel, mendapatkan kembali data yang tersimpan dan

memperbaharui data dalam file atau tabel. Contoh perangkat lunak yang termasuk dalam *DBMS* yaitu: *Ms-Access, MS-SQLServer, Oracle*.

Suatu *DBMS* harus mendukung fungsi - fungsi sebagai berikut :

1. Sebagai tempat penyimpanan data, pengambilan kembali data yang diinginkan, dan pembaharuan data. Ini merupakan fungsi pokok dari *DBMS*.
2. Sebagai kamus data (sistem katalog) yang bisa diakses oleh user.

   Sistem katalog atau kamus data adalah tempat penyimpanan dari informasi yang menjelaskan data yang ada di dalam basis data. Biasanya sistem katalog menyimpan:

   - Nama, tipe, dan ukuran dari item data;
   - Nama dari *relationship* (hubungan antar entitas/objek);
   - Batasan integritas dari suatu data;
   - Nama dari pemakai (user) yang memiliki hak akses terhadap data;
   - Skema eksternal, konseptual, dan internal dan pemetaan antar skema;
   - Pemakaian statistik seperti frekuensi dari transaksi.
3. Mendukung transaksi

   DBMS harus menyediakan mekanisme yang dapat menjamin isi dari basis data selalu mengikuti perubahan sesuai dengan transaksi yang dilakukan sehingga data yang ada di dalam basis data selalu akurat dan terjamin kebenarannya.
4. Mengontrol *Concurrency*

   DBMS memungkinkan pemakai untuk mengakses data bersama secara bersamaan dan menjamin tidak terjadinya gangguan pada basis data

5. Menyediakan perbaikan data kembali (*recovery*). Jika terjadi error pada hardware atau software atau adanya transaksi yang gagal dilakukan, maka basis data dapat dikembalikan pada kondisi yang semula (konsisten).
6. Menyediakan autorisasi bagi user

   DBMS harus dapat menjamin bahwa hanya user yang memiliki otorisasi/hak akses saja yang dapat mengakses basis data.
7. Membantu mendukung komunikasi data
8. Meningkatkan integritas data
9. Mengurangi ketergantungan data
10. Menyediakan kegunaan-kegunaan lainnya pada level internal seperti: memonitor pemakaian basis data dan operasi, pemakaian program analisa statistik, dll.

Adapun keuntungan dan kerugian DBMS adalah sebagai berikut:

Keuntungan DBMS :

- Mengurangi redudansi data

  Yaitu mencegah adanya data yang sama dalam suatu tempat.
- Konsistensi data

  Dengan adanya pencegahan terhadap munculnya redudansi data, maka mengurangi ketidakkonsistenan data yang terjadi.
- Mendapatkan informasi data yang lebih banyak, tanpa kehilangan informasi data yang telah ada/data aslinya.
- Data dapat digunakan secara bersama-sama (sharing data)
- Meningkatkan integritas data, keabsahan dan kekonsistenan data yang disimpan.
- Meningkatkan keamanan data dari yang tidak berotorisasi.
- Meningkatkan standar mutu.

- Penghematan dalam segi ekonomi. Schingga anggaran yang masih ada dapat dialokasikan untuk perkembangan dan pemeliharaan sistem yang lain.
- Mengurangi terjadinya konflik terhadap kebutuhan data antara user yang satu dengan yang lainnya.
- Meningkatkan kemampuan dalam pengaksesan data
- Meningkatkan produktivitas
- Meningkatkan pemeliharan terhadap data
- Meningkatkan konkuransi
- Meningkatkan pelayanan terhadap back up dan dan perbaikannya.

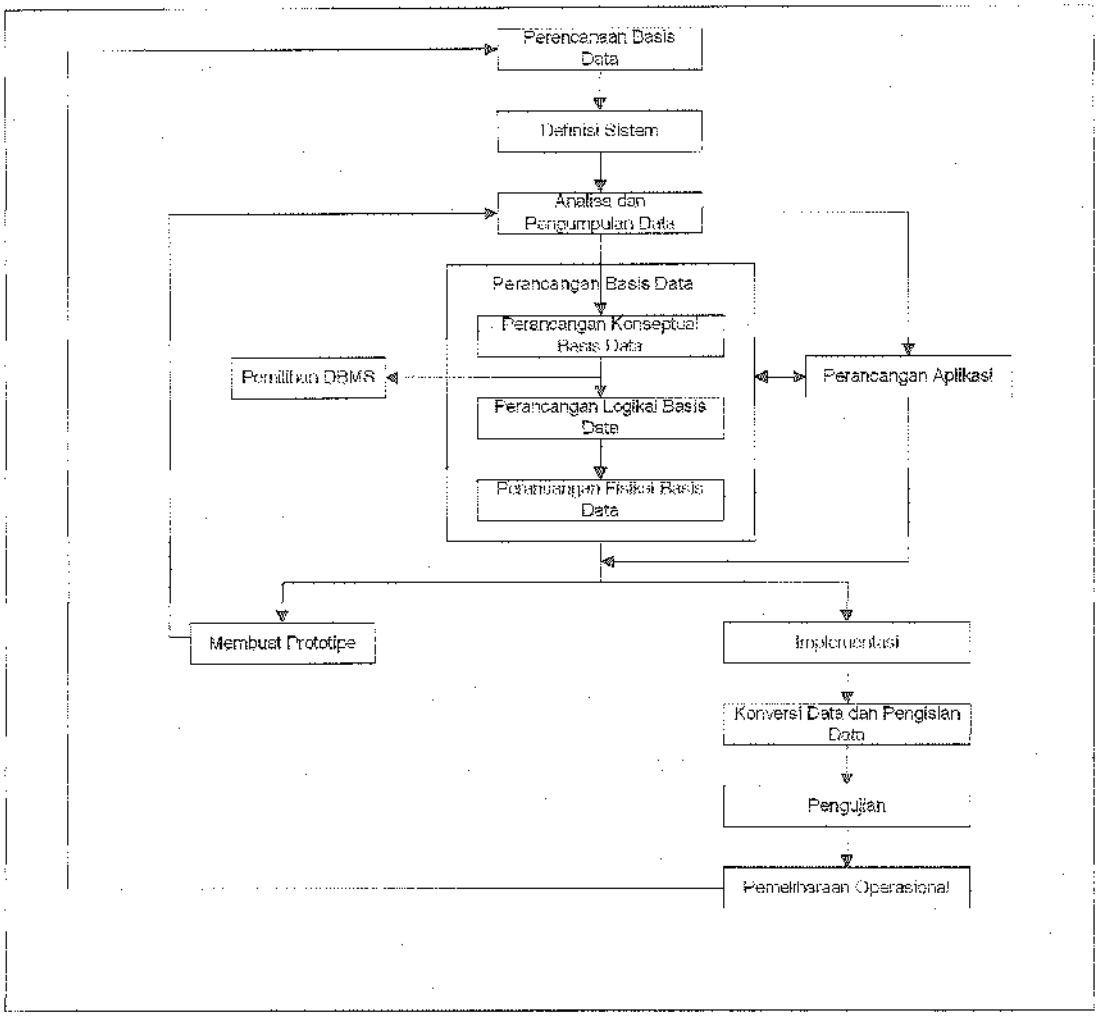
Kerugian DBMS :

- Software yang kompleks schingga perancang harus mengerti benar fungsi-fungsi yang ada untuk mendapatkan hasil rancangan yang baik.
- Membutuhkan tempat (*disk space*) / memori yang cukup besar untuk installasinya.
- Biaya tambahan terhadap kebutuhan hardware
- Biaya tambahan lain seperti pelatihan para staff, spesialis dan pekerja lainnya untuk menjalankan sistem yang baru.

2.1.6 Siklus Hidup Aplikasi Basis Data

Struktur dalam suatu siklus hidup basis data tidak selalu harus berurutan, tetapi dapat memiliki sejumlah pengulangan dari tahapan-tahapan yang sebelumnya melalui suatu umpan balik (*feedback*). Misalnya pada waktu melakukan perancangan basis data ditemukan masalah yang membuat kita perlu untuk melakukan analisa ulang terhadap kebutuhan tambahan (*requirement*) dari pengguna / user.

Gambar dibawah ini menunjukkan aktivitas-aktivitas utama yang berhubungan dengan tingkatan-tingkatan dalam siklus hidup basis data lebih lanjut / detail :



Gambar 2.1 Tahapan Siklus Hidup Aplikasi Basis Data

Tingkatan	Aktifitas Utama
Perencanaan basis data	Merencanakan tingkatan dari siklus hidup agar dapat direalisasikan secfektif dan seefisien mungkin.
Definisi sistem	Menetapkan ruang lingkup dan batasan dari aplikasi basis data, user dan area aplikasi.
Analisa dan pengumpulan kebutuhan	Mengumpulkan dan menganalisa kebutuhan dari user dan area aplikasi.
Perancangan basis data	Perancangan konseptual, logikal, dan fisikal basis data.
Pemilihan *DBMS*	Memilih *DBMS* yang tepat untuk aplikasi basis data.
Perancangan aplikasi	Merancang *user interface* dan memilih program aplikasi yang akan digunakan untuk memproses basis data.
Membuat prototipe	Membangun model aplikasi basis data, yang memungkinkan perancang atau user untuk melihat dan mengevaluasi bagaimana fungsi dari sistem akhir yang dihasilkan.
Implementasi	Realisasi fisik dari basis data dan rancangan aplikasi.
Konversi data dan *pengisian data*	Memasukkan data dari sistem yang lama ke dalam sistem yang baru.
Pengujian	Melakukan pengujian terhadap error/kesalahan yang terjadi dan menvalidasi basis data agar sesuai dengan kebutuhan yang dispesifikasi oleh user.
Pemeliharaan operasional	Melakukan implementasi basis data. Kemudian melakukan pengawasan dan pemeliharaan secara terus menerus. Jika perlu, kebutuhan yang baru dimasukkan ke dalam basis data melalui tahap sebelumnya pada siklus hidup.

Tabel 2.1 Ringkasan Aktifitas Utama dari Tahapan Siklus Hidup Aplikasi Basis Data

2.1.7 Kamus Data

Kamus data menurut McLeod (1996, p 328) adalah suatu ensiklopedi dari informasi mengenai tiap elemen data.

Kamus data adalah suatu penjelasan tertulis mengenai data yang berada didalam basis data. Yang termasuk didalam kamus data antara lain : file, aliran masukan, keluaran data yang ada pada diagram arus data.

	Simbol	Keterangan
- *Assign*	=	Menyatakan "terdiri dari"
- *Concatenation*	+	Menggabungkan elemen data dengan elemen data lainnya
- *Iteration*	{}	Pengulangan elemen data
- *Selection*	[]	Pilih satu dari beberapa alternatif
- *Option*	( )	Data ditambah, boleh ada atau tidak
- *Comment*	*...*	Penjelasan atau keterangan tentang suatu data – data elemen yang memilki "*unique value*" untuk setiap entry

- *Primary Key* @ *Candidate Key* yang dipilih ditandai dengan garis bawah

atau

2.1.8 Normalisasi

Normalisasi adalah suatu proses untuk mengorganisasikan file, yang bertujuan untuk menghilangkan elemen grup yang berulang-ulang. Normalisasi yang umum dipakai sampai dengan bentuk normal ketiga, yaitu :

1. **Bentuk normal pertama (1NF)**

   Pada tahap ini, bentuk tabel yang tidak normal yang diperoleh dari sumber data diubah kedalam bentuk normal pertama dengan menghilangkan *repeating group* yang terjadi pada tabel tersebut. Sebuah relasi dalam bentuk normal pertama jika tidak mengandung repeating group. *Repeating group* adalah suatu atribut atau sejumlah atribut dalam suatu tabel yang memiliki banyak nilai yang sama pada suatu kejadian yang muncul secara berulang-ulang. *Repeating group* dapat dihilangkan dengan menempatkan data yang berulang tersebut kedalam sebuah relasi baru dengan menghubungkan atribut *key* yang sama diantara dua tabel tersebut.

2. **Bentuk normal kedua (2NF)**

   Sebuah relasi dalam bentuk normal kedua jika relasi tersebut sudah dalam bentuk normal pertama dan setiap atribut yang bukan *primary key* memiliki ketergantungan fungsional secara penuh/utuh (*full functional dependency*) pada atribut yang adalah *primary key*. Ketergantungan fungsional (*functional dependency*) dapat digambarkan sebagai berikut :

Diberikan sebuah tabel T berisi paling sedikit 2 buah atribut, yaitu A dan B. Kita dapat menyatakan notasi berikut ini:

$$A \Rightarrow B$$

Yang berarti A secara fungsional menentukan B atau B secara fungsional tergantung pada A, jika dan hanya jika untuk setiap kumpulan baris data (row) yang ada di tabel T, pasti ada 2 baris data (row) di tabel T dengan nilai untuk A yang sama, maka nilai untuk B juga pasti sama.

Sebuah tabel dikatakan tidak memenuhi normal kedua jika ketergantungannya hanya bersifat parsial (hanya tergantung pada sebagian dari *primary key*).

3. **Bentuk normal ketiga (3NF)**

   Sebuah relasi dalam bentuk normal ketiga jika relasi tersebut sudah berbentuk normal pertama dan kedua, dan tidak ada atribut yang bukan *primary key* yang memiliki ketergantungan transitif (*transitive dependency*) pada atribut yang adalah *primary key*. Ketergantungan transitif (*transitive dependency*) terjadi ketika ada atribut yang bukan merupakan *primary key*, yang memiliki ketergantungan pada atribut lain yang juga bukan merupakan *primary key*.

2.1.9 Diagram Hubungan Antar Entitas (E-R Diagram)

Diagram hubungan antar entitas adalah diagram yang digunakan untuk menggambarkan hubungan (*relationship*) antara satu obyek data dengan obyek data yang lain. Notasi-notasi simbolik di dalam E-R diagram yang dapat kita gunakan adalah:

- Persegi panjang, menyatakan Entitas (*Entity*).

  Entitas merupakan individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain. Sederhananya, entitas menunjuk pada individu suatu objek. Contohnya: Pelanggan, Mahasiswa.

- Lingkaran/Elip, menyatakan Atribut (Atribut yang berfungsi sebagai *Key* digarisbawahi).

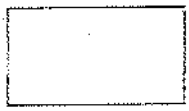
  Atribut seringkali merupakan istilah/sebutan dari kolom data. Atribut juga disebut sebagai elemen data, data field, data item. Istilah atribut lebih umum digunakan dalam perancangan basis data, karena lebih impresif dalam menunjukkan fungsinya sebagai pembentuk karakteristik (sifat-sifat) yang melekat pada sebuah tabel. Misalnya pada tabel Pelanggan, atributnya yaitu: Kode_pelanggan, nama_pelanggan, alamat_pelanggan. Pada tabel Barang, contoh atributnya yaitu: kode_barang, nama_barang, satuan.

- Belah ketupat menyatakan Relasi/hubungan.

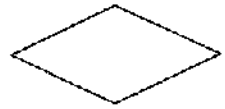
  Relasi menunjukkan adanya hubungan antara entitas yang satu dengan entitas yang lain.

- *Link*/Garis, sebagai penghubung antara relasi dengan entitas dan entitas dengan atributnya.

- Kardinalitas Relasi dapat dinyatakan dengan banyaknya garis cabang atau dengan pemakaian angka (1 dan 1 untuk relasi *one-to-one* (satu-ke-satu), 1 dan N untuk relasi *one-to-many* (satu-ke-banyak) atau N dan N untuk relasi *many-to-many* (banyak-ke-banyak)).



Entitas



Relasi

Atribut

Link/Garis

Relationship/hubungan yang dapat terjadi di antara dua entitas dapat berupa:

- *One-to-one relationship* (satu-ke-satu)

  yang berarti hubungan antara entitas yang satu dengan entitas yang lain adalah satu berbanding satu.

- *One-to-many relationship* (satu-ke-banyak)

  yang berarti hubungan antara entitas yang satu dengan entitas yang lain adalah satu berbanding banyak.

- *Many-to-one relationship* (banyak-ke-satu)

  yang berarti hubungan antara entitas yang satu dengan entitas yang lain adalah banyak berbanding satu.

- *Many-to-many relationship* (banyak-ke-banyak)

  yang berarti hubungan antara entitas yang satu dengan entitas yang lain adalah banyak berbanding banyak.

2.1.10 Metodologi Perancangan Basis Data

Menurut Connoly (2002, p418), metodologi perancangan merupakan suatu pendekatan struktural yang menggunakan bantuan sejumlah prosedur, teknik, dan *tools* /

peralatan serta dokumentasi untuk mendukung dan memfasilitasi proses dari suatu perancangan.

Untuk merepresentasikan metodologi dari suatu perancangan basis data, maka diperlukan suatu proses dalam perancangan yang dibagi menjadi tiga fase utama yaitu: perancangan konseptual basis data, perancangan logikal basis data dan perancangan fisikal basis data.

2.1.10.1 Perancangan Basis Data Konseptual

Merupakan suatu proses dalam membangun suatu informasi yang digunakan oleh perusahaan dengan mengabaikan segi pengimplementasiannya terlebih dahulu, misalnya yang berkaitan dengan *DBMS* yang akan digunakan, progam aplikasi, bahasa pemograman, atau hardware yang akan digunakan, segi *performance* / tampilannya dan semua hal-hal lain yang terkait / berhubungan secara fisikal.

Kegiatan utama di dalam perancangan konseptual basis data, yaitu:

Langkah 1 : Membangun model data konseptual lokal untuk setiap *view*.

Tujuan dari tahap ini ialah untuk membangun model data konseptual lokal dari suatu perusahaan untuk masing-masing *view* / pandangan user tertentu.

Selama penganalisaan dilakukan, pandangan dari sejumlah user harus diindentifikasi, dan dikombinasikan sehingga membentuk satu pandangan yang kolektif yang nantinya akan menghasilkan nama yang tepat sebagai satu kesepakatan bersama.

Untuk setiap model data konseptual lokal terdiri dari :

- Entitas
- Hubungan relasi antar entitas
- Atribut dan atribut domain

- *Primary Key* dan *Alternate Key*
- Batasan integritas

Model data konseptual didukung oleh suatu dokumentasi misalnya kamus data. Rincian dari dokumentasi/kamus data ini dapat diperoleh melalui berbagai langkah-langkah dibawah ini yaitu :

Langkah 1.1 : Mengidentifikasi tipe entitas

Tujuannya : Menentukan tipe entitas utama yang diperlukan oleh user yang dilakukan dengan mengindentifikasi spesifikasi kebutuhan dari user tersebut. Dari spesifikasi ini kita menentukan suatu nama, biasanya berupa kata benda atau frase kata benda. Sebagai contoh : dalam peminjaman suatu ruang perkantoran, diperlukan data-data sebagai berikut yaitu kode_staff, nama_staff, alamat_staff, dan lain sebagainya.

Langkah 1.2 : Mengidentifikasi tipe relasi/hubungan

Tujuannya : Setelah menentukan tipe entitas pada langkah yang sebelumnya, maka langkah selanjutnya adalah menentukan relasi yang timbul diantara tipe-tipe entitas tersebut. Relasi itu biasanya ditunjukkan dengan menggunakan sebuah kata kerja atau sebuah ungkapan yang berhubungan dengan kata kerja, sebagai contoh :

- *Staff Manages PropertyForRent*, *manages* disini merupakan sebuah kata kerja yang diartikan mengelola atau mengatur.
- *PrivateOwner Own PropertyForRent*, *own* merupakan kata kerja yang mempunyai arti memiliki.

Untuk mencegah kehilangan suatu relasi ketika dilakukannya suatu validasi terhadap suatu transaksi maka hal-hal berikut dibawah ini perlu dilakukan, yaitu :

- Menggunakan Diagram Hubungan Antar Entitas (E-R Diagram)

Digunakan untuk merepresentasikan entitas-entitas dan bagaimana entitas tersebut mudah berhubungan satu sama lainnya juga membantu membuat suatu gambaran tentang bagian-bagian dalam suatu perusahaan yang akan kita modelkan.

- Menentukan *Multiplicity Constraint* dari tipe relasi

  Setelah menentukan relasi ke dalam suatu model, langkah berikutnya adalah menentukan *Multiplicity* dari masing-masing relasi.

  Multiplicity Constraint ini digunakan untuk mengecek dan memelihara kualitas dari data. *Constraint* menyatakan tentang kejadian dari suatu entitas yang digunakan ketika suatu basis data diupdate, Untuk menentukan apakah perubahan yang dilakukan melanggar peraturan yang ditetapkan oleh perusahaan.

- Mengecek *Fan* dan *Chasm Traps*

  Setelah relasi ditentukan, maka lakukan pemeriksaan terhadap relasi dalam model tersebut apakah sudah merupakan representasi/perwujudan dari dunia nyata yang sebenarnya, dan apakah *Fan* dan *Chasm Trap* yang dibuat sudah benar. Definisi dari *Fan Trap* adalah suatu model yang merepresentasikan suatu relasi antar entitas tetapi alur relasinya memperlihatkan ambiguitas (kesamaan).

  Definisi dari *Chasm Trap* adalah suatu model dimana ada hubungan antara entitas yang satu dengan yang lain tetapi tidak ada relasi antara kedua entitas utama.

- Memeriksa setiap entitas yang mempunyai relasi minimal satu

  Pada saat pembuatan E-R Diagram, pastikan entitas mempunyai minimal satu rolasi dengan entitas yang lain. Jika memang ada entitas sudah mempunyai minimal satu relasi dengan entitas yang lain, maka langkah berikutnya adalah perhatikan kamus data.

- Mendokumentasikan tipe relasi

Bersamaan dengan tipe relasi diindentifikasi, tetapkan nama yang berarti dan jelas kepada user, lalu dokumentasikan relasi dan *Constraint* yang beragam (*Multiplicity Constraint*) tadi kedalam suatu kamus data.

Langkah 1.3 : Mengidentifikasi dan menghubungkan atribut dengan entitas maupun dengan tipe relasi.

Tujuannya : menentukan dan menghubungkan atribut dengan entitas maupun tipe relasi yang telah kita tentukan sebelumnya untuk direpresentasikan dalam suatu basis data. Untuk menentukan suatu entitas adalah dengan mencari *noun* (kata benda) atau *noun phrase* (ungkapan berupa kata benda) dalam spesifikasi kebutuhan yang ada pada user. Sedangkan atributnya dapat ditentukan dari suatu karakteristik yang dimiliki oleh salah satu entitas atau relasi.

Cara yang mudah dalam mengidentifikasi suatu entitas maupun sebuah relasi dalam sebuah spesifikasi kebutuhan adalah dengan mempertimbangkan informasi apa yang diperlukan dalam entitas maupun relasi tersebut. yang kemudian harus dideskripsikan dalam spesifikasi.Informasi-informasi tersebut adalah :

- Atribut Sederhana *(Simple attribute)* dan Atribut Komposit (*Composite Attribute*).

  Atribut Sederhana adalah atribut atomik yang tidak dapat dipilah lagi, Atribut Komposit merupakan atribut yang masih dapat diuraikan lagi menjadi sub-sub atribut yang masing-masing memiliki makna. Misalnya pada tabel Mahasiswa, atribut nama_mhs merupakan atribut sederhana, sedangkan atribut alamat_mhs dapat dikategorikan sebagai atribut komposit, karena dapat diuraikan menjadi beberapa sub atribut seperti alamat, nama_kota dan kode_pos, yang masing-masing memiliki makna.

- Atribut Bernilai Tunggal (*Single-Valued Attribute*) dan Atribut Bernilai Banyak (*Multivalued Attribute*).

  Atribut Bernilai Tunggal ditujukan pada atribut-atribut yang memiliki paling banyak satu nilai untuk setiap baris data. Contohnya pada data mahasiswa semua atribut (nim, nama_mhs, alamat_mhs, dan tgl_lahir) merupakan Atribut Bernilai Tunggal, karena atribut-atribut tersebut hanya dapat berisi 1 (satu) nilai. Sedangkan, Atribut Bernilai Banyak ditujukan pada atribut-atribut yang dapat kita isi dengan lebih dari 1 (satu) nilai, tetapi jenisnya sama. Kita dapat menambahkan atribut hobi pada data mahasiswa tersebut, seorang mahasiswa ada yang mempunyai 1 hobi saja, ada juga yang mempunyai banyak hobi dan bahkan ada mahasiswa yang tidak memiliki hobi, Atribut semacam ini tergolong Atribut Bernilai Banyak (*Multivalued Attribute*)

- Atribut Turunan *(Derived attribute)*

  Atribut Turunan adalah atribut yang nilai-nilainya diperoleh dari pengolahan atau dapat diturunkan dari atribut atau tabel lain yang berhubungan.

  Contohnya: penambahan atribut Angkatan dan IP (indeks prestasi) pada tabel mahasiswa, yang mana nilai-nilai pada atribut angkatan dapat diketahui dari atribut nim dengan nim misalnya : 980001, dimana dua karakter pertama dalam nim menyatakan dua digit bilangan tahun masuknya mahasiswa yang bersangkutan, sedang pada atribut IP, nilai-nilainya diperoleh dari pengolahan yang lebih rumit, yakni dengan menerapkan formula tertentu yang melibatkan atribut indeks_nilai yang ada di tabel nilai dan atribut sks yang ada di tabel kuliah.

Langkah 1.4 : Menentukan Domain Atribut

Tujuannya : menentukan Domain untuk semua atribut yang terdapat pada model data konseptual lokal

Domain adalah suatu nilai valid dari satu atau lebih atribut (nilai valid yang berlaku bagi suatu atribut).

Misalnya :

- Domain atribut dari Nomor_Pegawai (NoPegawai) mempunyai panjang lima karakter string, dengan dua karakter pertama berupa huruf, dan tiga karakter berikutnya berupa digit yang memiliki rentang 1-999. misalnya : NoPegawai → AB012.
- Atribut jenis kelamin dari suatu entitas Pegawai hanya mempunyai dua kemungkinan yaitu 'L'(Laki – laki) dan 'P' (Perempuan), domain dari atribut tersebut mempunyai satu karakter string yang terdiri dari 'L' dan 'P'.
- Domain NIM adalah bilangan bulat sepanjang sepuluh digit.

Spesifikasi dari suatu Domain untuk tiap atribut yang terdiri dari :

- Penentuan nilai dari suatu atribut
- Ukuran dan format dari suatu atribut
- Operasi dari suatu atribut
- Dan atribut mana yang dapat dibandingkan atau dikombinasikan dengan atribut yang lain.

Mendokumentasikan Domain dari atribut dilakukan setelah menentukan suatu Domain, yaitu dengan merecord nama dan karakteristik dari atribut ke dalam suatu kamus data.

Langkah 1.5 : Menentukan *Primary Key* dan *Candidate Key* dari suatu atribut

Tujuannya : menentukan *Candidate Key* untuk tiap tipe entitas dan jika terdapat lebih dari satu *Candidate Key*, pilih salah satu untuk dijadikan sebagai *Primary Key*.

Ada beberapa macam *key* yang dapat diterapkan pada suatu tabel yaitu, *Candidate Key, Primary Key, Alternate Key, Foreign Key.*

Candidate Key adalah suatu set atribut dari sebuah entitas yang memiliki nilai yang unik. Biasanya salah satu atribut didalam suatu relasi mempunyai nilai yang unik.

Primary Key adalah beberapa *Candidate Key* dalam suatu relasi yang dipilih salah satu menjadi *Primary Key*, harga atau nilai atribut yang unik dan dipakai untuk membedakan satu record dengan lainnya.

Contoh :

Supplier (S#, SNAME, STATUS CITY)

Purchase_Header (Purch#,S#, DATE)

Candidate Key : S#, SNAME

Primary Key : S#

Alternate Key adalah *candidate key* yang tidak dipilih menjadi *primary key*.

Contohnya : SNAME dari relasi supplier diatas.

Foreign Key adalah satu atau sejumlah atribut yang melengkapi suatu relasi(hubungan) yang menunjuk ke induknya. *Foreign Key* ditempatkan pada entitas anak dan sama dengan kunci primary induk direlasikan. Hubungan antara entitas induk dengan anak adalah hubungan satu lawan banyak (*one-to-many relationship*). Contohnya : S# pada Purchase_Header adalah Foreign Key yang memiliki referensi pada S# di Supplier.

Langkah 1.6 : Memeriksa redudansi yang terjadi terhadap suatu model.

Disini kita melakukan pengecekan terhadap model data konseptual lokal apakah ada redudansi yang terjadi, lalu hilangkan redudansi tersebut.

Langkah-langkahnya adalah sebagai berikut :

1. Periksa ulang *one-to-one relationship*

   Disini kita harus mengindentifikasi dua entitas yang menampilkan objek yang sama sebagai contoh : dua entitas yaitu Klien dan Penyewa, yang sesungguhnya sama, Klien merupakan sinonim dari Penyewa, maka kedua entitas tersebut dapat digabung menjadi satu. Dan jika kedua entitas tersebut mempunyai *Primary Key* yang berbeda, maka pilih satu dari mereka untuk menjadi *Primary Key*, dan yang lainnya sebagai *Alternate Key*.

2. Hilangkan redundansi dari suatu relasi

   Redudansi dari suatu relasi dapat terjadi jika informasi yang sama didapatkan melalui relasi yang lain. Sangat penting untuk memeriksa relasi diantara suatu entitas ketika melakukan pengecekan suatu redudansi.

Langkah 1.7 : Menvalidasi model konseptual lokal oleh user

Tujuannya : meyakinkan bahwa model konseptual lokal dapat mendukung kebutuhan yang diperlukan oleh user.

Disini kita memeriksa kemungkinan yang terjadi untuk memastikan bahwa model tersebut mendukung transaksi yang diperlukan oleh user, dua pendekatan yang dilakukan tersebut adalah :

1. Mendeskripsikan transaksi

   Dalam pendekatan ini kita melakukan pemeriksaan terhadap semua informasi yang dibutuhkan seperti entitas, relasi, dan atribut yang diperlukan dalam setiap transaksi yang disediakan oleh suatu model, lalu mendokumentasikan transaksi tersebut kedalam tiap-tiap kebutuhan transaksi.

2. Menggunakan jalur transaksi.

   Pendekatan ini menggunakan suatu diagram yang merepresentasikan suatu jalur pada tiap-tiap transaksi secara langsung dalam E-R Diagram. Pendekatan ini juga membantu perancang untuk memvisualisasikan suatu model yang tidak diperlukan dan yang sangat kritis dalam suatu transaksi

Langkah 1.8 : Meninjau ulang (*review*) model konseptual lokal dengan user.

Sebelum menyelesaikan langkah satu maka hal yang harus kita lakukan adalah dengan mereview ulang model konseptual lokal ini dengan user. Model konseptual data ini termasuk E-R diagram dan dokumentasi yang mendukung gambaran dari suatu model data.

Jika muncul kejanggalan dalam suatu model data , kita harus membuat suatu perubahan yang tepat, dimana memerlukan langkah-langkah sebelumnya. langkah-langkah ini akan terus diulang sampai user tersebut yakin dengan model data tersebut, yang mana model data tersebut merupakan representasi nyata dari suatu perusahaan yang kita modelkan.

2.1.10.2 Perancangan Basis Data Logikal

Perancangan Basis Data Logikal merupakan proses membangun sebuah model dari informasi yang digunakan di dalam perusahaan berdasarkan model data yang spesifik tanpa menggunakan *DBMS* atau peralatan fisikal lainnya. Model data logikal meliputi : E-R Diagram, Skema Relational, dan dokumentasi pendukung seperti kamus data.

Ada dua kegiatan utama di dalam perancangan basis data logikal, yaitu :

Langkah 2 : Membangun dan menvalidasi model data logikal lokal

Pada tahap ini kita memetakan setiap model data konseptual yang telah dibuat pada tahap perancangan konseptual ke dalam model data logikal yang terdiri dari E-R Diagram, skema relational, dan dokumentasi pendukung.

Adapun kegiatan-kegiatan yang dilakukan pada tahap ini adalah:

Langkah 2.1 : Menghilangkan fitur-fitur yang tidak kompatibel dengan model relational

Hal-hal yang dilakukan antara lain:

- Menghilangkan tipe relasi binari *many-to-many* (banyak-ke-banyak)

  Jika masih ada relasi *many-to-many* (banyak-ke-banyak) pada model data konseptual, maka kita gantikan dengan dua buah relasi *one-to-many* (satu-ke-banyak) dengan mengidentifikasikan sebuah entitas baru sebagai penghubung.

- Menghilangkan tipe relasi rekursif *many-to-many* (banyak-ke-banyak)

  Relasi rekursif merupakan jenis relasi dimana setiap jenis entitas memiliki sebuah relasi dengan dirinya sendiri. Relasi tersebut dapat digantikan dengan mengidentifikasi sebuah entitas baru sebagai penghubung dengan hubungan *one-to-many*.

- Menghilangkan jenis relasi yang kompleks

  Relasi yang kompleks adalah sebuah relasi yang terjadi antara tiga atau lebih entitas. Relasi yang kompleks ini dapat disederhanakan dengan menempatkan sebuah entitas baru di tengah entitas yang sebelumnya dan mendefinisikan relasi (binari) *one-to-many* diantara setiap entitas.

- Menghilangkan atribut yang bernilai banyak (*Multivalued attribute*)

Atribut yang bernilai banyak adalah atribut yang memiliki berbagai macam nilai di dalam sebuah entitas. Kita bisa mendekomposisi atribut tersebut dan menempatkannya ke dalam entitas baru.

Langkah 2.2 : Mendefinisikan relasi pada model data logikal lokal

Pada tahap ini kita menentukan relasi/hubungan antar entitas yang telah dibuat. Relasi yang dimiliki oleh entitas yang satu dengan entitas yang lain digambarkan dengan *Primary Key / Foreign Key*. Hal yang dilakukan pertama kali ialah mengidentifikasikan terlebih dahulu entitas *'parent'* (induk) dan entitas *'child'* (anak). Kemudian posting *Primary Key* yang ada di entitas *"parent"* ke dalam entitas *'child'* untuk berperan sebagai *Foreign Key*. Untuk relasi *one-to-many*, entitas *'parent'* yang ditempatkan pada sisi yang memiliki relasi satu dan entitas anak pada sisi yang memiliki relasi banyak. Untuk relasi *many-to-many*, buat sebuah entitas baru, kemudian copykan *Primary Key* dari setiap entitas parent ke dalam entitas baru tersebut untuk berperan sebagai *Foreign Key* dan hubungkan kedua entitas parent tersebut dengan entitas yang baru.

Langkah 2.3 : Validasi relasi menggunakan normalisasi

Pada tahap ini kita menvalidasi sejumlah atribut di dalam setiap relasi/hubungan dengan menggunakan aturan normalisasi. Normalisasi digunakan untuk menghindari terjadinya duplikasi dari data. Normalisasi menjamin bahwa hasil dari model yang telah kita rancang benar-benar mencerminkan model yang ada di perusahaan, yang memiliki konsistensi, redundansi yang minimal dan stabilitas yang maksimum.

Langkah 2.4 : Validasi relasi dengan transaksi-transaksi yang sesuai dengan kebutuhan pengguna

Pada tahap ini kita mengecek relasi/hubungan yang telah dibuat pada tahap sebelumnya dan menjamin tidak ada kesalahan dalam menentukan relasi/hubungan antar

entitas.Tujuan dari tahap ini adalah menvalidasi model data logikal lokal untuk menjamin agar model yang telah dirancang dapat mendukung transaksi-transaksi yang diperlukan sesuai dengan kebutuhan user.

Langkah 2.5 : Mendefinisikan batasan-batasan integritas

Batasan-batasan integritas merupakan batasan yang digunakan untuk menjaga basis data dari inkonsistensi. Pada tahap ini kita menentukan batasan-batasan integritas yang diperlukan. Ada lima jenis batasan integritas yang perlu diperhatikan, yaitu:

1. Data yang diperlukan (*Required Data*)

   Beberapa atribut harus memiliki nilai yang valid, dengan kata lain tidak boleh ada yang bernilai *null* (tidak bernilai). Ketentuan tersebut harus diidentifikasikan pada waktu kita mendokumentasikan atribut-atribut di dalam kamus data.

2. Batasan Atribut Domain (*Attribute Domain Constraint*)

   Setiap atribut memiliki domain, yaitu sejumlah set dari nilai-nilai yang legal. Contohnya: Jenis kelamin dari entitas pegawai adalah salah satu dari "P" atau "L", jadi domain dari atribut jenis kelamin adalah sebuah karakter string tunggal yang berupa "P" atau "L"

3. Integritas Entitas (*Entity Integrity*)

   Ketentuan dari sebuah entitas yang benar adalah *Primary Key* dari sebuah entitas tidak boleh bernilai *null* (tidak bernilai).

4. *Referential Integrity*

   Maksud dari *Referential Integrity* ialah jika *Foreign Key* memiliki sebuah nilai, maka nilai tersebut harus menunjuk kepada *tuple* yang terdapat pada relasi induk (*parent*).

5. Batasan Perusahaan (*Enterprise Constraints*)

Batasan ini seringkali disebut sebagai aturan bisnis. Untuk melakukan perubahan pada suatu entitas harus sesuai dengan ketentuan perusahaan yang ada di dunia nyatanya.

Langkah 2.6 : Mereview kembali model data logikal lokal dengan pengguna/user

Tujuan dari tahap ini adalah untuk menjamin model data logikal lokal dan dokumentasi pendukung yang menjelaskan model tersebut benar-benar menggambarkan proses bisnis yang nyata dari perusahaan dan dapat mendukung transaksi-transaksi sesuai dengan kebutuhan pengguna.

Langkah 3 : Membangun dan menvalidasi model data logikal global.

Aktifitas pada tahap ini meliputi:

Langkah 3.1 : Menggabungkan model data logikal lokal ke dalam model global

Untuk menggabungkan model data logikal lokal ke dalam model global ada beberapa hal yang perlu dilakukan antara lain:

- Memeriksa kembali nama dan isi dari sejumlah entitas/relasi dan *Candidate Key*-nya. Jangan sampai dua entitas/relasi atau lebih memiliki nama yang sama tetapi pada kenyataannya memiliki peran yang berbeda demikian juga sebaliknya.
- Memeriksa kembali nama dan isi dari relasi / *Foreign Key*

  Pada tahap ini kita menjamin bahwa entitas/relasi yang memiliki nilai yang sama menggambarkan konsep yang sama di dunia nyatanya.
- Menggabungkan relasi / *Foreign Key* dari model data lokal yang memiliki nama yang sama dan fungsi yang sama atau nama yang berbeda tetapi memiliki fungsi yang sama.

- Memasukkan (tanpa menggabungkan) relasi / *Foreign Key* yang unik ke dalam model global.
- Mengecek apakah ada entitas / relasi yang belum diidentifikasi
- Mengecek kebenaran dari *Foreign Key* yang terdapat di dalam relasi anak(child) dan lakukan perubahan bila perlu.
- Mengecek batasan integritas
- Menggambarkan E-R Diagram
- Melakukan update pada dokumentasi terhadap perubahan yang terjadi selama proses pengembangan model data global.

Langkah 3.2 : Validasi model data logikal global

Di tahap ini kita menvalidasi relasi / hubungan yang telah dibuat dari model data logikal global menggunakan teknik normalisasi untuk menjamin agar model data tersebut dapat mendukung transaksi yang dibutuhkan.

Langkah 3.3 : Mengecek untuk kebutuhan masa depan

Menjamin bahwa model data tersebut dapat juga mendukung kebutuhan perusahaan di masa yang akan datang.

Langkah 3.4 : Mereview model data tersebut dengan pengguna

Model data yang telah jadi direview kembali oleh user untuk menjamin bahwa model tersebut mencerminkan gambaran dari proses bisnis perusahaan di dunia nyatanya.

2.1.10.3 Perancangan Basis Data Fisikal

Merupakan suatu proses yang menghasilkan gambaran implementasi basis data yang kemudian disimpan dalam suatu penyimpanan sekunder. Gambaran implementasi

yang dihasilkan misalnya *base relations* dalam basis data, file organisasi dan suatu indeks yang digunakan sebagai suatu sarana untuk mengakses kedalam suatu basis data dengan efisien, serta berbagai batasan integritas dan keamanan lainnya yang berhubungan.

Dengan perancangan fisikal basis data dapat membantu seorang perancang dalam membuat suatu keputusan bagaimana basis data tersebut diimplementasikan ke dalam suatu *DBMS* tertentu.

Kegiatan utama di dalam perancangan fisikal basis data, yaitu:

Langkah 4 : Menerjemahkan model data logikal global untuk *DBMS* yang digunakan

Tujuan : menghasilkan skema basis data *relational* dari model data logikal global kedalam bentuk yang dapat diimplementasikan pada *relational DBMS* yang digunakan.

Aktifitas pertama dari perancangan fisikal basis data menerjemahkan relasi didalam model data logikal global kedalam suatu bentuk yang dapat diimplementasikan pada *relational DBMS* yang digunakan. Bagian pertama dari proses ini memerlukan pembandingan informasi yang dikumpulkan selama perancangan basis data logikal dan didokumentasikan kedalam kamus data. Aktifitas kedua yaitu menggunakan informasi ini untuk menghasilkan rancangan dari *base relations*. Proses ini membutuhkan pengetahuan yang mendalam dari fungsi-fungsi yang dihasilkan oleh *DBMS* yang digunakan. Contoh :

- Bagaimana membuat *base relations*
- Apakah sistem mendukung pendefinisian *Primary Key, Foreign Key dan Alternate Key*

- Apakah sistem mendukung pendefinisian atas data yang dibutuhkan (apakah sistem memperbolehkan atribut didefinisikan 'NOT NULL')
- Apakah sistem mendukung definisi dari domain
- Apakah sistem mendukung batasan integritas *relational*
- Apakah sistem mendukung definisi dari batasan-batasan perusahaan

Tiga aktifitas dari langkah 4 meliputi :

Langkah 4.1 : Merancang *base relations*

Tujuan : menentukan bagaimana menggambarkan *base relations* yang diidentifikasikan pada model data logikal global didalam *DBMS* yang digunakan..

Untuk memulai proses perancangan fisikal, kita mulai dengan mengumpulkan dan mengolah informasi tentang relasi yang dihasilkan selama perancangan logikal basis data. Informasi yang dibutuhkan dapat diperoleh dari kamus data dan definisi dari relasi yang digambarkan menggunakan *Database Design Language (DBDL)*. Untuk setiap relasi diidentifikasikan pada model data logikal global harus memiliki definisi sebagai berikut :

- Nama relasi
- Daftar dari atribut sederhana dalam tanda kurung besar
- *Primary Key, Alternate Keys* (AK) dan *Foreign Keys* (FK)
- Daftar dari *Derived attribute* dan bagaimana mereka dihitung
- *Referensial Integrity Constraints* untuk setiap *Foreign Keys* yang diidentifikasi

Dari kamus data, setiap atribut mempunyai :

- Domain yang terdiri dari tipe data, panjang, dan batasan lainnya yang ada pada domain

- Suatu nilai *default optional* untuk atribut-atribut
- Apakah atribut dapat bernilai *null* (tidak bernilai).

Untuk menggambarkan perancangan dari *base relations*, kita menggunakan suatu bentuk perluasan dari *DBDL* untuk mengartikan domain, *default values*, dan indikasi nilai *null*.

Langkah selanjutnya adalah memutuskan bagaimana mengimplementasikan *base relation*. Keputusan ini tergantung pada *DBMS* yang digunakan.

Rancangan dari *base relations* harus secara lengkap mendokumentasikan alasan dipilihnya rancangan yang diusulkan.

Langkah 4.2 : Representasi perancangan dari data yang diperoleh

Tujuan : menentukan bagaimana menggambarkan *Derived* data yang ada dalam model data logikal global pada *DBMS* yang digunakan.

Nilai suatu atribut yang dapat diperoleh dengan mengambil atau dari hasil perhitungan nilai atribut lainnya dikenal sebagai *Derived* atau *Calculated Attributes*.

- Sering kali, *Derived Attribute* tidak muncul dalam model data logikal tapi didokumentasikan dalam kamus data. Jika suatu *Derived Attribute* muncul dalam model, "*/*" digunakan untuk mengindikasikan bahwa itu adalah *Derived*.

Langkah 4.3 : Merancang batasan perusahaan

Tujuan : merancang batasan perusahaan untuk *DBMS* yang digunakan.

Rancangan dari batasan tergantung lagi dengan pilihan dari *DBMS*. Beberapa sistem menyediakan lebih banyak fasilitas dari lainnya untuk mengartikan batasan perusahaan. Seperti langkah sebelumnya, jika sistem dipenuhi oleh dengan standar *SQL*, beberapa batasan dapat dengan mudah untuk diimplementasikan.

Langkah 5 : Merancang gambaran fisikal

Tujuan : merancang file organisasi yang optimal untuk menyimpan *base relations* dan indeks yang dibutuhkan untuk mencapai hasil yang dapat diterima, ini adalah cara yang ditempuh dimana relasi akan berada pada penyimpanan sekunder.

Satu tujuan utama dari perancangan fisikal basis data adalah menyimpan data dengan cara yang efisien. Ada beberapa faktor yang dapat digunakan untuk mengukur efisiensi :

- *Transaction Throughput.*

  Merupakan sejumlah transaksi yang dapat diproses pada interval waktu yang diberikan.

- *Response Time*

  Merupakan waktu untuk menyelesaikan transaksi yang *single*.

- *Disk storage*

  Merupakan sejumlah *disk space* yang dibutuhkan untuk menyimpan file database. Perancang bisa meminimumkan jumlah dari penyimpanan *disk* yang digunakan.

Untuk meningkatkan kinerja, perancang fisikal basis data harus berhati-hati pada bagaimana empat komponen *hardware* dasar mempengaruhi sistem *performance*, yaitu :

- *Main memory*
- *CPU*
- *Disk I/O*
- *Network* .

Aktifitas yang terdapat pada langkah 5 terdiri dari :

Langkah 5.1 : Analisa transaksi

Tujuan : mengerti kemampuan dari transaksi yang akan berjalan dalam basis data dan untuk menganalisa transaksi yang penting.

Untuk merancang fisikal basis data secara efektif, adalah penting untuk mempunyai pengetahuan dari transaksi atau *query* yang dijalankan dalam basis data Ini meliputi informasi yang kualitatif dan kuantitatif. Dalam menganalisa transaksi, kita mengidentifikasikan kriteria *performance* seperti :

- Transaksi yang seringkali dilakukan dan mempunyai pengaruh yang signifikan pada *performance*.
- Transaksi yang bersifat kritis pada operasi bisnis
- Tingkat transaksi yang tinggi selama satu hari/minggu yang dihasilkan di dalam basis data.

Langkah 5.2 : Menentukan file organisasi yang efisien untuk setiap *base relations*

Satu tujuan utama perancangan fisikal basis data adalah menyimpan data dengan cara yang efektif. Tujuan utama dari langkah ini adalah untuk memilih file organisasi yang optimal untuk setiap relasi sesuai dengan ketentuan *DBMS* yang digunakan.

Langkah 5.3 : Menentukan bahwa penambahan indeks akan meningkatkan *performance* dari sistem

Tujuan : menentukan apakah dengan menambah indeks akan meningkatkan *performance* sistem. *Secondary Indexes* menyediakan mekanisme untuk menetapkan suatu *Key* tambahan untuk suatu *base relations* yang dapat digunakan untuk mendapatkan kembali data secara lebih efisien.

Langkah 5.4 : Mengestimasi kapasitas media penyimpanan yang dibutuhkan

Didalam proses perancangan perlu untuk melakukan estimasi terhadap jumlah *disk space* yang dibutuhkan untuk menyimpan basis data Sama seperti langkah sebelumnya, mengestimasi *disk space* sangat tergantung pada *DBMS* dan *hardware* yang akan digunakan untuk mendukung basis data. Secara umum, estimasi ini berdasarkan pada ukuran setiap *record* dan jumlah *record* dalam relasi. Selain itu juga perlu dipertimbangkan faktor peningkatan transaksi di masa mendatang.

Langkah 6 : Merancang *user view*

Tujuan dari langkah ini adalah untuk merancang *user view* yang telah diidentifikasikan pada tahap sebelumnya. *User view* didefinisikan untuk menyederhanakan permintaan pada basis data. Dalam *multiuser DBMS*, *user view* memainkan peran penting dalam mendefinisikan struktur dari basis data dan menjalankan keamanan.

Langkah 7 : Merancang standar keamanan

Tujuan : merancang standar keamanan basis data seperti yang dispesifikasi oleh user. Suatu basis data menggambarkan suatu sumber daya perusahaan yang penting, jadi keamanan untuk sumber daya ini sangat penting. Selama pengumpulan kebutuhan dan tahap analisa dari siklus hidup aplikasi basis data, kebutuhan keamanan yang spesifik seharusnya didokumentasikan dalam spesifikasi *system requirement*. Beberapa sistem menawarkan fasilitas berbeda dari sistem lainnya. Perancang basis data harus berhati-hati dengan fasilitas yang ditawarkan oleh *DBMS* yang digunakan. Secara umum *relational DBMS* menyediakan 2 tipe dari keamanan basis data yaitu :

- Keamanan sistem

Keamanan sistem melindungi akses dan kegunaan basis data pada sistem level, seperti *user name* dan *password*.

- Keamanan data

  Keamanan data melindungi akses dan kegunaan dari basis data objek (seperti *relations dan view)* dan kegiatan yang user dapat lakukan pada objek.

2.2 Teori – Teori Persediaan, Pembelian, dan Penjualan

2.2.1 Definisi Pembelian

Menurut Clifton, H. D dan Sutliffe, A. G (1994, p8), pembelian berkaitan dengan prosedur-prosedur yang memastikan bahwa semua materi, komponen, tools, peralatan dan item-item yang lain yang diperlukan oleh perusahaan tersedia pada waktu, tempat dan harga yang tepat.

Menurut Leenders, Feacon, Flynn dan Johnson (2002, p6) mendefinisikan pembelian adalah proses membeli, mempelajari kebutuhan, mencari dan memilih supplier, negosiasi harga dan kepastian pengiriman.

Menurut Mulyadi (1993, p302), Fungsi yang terkait dengan sistem pembelian adalah :

a. Fungsi Gudang, bertanggung jawab untuk mengajukan permintaan pembelian sesuai dengan posisi persediaan yang ada di gudang dan untuk menyimpan barang yang telah diterima oleh fungsi penerimaan.
b. Fungsi Pembelian, bertanggung jawab untuk memperoleh informasi mengenai harga barang, menentukan pemasok yang dipilih dalam pengadaan barang, dan mengeluarkan order 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.

2.2.2 Definisi Penjualan

Menurut Arens (1996, p368) penjualan merupakan kegiatan perusahaan untuk mengalihkan kepemilikan atas barang dan jasa yang telah tersedia, untuk dijual kepada pelanggan.

Menurut Baridwan (1994, p180) penjualan adalah kegiatan dimana penjual menentukan, melakukan dan menyerahkan barang–barang yang dijual kepada pembeli untuk memuaskan kebutuhan dari pembeli dengan mendapatkan keuntungan atas barang atau jasa tersebut.

Menurut Mulyadi (1995, p213) Fungsi yang terkait dengan sistem penjualan adalah :

a. Fungsi Penjualan, bertanggung jawab untuk menerima order, mengedit order, meminta otorisasi kredit, menentukan tanggal pengiriman dan bertanggung jawab atas transaksi penjualan.
b. Fungsi Gudang, bertanggung jawab untuk menyimpan dan menyiapkan barang yang dipesan, dan mengirimkan ke bagian pengiriman.
c. Fungsi Pengiriman, bertanggung jawab untuk menyerahkan barang ke pelanggan berdasarkan order pengiriman ke bagian penjualan.

2.2.3 Definisi Persediaan

Menurut Fress dan Warren (1996, p388) persediaan digunakan untuk mengartikan barang dagang yang disimpan untuk dijual dalam kegiatan operasional perusahaan dan bahan yang terdapat dalam proses produksi atau yang disimpan untuk tujuan tertentu.

Menurut Rangkuti (1998, p1) persediaan adalah sebagai suatu aktiva yang meliputi barang–barang milik perusahaan dengan maksud untuk dijual dalam suatu periode usaha tertentu / persediaan barang-barang yang masih dalam pengerjaan atau proses produksi ataupun persediaan bahan baku yang menunggu penggunaanya dalam suatu proses produksi.

Menurut Rangkuti (1998, p7), jenis-jenis persedian menurut fungsinya dapat dibedakan :

a. Batch Stock / Lot Size Inventory, yaitu persediaan yang diadakan karena pembelian atau pembuatan bahan-bahan atau barang-barang dalam jumlah yang lebih besar dari jumlah yang dibutuhkan saat itu.
b. Fluctuation Stock, yaitu persediaan yang diadakan untuk menghadapi fluktuasi, permintaan konsumen yang tidak dapat diramalkan.
c. Anticipation Stock, yaitu persediaan yang diadakan untuk menghadapi fluktuasi permintaan yang dapat diramalkan berdasarkan pola maksimum yang terdapat dalam satu tahun dan untuk menghadapi penggunaan atau penjualan atau permintaan meningkat.