BAB 2

LANDASAN TEORI

2.1 Teori Umum

2.1.1 Pengertian Data

Menurut Atzeni (2003, p2), data adalah informasi yang disimpan, yang memerlukan terjemahan untuk menghasilkan informasi.

Menurut Hoffer, Prescott, dan McFadden (2005, p5), data adalah fakta yang telah diketahui, yang dapat dikumpulkan dan disimpan dalam media komputer.

Data terdiri dari fakta-fakta dan simbol-simbol yang secara umum kurang berguna dikarenakan jumlahnya yang besar dan sifatnya yang kasar atau belum diolah (McLeod dan Schell, 2001, p9).

2.1.2 Pengertian Basis Data

Pengertian basis data menurut Connolly (2002, p14) adalah kumpulan data yang saling berhubungan secara logikal, dan keterangan mengenai data itu, yang dibuat untuk menemukan informasi yang dibutuhkan oleh sebuah organisasi. Basis data yang tunggal dapat digunakan oleh berbagai pengguna atau bagian dalam waktu yang bersamaan.

Menurut C. J. Date (2000, p 10) basis data adalah kumpulan data *persistent* yang digunakan oleh sistem aplikasi dari perusahaan tertentu. *Persistent* maksudnya adalah sekali data telah diterima oleh DBMS untuk dimasukkan ke dalam basis data, data tersebut hanya bisa dihapus dari basis data dengan menggunakan permintaan eksplisit ke DBMS.

Menurut Post (2005, p2), basis data adalah sebuah kumpulan data yang disimpan dalam suatu format yang sudah distandarisasi, yang dirancang agar dapat digunakan oleh *multiple users.*

2.1.3 Pengertian Sistem Basis Data

Menurut Connolly (2002, p4), sistem basis data pada dasarnya adalah sistem penyimpanan *record* yang terkomputerisasi di mana tujuan sebenarnya adalah menyimpan informasi dan membuat informasi tersebut selalu tersedia pada saat dibutuhkan. Keseluruhan sistem terkomputerisasi tersebut memperbolehkan pengguna menelusuri kembali dan mengubah informasi tersebut sesuai kebutuhan.

Sistem basis data adalah sistem penyimpanan yang terkomputerisasi di mana tujuan sebenarnya adalah menyimpan informasi dan membuat informasi tersebut selalu tersedia pada saat dibutuhkan.

2.1.4 DBMS *(Database Management System)*

Menurut Post (2005, p2), *database management system* adalah perangkat lunak yang mendefinisikan sebuah basis data, menyimpan data, mendukung bahasa *query*, menghasilkan laporan-laporan, dan menciptakan layar *entry* data.

Menurut Connolly (2002, p16), *database management system* adalah sistem perangkat lunak yang memungkinkan pengguna untuk mendefinisikan, membuat, memelihara, dan mengontrol akses menuju basis data.

2.1.4.1 Keuntungan DBMS

1. Mengontrol duplikasi data

   Basis data dapat mengurangi duplikasi data yang terjadi, tetapi tidak semua duplikasi data dapat dihilangkan, melainkan hanya dapat dikontrol. Hal ini dikarenakan terkadang duplikasi data tetap dibutuhkan untuk meningkatkan kemampuan.

2. Data yang konsisten

   Basis data dengan menghilangkan atau mengontrol duplikasi data yang akan mengurangi resiko tidak konsistennya dari data yang mungkin terjadi.

3. Data yang sama lebih memiliki banyak informasi

   Basis data dengan adanya integrasi data operasional maka memungkinkan bagi sebuah organisasi mendapatkan tambahan informasi dari data yang sama.

4. Penggunaan data bersama-sama

   Biasanya *file* dimiliki oleh seseorang atau departemen yang menggunakannya. Sebaliknya, suatu basis data dimiliki oleh organisasi secara keseluruhan dan dapat diakses kepada pengguna yang memiliki hak akses.

5. Integritas data telah dikembangkan

   Integritas data memiliki arti dari konsisten dan keabsahan dari data yang disimpan. Integritas biasanya menegaskan arti dari kata *constraint*, yang merupakan aturan yang tidak dapat dilanggar oleh pemakai basis data.

6. Keamanan telah dikembangkan

   Keamanan basis data akan melindungi dari pemakai yang tidak memiliki hak mengakses. Hal ini biasanya dilakukan dengan adanya nama pemakai dan *password* untuk mengidentifikasi hak yang dimiliki oleh setiap pemakai.

2.1.4.2 Kerugian DBMS

1. Kompleksitas

   Fungsionalitas yang diharapkan dari DBMS membuat DBMS menjadi *software* yang sangat kompleks. Pengguna DBMS harus memahami dengan baik setiap fungsi dari DBMS untuk mendapatkan keuntungan maksimum dari DBMS tersebut. Ketidakpahaman atas sistem dapat menyebabkan pengambilan keputusan perancangan yang buruk akan mengakibatkan dampak serius pada organisasi.

2. Ukuran

   Kompleksitas dan fungsional yang baik membuat DBMS menjadi sebuah *software* yang berukuran sangat besar dan membutuhkan *disk space* yang sangat besar serta jumlah *memory* yang besar agar dapat dijalankan seefisien mungkin.

3. Harga DBMS

   Harga DBMS sangat bervariasi tergantung dari lingkungan dan fungsi yang disediakan termasuk biaya pemeliharaan.

4. Biaya perangkat keras tambahan

   Kebutuhan *disk storage* untuk DBMS dan *database* menyebabkan kemungkinan untuk membeli *storage*

tambahan. Selain itu, untuk mendapatkan kemampuan yang diinginkan, dibutuhkan perangkat keras yang cepat dan baik tapi mahal.

5. Biaya konversi

Pada beberapa situasi, biaya yang dibutuhkan untuk mengubah aplikasi agar dapat berjalan pada DBMS yang baru dapat jauh lebih mahal dari biaya perangkat keras tambahan. Biaya tersebut juga termasuk biaya pelatihan pegawai untuk menggunakan sistem yang baru, dan mungkin juga biaya pegawai spesialis untuk membantu konversi dan pelaksanaan sistem.

2.1.4.3 Komponen – Komponen DBMS

Menurut Connolly (2002, p18), ada lima komponen DBMS, yaitu:

1. Perangkat keras (*Hardware*)

Hal ini berupa komputer tunggal pribadi (PC), *mainframe* tunggal hingga jaringan komputer. Perangkat keras dapat tergantung pada kebutuhan perusahaan dan DBMS yang digunakan.

2. Perangkat lunak (*Software*)

   Hal ini terdiri dari perangkat lunak DBMS sendiri, program-program aplikasi, dan sistem operasi termasuk perangkat lunak.

3. Data

   Hal ini merupakan komponen paling penting dalam lingkungan DBMS. Data berperan sebagai penghubung antara komponen mesin dengan komponen manusia. Basis data mengandung data operasional dan meta-data. Struktur basis data disebut skema.

4. Prosedur

   Hal ini berarti sejumlah instruksi dan pengaturan yang menentukan rancangan dan pengguna dari basis data. Pemakai sistem dan pegawai yang mengelola basis data membutuhkan dokumen prosedur tentang bagaimana untuk menggunakan atau menjalankan sistem. Instruksi-instruksi tersebut dapat berupa:

   a. *Log on* ke DBMS.
   b. Penggunaan sebagian fasilitas DBMS atau program aplikasi.
   c. Memulai dan menghentikan DBMS.

d. Membuat *backup* dari basis data.
e. Menangani kegagalan perangkat keras atau perangkat lunak.
f. Mengubah struktur dari sebuah tabel, mengatur kembali basis data dalam *multiple disk*, meningkatkan performa, atau membuat arsip data pada penyimpanan sekunder.

5. Orang

Komponen terakhir adalah orang yang terlibat dalam sistem, termasuk data dan *database administrator*, perancang basis data, pengembang aplikasi, dan *end user*.

2.1.4.4 Fungsi DBMS

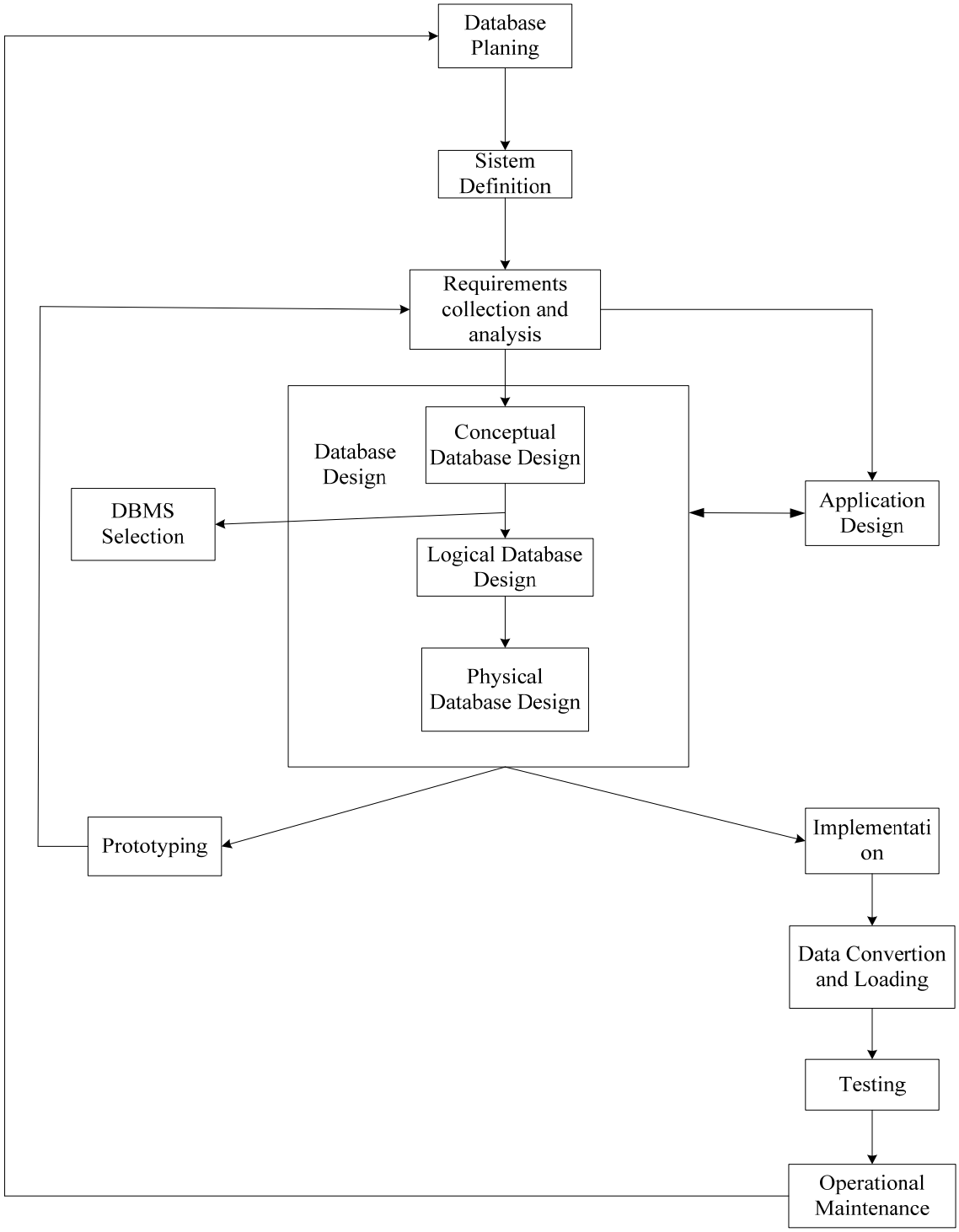
Menurut Connolly (2002, p16), fungsi DBMS terdiri dari:

1. Penyimpanan, pengambilan, dan *update* data merupakan kemampuan untuk menyimpan, *restore*, dan *update* data di dalam basis data.
2. Katalog yang dapat diakses oleh pengguna, dimana deskripsi data disimpan dan dapat diakses oleh pengguna.
3. Pendukung transaksi, merupakan sebuah mekanisme di mana akan memastikan bahwa semua *update* yang dibuat oleh transaksi telah dibuat.

4. Layanan *control concurrency*, merupakan sebuah mekanisme untuk memastikan bahwa basis data di-*update* secara benar ketika banyak pengguna telah meng-*update* basis data pada saat yang bersamaan.
5. Layanan perbaikan, merupakan sebuah mekanisme untuk memperbaiki basis data yang apabila dalam sebuah kejadian rusak dengan berbagai cara.
6. Layanan autorisasi, merupakan sebuah mekanisme untuk memastikan bahwa hanya pengguna yang memiliki ijin, yang dapat mengakses basis data.
7. Pendukung komunikasi data, kemampuan berintegrasi dengan perangkat lunak komunikasi.

2.1.5 Daur Hidup Basis Data (*Database Lifecycle*)

Sistem informasi adalah sumber yang dapat mengumpulkan, mengelola, mengatur, dan membagi segala informasi untuk organisasi. Menurut Connolly (2002, p270) sejak tahun 1970, sistem basis data telah menggantikan penggunaan *file based system* sebagai bagian sistem informasi dari perusahaan. Menurut Connolly (2002, p284), suatu aplikasi basis data dianalisis dan dirancang dalam tahapan-tahapan berikut:



Gambar 2.1 *Database Application Lifecycle*

1. Perencanaan basis data (*Database Planing)*

   Perencanaan basis data adalah kegiatan pengaturan yang memungkinkan tahap-tahap dalam aplikasi basis data dapat diwujudkan secara efisien dan secara seefektif mungkin, hal ini disebut oleh Connolly (2002, p273). Perencanaan basis data harus diintegrasikan dengan keseluruhan strategi informasi dari perusahaan.

2. Pendefinisian sistem (*System Definition)*

   Menurut Connolly (2002, p274), pendefinisian sistem menggambarkan ruang lingkup dan batasan aplikasi basis data dan pandangan pengguna. Hal ini sangat penting dilakukan dalam proses perancangan basis data agar lebih terfokus pada proyek basis data yang dibuat.

   Menurut Connolly (2002, p275), pandangan pengguna sangat diperlukan untuk mengidentifikasi informasi-informasi yang dibutuhkan oleh pengguna. Pandangan pengguna menggambarkan apa yang dibutuhkan oleh aplikasi basis data dari sudut pandang jabatan tertentu, seperti manager atau pengawas, maupun dari sudut pandang area aplikasi perusahaan, seperti pemasaran, personalia, dan pengawasan persediaan, dalam hubungannya dengan data yang akan disimpan dan transaksi yang akan dijalankan terhadap data itu.

3. Analisa dan pengumpulan kebutuhan (*Requirement Collection and Analysis)*

Tahap selanjutnya yang akan dilakukan setelah pendefinisian sistem adalah tahap pengumpulan kebutuhan dan analisis. Dalam tahap ini dilakukan proses pengumpulan dan analisis informasi tentang bagian organisasi yang akan didukung oleh aplikasi basis data, dan menggunakan informasi ini untuk mendefinisikan kebutuhan pengguna terhadap sistem yang baru.

Menurut Connolly (2002, p302), suatu proses resmi dalam menggunakan teknik-teknik seperti wawancara atau kuisoner untuk mengumpulkan fakta-fakta tentang sistem dan kebutuhan-kebutuhannya dinamakan dengan teknik *fact-finding*. Ada lima kegiatan yang dipakai dalam teknik ini, yaitu:

a. Pengkajian terhadap dokumentasi

Pengkajian terhadap dokumen seperti laporan, dokumen penting, faktur, dan berbagai data yang berkaitan dengan sistem. Kajian tersebut akan sangat berguna dalam menentukan kebutuhan dari basis data yang akan dibangun.

b. Wawancara

Digunakan untuk mengumpulkan informasi secara langsung melalui tatap muka dengan individual yang berkaitan. Beberapa tujuan dari teknik ini adalah mengumpulkan fakta-fakta, memeriksa kebenaran fakta yang ada dan mengklarifikasinya, membangkitkan semangat, melibatkan pengguna akhir, mengidentifikasi kebutuhan-kebutuhan, dan mengumpulkan ide-ide dan pendapat.

c. Mengobservasi proses kegiatan kerja perusahaan

Sangat berguna apabila kita menemukan kejanggalan analisis data dari metode yang telah dilakukan sebelumnya dengan berpartisipasi secara langsung untuk mendapatkan gambaran langsung dari sistem yang berjalan.

d. Penelitian

Mencari referensi seperti jurnal, buku, dan melalui *internet* yang memiliki pemecahan atas masalah serupa dengan masalah yang sedang dihadapi.

e. Kuisoner

Pengumpulan dokumen dengan tujuan khusus yang didapat dari pengumpulan fakta-fakta dari banyak orang sambil menjaga kontrol terhadap tanggapan yang diberikan.

4. Perancangan basis data (*Database Design)*

Perancangan basis data adalah proses pembuatan suatu rancangan untuk basis data yang akan mendukung operasi dan tujuan perusahaan.

Pada tahap ini dibahas tujuan dan kegunaan dari pemodelan data dan menggambarkan tiga fase perancangan basis data yaitu perancangan basis data konseptual, logikal, dan fisikal.

a. Desain basis data konseptual

Hal ini merupakan proses membangun sebuah model dari informasi yang digunakan dalam perusahaan, bebas dari semua pertimbangan fisikal.

b. Desain basis data logikal

Hal ini merupakan proses membangun sebuah model dari informasi yang digunakan dalam sebuah dasar perusahaan pada sebuah data model yang khusus, tetapi bebas dari fakta DBMS dan pertimbangan fisikal.

c. Desain basis data fisikal

Hal ini merupakan proses menghasilkan sebuah penjelasan dari implementasi pada basis data di *secondary storage*, termasuk menentukan relasi dasar, catatan atau arsip organisasi, dan *index* yang digunakan untuk mencapai akses data yang efisien dan beberapa penyesuaian keutuhan yang mendesak tingkat keamanan.

5. Pemilihan DBMS (DBMS *Selection)*

Pemilihan DBMS adalah pemilihan DBMS yang sesuai untuk mendukung aplikasi basis data (dilakukan pada tahap konsetual dan logikal).

Pada tahap ini dilakukan seleksi DBMS yang cocok untuk mendukung aplikasi basis data. Berikut ini adalah tahapan utama untuk menyeleksi basis data adalah:

a. Menggambarkan cakupan tugas berdasarkan kebutuhan perusahaan
b. Membuat perbandingan mengenai dua atau tiga produk
c. Mengevaluasi produk-produk DBMS yang dipilih
d. Merekomendasikan pemilihan DBMS dan membuat laporan hasil evaluasi produk-produk DBMS tersebut

6. Perancangan aplikasi (*Application Design)*

Perancangan aplikasi adalah perancangan antarmuka pengguna dan program aplikasi yang menggunakan dan memproses basis data.

Dalam perancangan aplikasi basis data, perlu adanya pemastian bahwa semua fungsi yang dinyatakan dalam spesifikasi kebutuhan pengguna tersedia dalam perancangan aplikasi untuk aplikasi basis data. Hal ini melibatkan perancangan program aplikasi yang mengakses basis data dan perancangan desain transaksi (metode pengaksesan basis data).

7. *Prototyping*

P*rototyping* adalah pembuatan model kerja yang tidak dimiliki semua fitur yang diperlukan atau menyediakan semua fungsi dari sistem akhir.

Tujuan dari pengembangan *prototyping* pada aplikasi basis data adalah untuk memungkinkan pengguna menggunakan *prototyping* untuk mengenal fitur-fitur sistem yang berkerja dengan baik atau yang kurang baik.

Membangun sebuah model kerja dari aplikasi basis data yang mengijinkan perancang atau pengguna untuk memvisualisasikan dan mengevaluasi bagaimana gambaran sistem secara keseluruhan dan fungsinya. Ada dua strategi *prototyping*, yaitu:

a. *Requirement Prototyping* (Prototipe yang Dibutuhkan)

Menggunakan *prototyping* untuk menetapkan kebutuhan dari aplikasi basis data yang diusulkan setelah kebutuhan terpenuhi, *prototyping* dibuang.

b. *Evolutionary Prototyping* (Solusi Prototipe)

Digunakan untuk tujuan yang sama, tetapi *prototyping*-nya tetap digunakan kembali dan dengan pengembangan yang lebih jauh *prototyping*-nya menjadi aplikasi basis data yang berjalan.

8. Implementasi (*Implementation)*

Implementasi adalah perwujudan fisik dari perancangan basis data dan aplikasi.

Implementasi basis data dapat dicapai dengan menggunakan *Data Definition Language* (DDL) dari DBMS yang dipilih atau *Graphical User Interface* (GUI) yang menyediakan fungsi yang sama.

Program aplikasi diimplementasikan menggunakan bahasa generalisasi ketiga atau keempat (3GL atau 4GL) yang lebih disukai. Bagian dari program aplikasi ini adalah transaksi basis data yang diimplementasikan dengan menggunakan *Data Manipulation*

Language (DML) yang mungkin dilekatkan ke dalam bahasa pemograman Visual Basic, C++, Java, dan lainnya.

9. Konversi Data dan Pemuatan (*Data Conversion and Loading)*

   Konversi data dan pemuatan adalah memindahkan data yang ada ke dalam basis data yang baru dan mengubah aplikasi yang ada agar dapat berjalan pada basis data yang baru.

   Tahap ini digunakan hanya pada saat sistem basis data baru menggantikan sistem lama. DBMS memiliki fungsi yang memuat *file* yang ada ke dalam basis data baru. Fungsi tersebut biasanya memerlukan spesifikasi *file source* dan target basis data, dan kemudian mengubah data ke format yang diperlukan dari *file* basis data baru.

10. Pengujian (*Testing)*

    Testing adalah proses mengeksekusi program aplikasi dengan maksud untuk menemukan kesalahan.

    Testing pada sistem aplikasi basis data yang baru dibuat harus dilakukan sebelum digunakan dan diserahkan kepada pengguna. Jika *testing* menunjukkan keberhasilan, maka pengujian akan menemukan kesalahan pada program aplikasi.

Seperti dengan perancangan basis data, pengguna dari sistem baru *testing* adalah menguji basis data pada sistem *hardware* yang berbeda. Jika data asli digunakan, maka perlu dibuat *backup* untuk mencegah kesalahan.

11. Pemeliharaan Operasional (*Operational Maintenance*)

Pemeliharaan operasional adalah proses pengawasan dan pemeliharaan sistem setelah instalasi.

Tahap pemeliharaan melibatkan aktivitas sebagai berikut:

a. Mengawasi kerja sistem

Jika kinerja menurun ke bawah tingkatan yang ditetapkan, maka pengaturan ulang basis data perlu dilakukan.

b. Memelihara dan meng-*upgrade* aplikasi basis data ketika diperlukan

Biasanya DBMS menyediakan berbagai kegunaan untuk membantu administrasi basis data seperti mengisi data ke dalam basis data dan mengawasi sistem.

2.1.6 Normalisasi

Menurut Connolly (2002, p376), normalisasi adalah sebuah teknik untuk menghasilkan sebuah kumpulan dari relasi-relasi dengan atribut-

atribut yang diinginkan, yang berdasarkan kebutuhan-kebutuhan data sebuah perusahaan.

Tahapan normalisasi menurut Connolly (2002, p386), adalah sebagai berikut:

1. *Unnormalized* (UNF)

   Sebuah tabel yang berisi satu atau lebih grup yang berulang. Yang dimaksud grup yang berulang itu adalah atribut-atribut yang *multivalued.*

2. Normalisasi pertama (1NF)

   Sebuah relasi dimana setiap baris dan kolom hanya berisi satu nilai saja. Bentuk normal pertama ini, dicapai apabila setiap nilai atribut adalah tunggal. Kondisi ini dapat diperoleh dengan melakukan eliminasi terhadap terjadinya data ganda (*repeating groups*). Pada kondisi normal pertama ini kemungkinan masih terjadi adanya data rangkap.

3. Normalisasi kedua (2NF)

   Suatu relasi yang terdapat dalam 1NF dan setiap atribut yang bukan merupakan *primary key* bergantung sepenuhnya secara fungsional terhadap *primary key*.

4. Normalisasi ketiga (3NF)

Sebuah relasi dalam bentuk normal pertama dan kedua serta setiap atribut bukan *key* yang bergantung secara transitif kepada bukan *key* juga. Bentuk normal ketiga adalah berdasarkan pada konsep peralihan ketergantungan (*transitive dependency*).

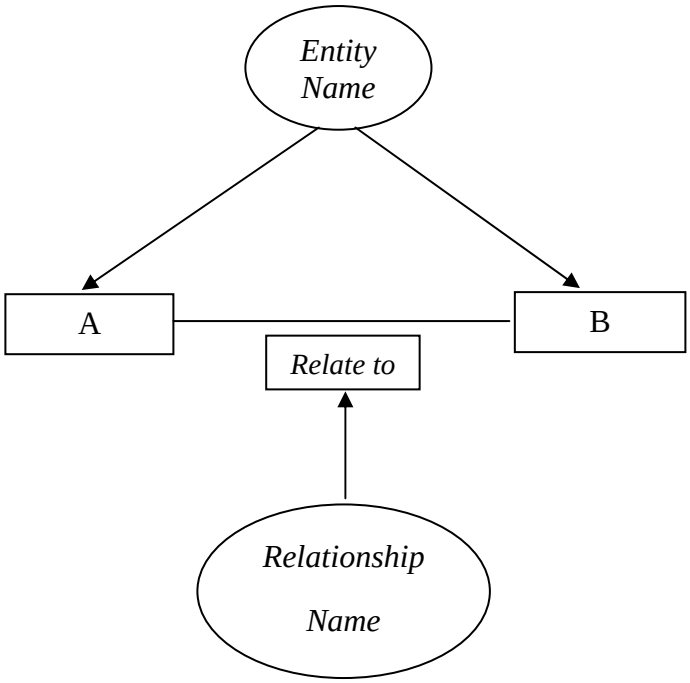
2.1.7 *Entity Relationship*

Menurut Connolly (2002, p330), salah satu bagian yang sulit dalam perancangan basis data adalah suatu fakta bahwa para perancang, pembuat program, dan *end-user* cenderung untuk melihat data dan menggunakannya dengan cara-cara yang berbeda, kecuali diperoleh sebuah pemahaman sama yang mencerminkan bagaimana suatu perusahaan beroperasi, suatu perancangan yang dihasilkan akan gagal untuk memenuhi kebutuhan-kebutuhan *user*. Untuk menyakinkan bahwa didapatkan sebuah pemahaman yang tepat dari suatu data dan bagaimana data tersebut digunakan oleh suatu perusahaan, sudah seharusnya mempunyai sebuah model untuk membuat komunikasi yang non teknikal dan tidak bersifat ambigu, *Entity relationship* (ER) adalah salah satu contohnya.

Pemodelan *entity relationship* adalah sebuah pendekatan *top-down* untuk perancangan basis data yang dimulai dengan mengidentifikasi suatu data penting yang disebut entiti-entiti dan hubungan diantara suatu data yang harus dipresentasikan dalam suatu model. Lalu tambahkan

perincian-perincian lagi seperti suatu informasi yang ingin diambil tentang suatu entiti-entiti dan hubungannya yang disebut atribut-atribut dan batasan-batasan yang lain pada suatu entiti-entiti *relationship*, atribut-atribut.

Berikut ini adalah notasi *Entity Relationship Modelling* menurut Connolly (2002, p333) :



Gambar 2.2 Notasi *Entity Relationship Modelling*

Pengertian *Multiplicity* adalah sejumlah kemungkinan kejadian-kejadian dari sebuah tipe entiti di dalam sebuah hubungan *n-nary* ketika nilai-nilai yang lain (n-1) ditentukan. *Multiplicity* biasanya terdiri dari dua batasan terpisah, yaitu:

1. *Cardinality*

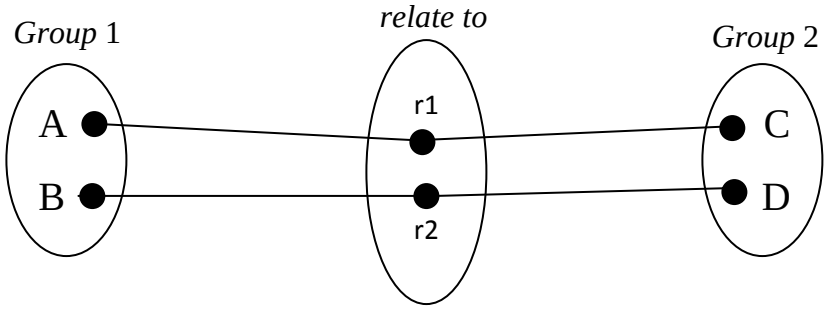
   Hal ini mendeskripsikan jumlah maksimum dari kemungkinan kejadian-kejadian yang saling berhubungan untuk sebuah partisipasi entiti dalam proses penentuan tipe *relationship*.

2. *Participations*

   Dalam hal ini menentukan apakah semua kejadian kejadian entiti akan ikut berpartisipasi dalam sebuah *relationship* atau hanya beberapa saja yang ikut berpartisipasi.

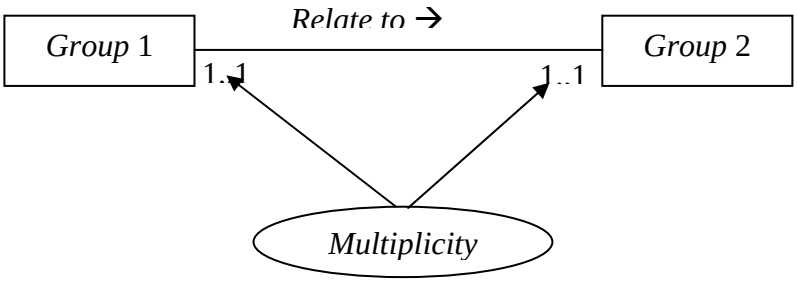
Jenis-jenis *multiplicity* adalah:

1. *One to one* (1:1) *Relationship*



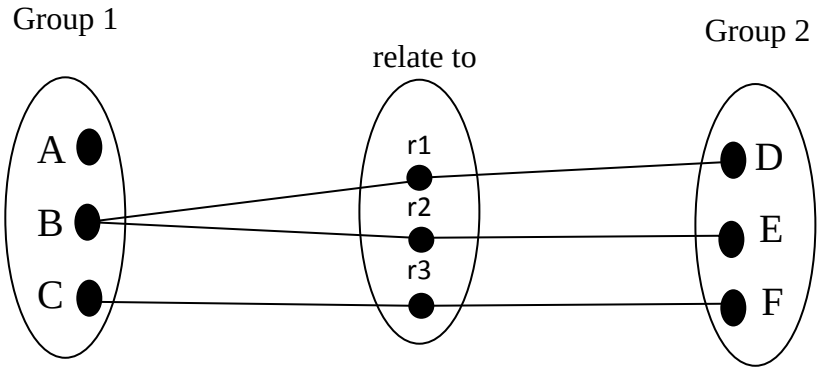
Gambar 2.3 *One-to-One Relationship*

Pada gambar 2.3, dapat dilihat bahwa A hanya terhubung *One-to-One* (1:1) dengan C, dan B hanya terhubung *One-to-One* (1:1) dengan D. Jadi dari gambar tersebut dapat dituliskan notasi *multiplicity*-nya dengan gambar di bawah ini.



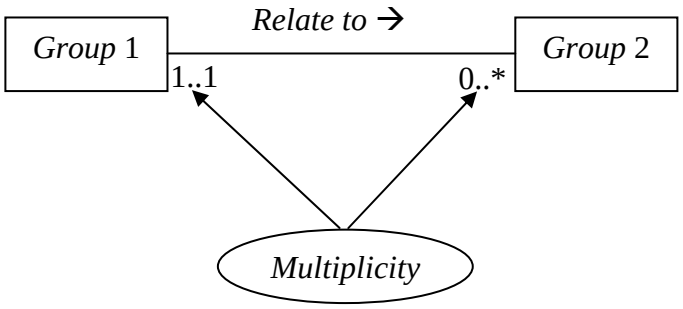
Gambar 2.4 Notasi *One-to-One Relationships*

2. *One-to-Many* (1:*) *Relationships*



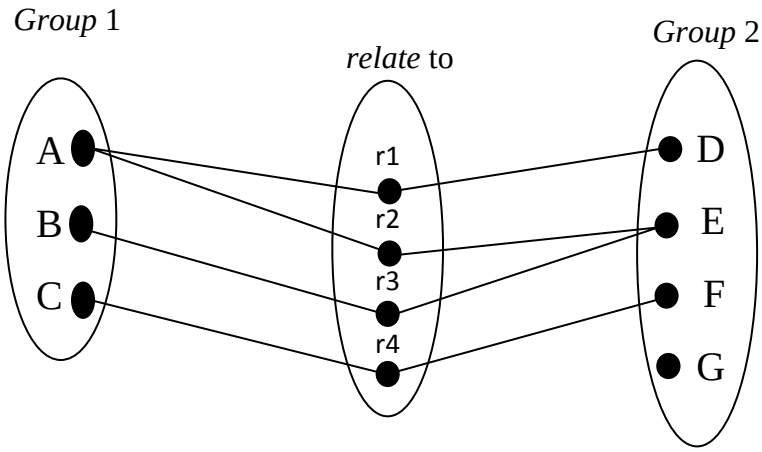
Gambar 2.5 Notasi *One-to-Many Relationships*

Pada gambar 2.5, dapat dilihat bahwa B terhubung *One-to-Many* (1:*) dengan D dan E. Jadi dari gambar tersebut dapat dituliskan notasi *multiplicity*-nya dengan gambar di bawah ini.



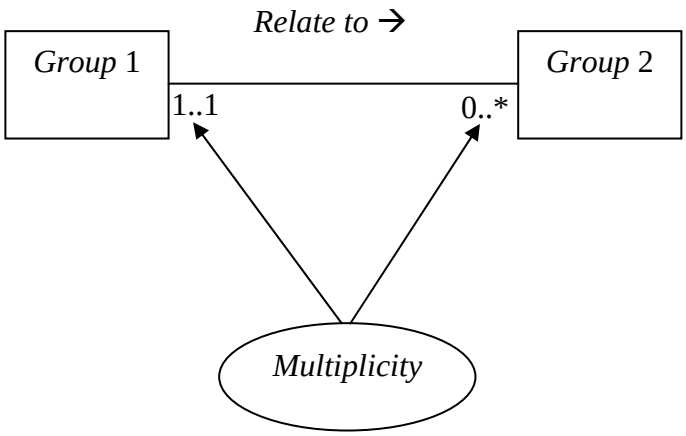
Gambar 2.6 Notasi *One-to-Many Relationships*

3. *Many-to-Many* (*:*) *Relationships*



Gambar 2.7 *Many-to-Many Relationships*

Pada gambar 2.7, dapat dilihat bahwa A terhubung *One-to-Many* (1:*) dengan D dan E. Sedangkan E terhubung *One-to-Many* (1:*) dengan A dan B. Jadi dari entiti Group 1 (*value*-nya A dari gambar di atas) dan Group 2 (*value*-nya E dari gambar di atas) terhubung *Many-to-Many* (*:*). Dari gambar tersebut dapat dituliskan notasi *multiplicity*-nya dengan gambar di bawah ini.



Gambar 2.8 Notasi *Many-to-Many Relationships*

2.1.8 Perancangan Basis Data Konseptual

Proses mengkontruksi atau membangun model informasi yang digunakan dalam sebuah *enterprise*, terbebas dari semua pertimbangan fisik. Pada konseptual basis data desain ini terdiri dari langkah-langkah seperti:

Langkah pertama: Membangun data model konseptual lokal untuk setiap pandangan, tahap-tahapnya sebagai berikut:

a. Mengidentifikasi tipe entiti

Pada tahapan ini dilakukan identifikasi tipe entiti utama yang diperlukan oleh sebuah *view*. Salah satu metode dalam mengidentifikasi tipe entiti adalah dengan memeriksa kebutuhan user. Dari kebutuhan user ini, kita dapat mengidentifikasi kata benda atau frase yang ada, selain itu kita bisa mendapatkan objek utama seperti orang dan tempat.

b. Mengidentifikasi tipe relasi

Pada tahap ini dilakukan identifikasi relasi penting yang ada antara tipe entiti yang telah teridentifikasi. Ketika mengidentifikasi relasi kita dapat menggunakan kata kerja dalam spesifikasi kebutuhan user.

Adapun langkah-langkah dalam mengidentifikasi tipe relasi adalah sebagai berikut:

1. Gunakan *Entity Relationship Diagram* (ERD)
2. Menentukan pembatas *multiplicity* dari tipe relasi

Dengan adanya kebutuhan user ini, membuktikan bahwa relasi-relasi yang terjadi adalah penting dan harus dimasukkan dalam model.

c. Mengidentifikasi dan menghubungkan atribut entiti atau tipe relasi

Tujuannya adalah untuk menghubungkan atribut dengan entiti atau tipe relasi. Atribut yang dimiliki setiap entiti atau *relationship* memiliki identitas atau karakteristik yang sesuai dengan memperhatikan atribut berikut : *simple/composite attribute*, *single valued attribute*, dan *derived attribute*.

d. Menentukan domain atribut

Domain adalah sebuah penampung dari nilai yang dapat ditampung oleh atribut, misalnya domain atribut dari nomor *staff* hanya dapat menggunakan lima karakter *string* dengan dua karakter pertama adalah huruf dan tiga karakter berikutnya adalah angka.

e. Menentukan *candidate* dan *primary key* dari atribut

Candidate Key adalah set atribut minimal dari sebuah entiti yang secara unik mengidentifikasi setiap kemunculan dari entiti tersebut. *Candidate Key* dapat diidentifikasi lebih dari satu, tetapi harus dipilih satu sebagai *primary key*, sedangkan *candidate key* yang lain disebut *alternate key*.

Berikut adalah acuan dalam menentukan *primary key* dari *candidate key*:

1. *Candidate key* dengan set atribut yang minimal
2. *Candidate key* dengan nilai yang berubah paling sedikit
3. *Candidate key* dengan karakter paling sedikit
4. *Candidate key* dengan isi maksimum yang paling sedikit
5. *Candidate key* yang termudah digunakan dari sudut pandang *user*

f. Mempertimbangkan penggunaan konsep model yang lebih tinggi (*enchanced modeling concepts*) / *optional*

Tahap ini bersifat *optional*, apakah akan digunakan pengembangan dari entiti model dengan menggunakan *enchanced*

modeling concepts, seperti *specialization, generalization, aggregation* atau *composition*.

g. Memeriksa redudansi pada model

Tahapan ini memeriksa model data konseptual lokal apakah terjadi duplikasi atau tidak dengan dua langkah, yaitu:

1. Memeriksa ulang relasi *one-to-one*
2. Menghilangkan relasi yang terduplikasi (*redudant*)

h. Memvalidasi model data konseptual lokal terhadap transaksi *user*

Memeriksa model yang telah dihasilkan apakah mendukung transaksi pada *view*. Pemeriksaan ini dapat menggunakan dua langkah, yaitu:

1. Mendeskripsikan transaksi

Dengan pendekatan ini, dicek bahwa semua informasi (*entity, relationship* dan *attribute*-nya) dibutuhkan oleh masing-masing transaksi yang disediakan oleh model, dokumentasi sebuah deskripsi dari masing-masing kebutuhan transaksi.

2. Menggunakan jalur transaksi

Pada pendekatan yang kedua ini, digunakan untuk memvalidasi model data melawan transaksi yang dibutuhkan yang melibatkan representasi alur secara diagramatik yang diambil oleh masing-masing transaksi secara langsung pada ERD.

i. Memeriksa model data konseptual lokal dengan *user*

Memeriksa model data konseptual lokal termasuk *entity relationship*, jika terjadi ketidakcocokan (*anomaly*) maka harus dilakukan perubahan.

2.1.9 Perancangan Basis Data Logical

Proses membangun model informasi yang digunakan dalam sebuah *enterprise* yang didasarkan pada data model spesifik, dan terbebas dari DBMS dan semua pertimbangan fisik.

Pada desain logikal basis data terdiri dari langkah-langkah sebagai berikut:

Langkah kedua: Membangun dan memvalidasi model data logikal lokal untuk setiap *view*.

Dari model data konseptual lokal yang telah dibangun pada tahapan pertama akan diubah menjadi model data logikal lokal yang

terdiri dari *entity relationship diagram*, sebuah *relationship schema* dan dokumentasi-dokumentasi pendukung, yaitu:

a. Menghilangkan hal-hal yang tidak sesuai dengan model relational (langkah *optional*)

   Model data konseptual lokal yang telah ada dapat mengandung struktur yang tidak dapat dimodelkan oleh DBMS konvensional, oleh karena itu pada tahap ini dilakukan perubahan menjadi bentuk yang lebih mudah ditangani oleh sistem ini.

   Langkah-langkah yang dapat dilakukan dalam penyesuaian struktur adalah:

   1. Menghilangkan relasi binari *many to many* (*:*)
   2. Menghilangkan relasi rekursif *one to one* (1:1)
   3. Menghilangkan relasi rekursif *one to many* (1:*)
   4. Menghilangkan relasi rekursif *many to many* (*:*)
   5. Menghilangkan tipe relasi yang kompleks
   6. Menghilangkan atribut *multi-valued*

b. Mendapatkan relasi untuk lokasi model data logikal

   Membentuk relasi dari model data logikal lokal untuk merepresentasikan relasi antar entiti dan atribut yang telah

didefinisikan. Untuk mendapatkan relasi dari data model yang ada maka digunakan cara-cara berikut ini:

1. Tipe entiti yang kuat (*Strong Entity*)
2. Tipe entiti yang lemah (*Weak Entity*)
3. Tipe Relasi binari *one to many* (1:*)
4. Tipe Relasi binari *one to one* (1:1)
5. Tipe Relasi rekursif *one to one* (1:1)
6. Tipe Relasi rekursif *one to many* (1:*)
7. Tipe Relasi *superclass / subclass*
8. Tipe Relasi binary *Many-to-Many*
9. Tipe Relasi kompleks
10. *Attribute multivalue*

c. Memvalidasi relasi menggunakan normalisasi

Normalisasi digunakan untuk meningkatkan model yang telah terbentuk agar duplikasi data yang tidak diperlukan dapat dihindari.

Langkah utama dari proses normalisasi yaitu sebagai berikut:

1. Bentuk tidak normal (UNF)
2. Bentuk normal pertama (1NF), menghilangkan kelompok perulangan (*repeating*)

3. Bentuk normal kedua (2NF), menghilangkan ketergantungan sebagian (*partial dependencies*) pada *primary key*
4. Bentuk normal ketiga (3NF), menghilangkan ketergantungan transitif (*transitive dependencies*) pada *primary key*

d. Memvalidasi relasi dengan transaksi pemakai

Tujuannya adalah untuk memastikan bahwa relasi di dalam logikal data model mendukung transaksi yang diminta pemakai. Pada langkah ini, pengecekan bahwa relasi yang dibuat di langkah sebelumnya juga mendukung transaksi ini, dan juga pastikan bahwa tidak ada ekor dalam relasi yang telah dibuat.

e. Mendefinisikan *integrity constraints*

Integrity constraints adalah batasan yang digunakan untuk menjaga agar basis data tidak menjadi tidak konsisten. Ada lima tipe *integrity constraints*, yaitu :

1. *Required data* (data / nilai yang valid)
2. Batasan *domain atribut*
3. *Entity integrity* (*primary key* tidak boleh null)

4. *Referential integrity* (*foreign key* pada suatu entiti harus sesuai dengan *candidate key* pada entiti lain)
5. *Enterprise constraints* (batasan pada organisasi)

f. Memeriksa model data logikal lokal dengan *user*

Memastikan model data logikal lokal yang terbentuk merupakan representasi dari *user view*. Untuk memvalidasi model data logikal ini digunakan *Data Flow Diagram* (DFD). DFD dapat menunjukkan aliran data dari suatu organisasi.

Langkah ketiga: Membangun dan memvalidasi model data logikal global.

Pada tahap ini, digabungkan semua model data logikal lokal menjadi sebuah model data logikal global yang mempresentasikan organisasi tersebut.

a. Menggabungkan model data logikal lokal ke dalam model global

Menggabungkan model data logikal individual kedalam model data logikal global organisasi.

Beberapa tugas dari pendekatan ini adalah sebagai berikut:

1. Me-*review* nama dan isi entiti atau relasi serta *candidate key*-nya

2. Me-*review* nama dan isi dari *relationship* atau *foreign key*
3. Menggabungkan entiti atau relasi dari lokal data model
4. Meliputi (kecuali penggabungan) entiti atau relasi yang unik ke masing-masing lokal data model
5. Menggabungkan *relationship* atau *foreign key* dari lokal data model
6. Meliputi (kecuali penggabungan) *relationship* atau *foreign key* yang unik ke masing-masing lokal data model
7. Mengecek apakah ada kehilangan entiti atau relasi dan *relationship* atau *foreign key*
8. Mengecek *foreign key*
9. Mengecek *Integrity constraints*
10. Memperbaharui dokumentasi

b. Memvalidasi model data logikal global

Memvalidasi relasi yang telah dibuat dari model data global menggunakan teknik normalisasi dan memastikan relasi ini mendukung transaksi yang diperlukan. Langkah ini sama dengan langkah 3 dan 4 yang memvalidasi setiap model data logikal lokal.

c. Memeriksa perkembangan yang akan datang

Memastikan apakah ada perubahan yang signifikan yang dapat diperkirakan dan memastikan apakah model data logikal global ini dapat mendukung perubahan-perubahan ini.

Hal ini penting bahwa global logikal data model dapat diperluas dengan mudah. Jika model dapat menopang kebutuhan yang sekarang saja, maka kehidupan model tersebut mungkin relatif pendek dan pengerjaan yang signifikan mungkin dibutuhkan untuk mengakomodasikan kebutuhan yang baru. Hal ini penting untuk mengembangkan model yang dapat diperluas dan mempunyai kemampuan yang dapat meningkatkan untuk mendukung kebutuhan yang baru dengan efek yang minimal pada pemakai yang ada.

d. Memeriksa model data logikal global dengan *user*

Model dan dokumentasi yang menjelaskan model seharusnya di-*review* dengan pemakainya untuk memastikan bahwa ini adalah representasi perusahaan yang benar.

2.1.10 Perancangan Basis Data Fisikal

Proses memproduksi sebuah deskripsi dari implementasi basis data dalam *secondary storage*, yang menjelaskan relasi dasar, organisasi *file*, dan membuat *index* untuk mendapatkan akses yang efisien ke data, serta

setiap *integrity constraints* yang saling berhubungan dan juga pengukuran keamanan.

Pada desain fisikal basis data ini terdiri dari langkah-langkah sebagai berikut:

Langkah keempat: Menerjemahkan model data logikal global untuk DBMS yang digunakan.

Pada tahap ini akan dihasilkan suatu skema basis data relasional dari model data logikal global yang dapat diimplementasikan ke dalam DBMS yang akan digunakan.

a. Mendesain relasi dasar (*base relations*)

Tujuan dari langkah ini adalah untuk memutuskan bagaimana merepresentasikan relasional dasar yang didefinisikan dalam logikal data model global pada DBMS yang dipakai.

Untuk memulai proses perancangan basis data fisik, pertama-tama harus mengumpulkan dan mengasimilasi suatu informasi tentang relasional yang dirancang selama perancangan basis data logikal. Informasi yang diperlukan bisa berasal dari kamus data dan definisi dari relasional yang didefinisikan menggunakan *Database Design Language* (DBDL). Untuk setiap relasional yang didefinisikan pada global logikal data model, dapat didefinisikan salah satu sebagai berikut:

1. Nama dari relasional yang ada
2. Suatu *list* untuk atribut yang sederhana
3. *Primary key*, *alternate key*, dan *foreign key*
4. Suatu daftar dari atribut turunan dan bagaimana pembuatannya
5. *Integrity constraints* untuk setiap *foreign key* yang didefinisikan

Dari kamus data, setiap atributnya dapat diketahui:

1. Domain atribut tersebut yang terdiri dari tipe data, dan berbagai keterangan atribut tersebut
2. Suatu optional nilai *default* untuk atribut
3. Apakah atribut dapat diisi nilai *null*

b. Mendesain representasi dari data yang diturunkan

Tujuan dari langkah ini adalah untuk memutuskan bagaimana merepresentasikan suatu data turunan pada model data logikal global pada DBMS yang dipakai. Atribut yang nilainya didapatkan dengan mengevaluasi atribut lain dikenal sebagai atribut turunan atau kalkulasi, sebagai contoh:

1. Jumlah banyaknya *staff* yang bekerja pada suatu kantor
2. Total gaji yang dibayarkan ke seluruh *staff*
3. Total dari *property* yang ditangani oleh seorang *staff*

c. Mendesain *Entreprise Constraints*

Meng-*update* suatu relasi yang mungkin dibatasi oleh aturan perusahaan sesuai dengan transaksi yang sebenarnya bisa di-*update*. Perancangan batasan tersebut sekali lagi tergantung pada DBMS yang digunakan, fasilitas yang dipunyai oleh sistem dibandingkan dengan DBMS yang lain. Pada awalnya, jika sistem tersebut mempunyai aturan sesuai dengan aturan standar SQL, beberapa batasan dapat diterapkan.

Langkah kelima: Mendesain gambaran fisik dari basis data

Menentukan organisasi *file* yang akan digunakan dan *index* untuk menghasilkan performa yang diinginkan serta menentukan apa saja yang akan disimpan dalam *secondary storage*.

a. Menganalisis transaksi

Tujuan langkah ini adalah untuk mengerti fungsi dari suatu transaksi yang mana akan dijalankan pada basis data dan untuk menganalisis transaksi yang penting. Untuk membuat perancangan basis data fisikal yang efisien, maka perlu untuk mempunyai pengetahuan mengenai transaksi atau *query* yang akan dijalankan di dalam basis data ini. Ini termasuk kuantitatif

atau kualitatif informasi. Dalam menganalisa transaksi, dapat diidentifikasi kriteria performa sebagai berikut:

1. Transaksi yang sering digunakan dan akan berdampak besar terhadap performansi keseluruhan
2. Transaksi yang merupakan transaksi bisnis yang kritis
3. Durasi waktu dalam harian atau mingguan yang akan mendapatkan banyak permintaan pada *database*

b. Memilih organisasi *file* yang akan digunakan

Dalam banyak kasus yang ada, suatu relasional DBMS akan memberikan sedikit bahkan tanpa pilihan dalam memilih organisasi *file*, walaupun beberapa akan mempunyai *index* yang spesifik.

Bagaimanapun, berikut ini beberapa organisasi *file* yang ada adalah sebagai berikut:

1) *Heap*
2) *Hash*
3) *Indexed Sequential Access Method (ISAM)*
4) *B-tree*
5) *Chister*

c. Memilih *index* yang digunakan

Tujuannya adalah untuk menentukan penambahan *index* yang akan meningkatkan performansi dari suatu sistem. Biasanya atribut untuk *index* adalah sebagai berikut:

1. Suatu atribut yang digunakan paling sering untuk operasi penggabungan, yang akan membuat penggabungan tersebut lebih efektif.
2. Suatu atribut yang digunakan paling banyak untuk mengakses suatu *record* di dalam relasi yang ada.

d. Memperkirakan *disk space* yang diperlukan

Tujuannya adalah untuk mengestimasi ukuran kapasitas *disk* yang diperlukan untuk basis data. Sasaran dari langkah ini adalah menafsirkan jumlah ruang *disk* yang diperlukan untuk mendukung implementasi basis data pada tempat penyimpanan sekunder.

Langkah keenam: Desain tampilan pemakai

Tujuan dari langkah ini adalah merancang tampilan pemakai yang akan diidentifikasi selama pengumpulan informasi dan analisis dari siklus hidup aplikasi basis data.

Langkah ketujuh: Mendesain pengukuran keamanan (*security*)

Tujuan dari langkah ini adalah untuk merancang ukuran keamanan untuk basis data yang telah dispesifikasi pemakai. Definisi keamanan basis data adalah suatu mekanisme yang melindungi basis data dari suatu kejadian yang disengaja maupun yang tidak disengaja.

Suatu basis data merupakan sumber dari perusahaan yang *essensial* yang perlu dilindungi dengan menggunakan suatu kontrol yang memadai. Beberapa *issue* keamanan basis data yang perlu diperhatikan adalah sebagai berikut:

a. Pencurian data (*Thieft and Fraud*)
b. Kehilangan kerahasiaan suatu data (*loss of confidentially*)
c. Kehilangan hal pribadi (*loss of privacy*)
d. Kehilangan integritas (*loss of integrity*)
e. Kehilangan ketersediaan data (*loss of availability*)

Langkah kedelapan: Menentukan apakah redudansi data telah dapat dikontrol

Dilakukan normalisasi agar dapat meningkatkan performa dari sistem dan menghilangkan redudansi.

Langkah kesembilan: Memonitor sistem operasional

Memonitor dan meningkatkan performa dari sistem dengan memperbaiki desain yang tidak sesuai atau perubahan kebutuhan.

Hasil akhir dari perancangan fisik basis data adalah suatu proses yang mendeskripsikan suatu implementasi dari suatu basis data pada media penyimpanan. Ini mendeskripsikan suatu relasional dan struktur penyimpanan dan metodologi pengaksesan data oleh pemakai yang efisien, selama batasan integriti dan pengukuran keamanan.

2.2 Teori – Teori Khusus Berhubungan Dengan Topik

2.2.1 Manajemen Proyek

Menurut Berkun (2005, p2), manajemen proyek dapat berupa sebuah pekerjaan, sebuah peran yang terdapat di sekitar lingkungan kita dan dalam jangka waktu yang lama.

Manajemen proyek adalah penerapan dari pengetahuan, keterampilan, *tools* dan *techniques* pada aktivitas-aktivitas proyek supaya persyaratan dan kebutuhan dari proyek terpenuhi. Proses-proses dari manajemen proyek dapat dikelompokkan dalam lima kelompok yaitu inisialisasi proses, merancang proses, menjalankan proses, mengontrol proses, menutup proses.

Karakteristik proyek antara lain:

1. Kegiatannya dibatasi oleh waktu, sifatnya sementara, diketahui kapan mulai dan berakhirnya
2. Dibatasi oleh biaya / *budget*
3. Dibatasi oleh kualitas
4. Biasanya tidak berulang-ulang

Di bawah ini ada beberapa macam proyek, yang dapat dikelola dengan sebuah manajemen proyek:

1. Pembuatan rumah
2. Pembuatan jalan raya
3. Pembuatan jembatan
4. Pembuatan iklan perusahaan
5. Pembentukan tim evaluasi lokasi baru perusahaan
6. Pembuatan *prototyping* produk baru

2.2.2 *Visual Basic*

Microsoft Visual Basic (biasa disebut VB) merupakan sebuah bahasa pemrograman yang bersifat *event driven* dan menawarkan *Integrated Development Environment* (IDE) visual untuk membuat program aplikasi berbasis sistem operasi *microsoft windows* dengan menggunakan model pemrograman *Common Object Model* (COM). *Visual Basic* merupakan turunan bahasa *basic* dan menawarkan pengembangan aplikasi komputer

berbasis grafik dengan cepat, akses ke basis data menggunakan *Data Access Objects* (DAO), *Remote Data Objects* (RDO), atau *ActiveX Data Object* (ADO), serta menawarkan pembuatan kontrol *ActiveX* dan objek *ActiveX*. Beberapa bahasa skrip seperti *Visual Basic for Applications* (VBA) dan *Visual Basic Scripting Edition* (VBScript), mirip dengan *Visual Basic*, tetapi cara kerjanya yang berbeda.

Para *programmer* dapat membangun aplikasi dengan menggunakan komponen-komponen yang disediakan oleh *Microsoft Visual Basic*. Program-program yang ditulis dengan *Visual Basic* juga dapat menggunakan *Windows* API, tetapi membutuhkan deklarasi fungsi eksternal tambahan.

Dalam pemrograman untuk bisnis, *Visual Basic* memiliki pangsa pasar yang sangat luas. Dengan menggunakan konsep *object oriented programming*(OOP) dan desain *form* secara visual serta adanya kemampuan untuk menggunakan komponen-komponen *ActiveX* yang dibuat oleh pihak lain, *Visual Basic* banyak digunakan oleh para *programmer* untuk kepentingan bisnis.