BAB 2

LANDASAN TEORI

2.1 Pengertian *Database*

Database adalah sebuah tempat penyimpanan yang besar dimana terdapat kumpulan data yang tidak hanya berisi data operasional tetapi juga deskripsi data. Seperti yang disampaikan oleh Connolly dan Begg (2010, p.65), bahwa *database* adalah kumpulan data yang saling terhubung secara logis dan deskripsi dari data tersebut, dirancang untuk menemukan informasi yang dibutuhkan oleh sebuah organisasi. Dalam merancang *database,* salah satu hal yang perlu diperhatikan adalah efisiensi. Banyaknya data yang redudansi dapat mengurangai efisiensi pada *database* sehingga perlu dilakukan normalisasi. *Database* ini digunakan tidak hanya oleh satu orang maupun satu departemen, *database* dapat digunakan oleh seluruh departemen dalam perusahaan. *Database* ini akan menjadi sumber data yang digunakan secara bersama dalm perusahaan. Hal ini kembali ditegaskan oleh Connolly dan Begg (2010, p.65), *database* tidak lagi dimiliki oleh satu departemen tetapi sumber perusahaan yang saling berbagi. Untuk mendapatkan *database* ,Dengan hanya *database* saja tidak cukup, diperlukan *Database Management System* (DBMS) untuk dapat menggunakan *database*.

2.1.1 Data

Data merupakan catatan atas kumpulan fakta yang dicatat dan disimpan oleh perusahaan untuk diolah sehingga dapat menghasilkan suatu informasi yang berguna bagi perusahaan itu sendiri. Data yang dikumpulkan di dalam *database* akan digunakan oleh pengguna akhir untuk mendapatkan informasi. Seperti pendapat yang diungkapkan oleh Laudon, (2007, p.375) data merupakan kumpulan fakta – fakta

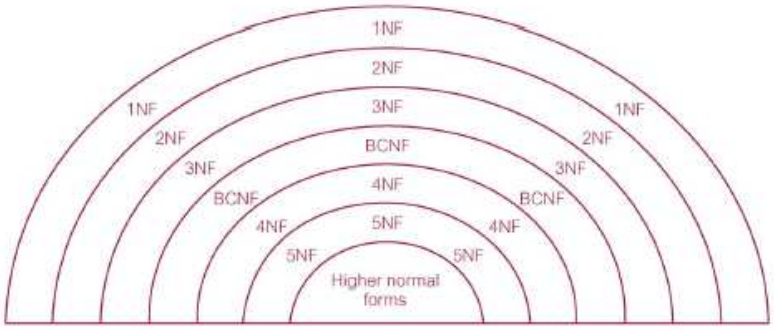
kasar yang menunjukkan kejadian yang terjadi dalam organisasi atau lingkungan fisik sebelum fakta tersebut diolah dan ditata menjadi bentuk yang dapat dipahami.

2.1.2 Normalisasi

Normalisasi adalah suatu teknik dimana digunakan untuk mengidentifikasikan hubungan antara atribut dengan memberikan kebutuhan data yang diperlukan oleh suatu perusahaan. Seperti yang disampaikan oleh Connolly dan Begg, (2010, p.416) normalisasi merupakan sebuah teknik untuk memproduksi satu set hubungan dengan sifat yang diinginkan, memberikan kebutuhan data pada perusahaan.

Proses Normalisasi antara lain :

- Suatu teknik formal untuk menganalisis relasi berdasarkan *primary key* dan fungsi dependensi antar atribut yang ada.
- Dieksekusi dalam beberapa cara. Setiap cara mengacu ke bentuk normal tertentu, sesuai dengan sifat yang dimilikinya.
- Setelah Normalisasi diproses, relasi akan secara bertahap lebih terbatas/kuat bentuk formatnya dan juga mengurangi tindakan anomali pada setiap *update*.



Gambar 2.1 Diagram ilustrasi dari hubungan antara bentuk normalisasi

Sumber : Connolly dan Begg (2010, p.429)

Bentuk-bentuk normalisasi menurut Connolly dan Begg, (2010, p.430-438) antara lain:

1. *Unnormalized Form* (UNF)

Merupakan sebuah tabel awal yang belum ternormalisasi yang berisikan satu atau lebih kumpulan data yang berulang. Untuk membuat tabel UNF yaitu dengan memindahkan data dari sumber informasi yang di dapat ke dalam tabel dengan format baris dan kolom, jika ada atribut yang mempunyai banyak nilai (*multivalue*) akan masuk ke dalam bentuk UNF.

2. *First Normal Form* (1NF)

Bentuk normalisasi tahap pertama yang merupakan sebuah relasi dimana sebuah titik pertemuan antara setiap baris dan kolom yang berisi satu dan hanya satu nilai.

3. *Second Normal Form* (2NF)

Tahapan kedua setelah 1NF terpenuhi yaitu 2NF dimana merupakan sebuah relasi yang terdapat di dalam 1NF dan setiap atribut yang bukan *primary key* bergantung pada *primary key*.

4. *Third Normal Form* (3NF)

Merupakan tahapan ketiga dalam normalisasi dimana sebuah relasi yang terdapat pada bentuk normalisasi pertama dan kedua, yang mana atribut *primary key* bergantung pada *primary key.*

Proses normalisasi meliputi langkah-langkah sebagai berikut :

a) *Unnormalized Normal Form* (UNF)

Ada tahap awal sebelum memulai 1NF yaitu *unnormalized form* (UNF). Pada bentuk UNF ini merupakan tabel yang terdapat satu atau lebih grup yang berulang (*repeating groups*).

b) *First Normal Form* (1NF)

Pada bentuk pertama ini merupakan sebuah hubungan dimana perpotongan setiap baris dan kolom yang berisi satu dan hanya satu nilai. Sebelum mentransformasikan

tabel tidak normal (UNF) ke bentuk normal pertama (1NF). terlebih dahulu mengidentifikasi *repeating groups* yang terdapat pada tabel relasi. Kemudian menghilangkan *repeating groups* untuk menghilangkan data rangkap.

c) *Second Normal Form* (2NF)

Pada bentuk normalisasi kedua ini yaitu 2NF merupakan relasi yang terdapat dalam bentuk 1NF dan tiap atribut yang bukan *primary key* sifatnya bergantung penuh secara fungsional pada *primary key (full functional dependency)*

d) *Third Normal Form* (3NF)

Untuk menjalankan bentuk ini, maka bentuk 1NF dan 2NF haruslah terpenuhi terlebih dahulu. Dimana tidak ada atribut bukan *primary key* yang bergantung transitif terhadap *primary key*. Bentuk normal ketiga ini lebih berdasarkan pada konsep peralihan ketergantungan (*transitive dependency*). *Transitive dependency* adalah kondisi dimana A, B, dan C adalah atribut dari sebuah relasi bahwa jika A→B dan B→C, maka C adalah *transitive independent* pada A melewati B (menyatakan bahwa A bukan merupakan *functional dependent* pada B atau C).

e) *Boyce-Codd Normal Form* (BCNF)

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

f) *Fourth Normal Form* (4NF)

Bentuk normal 4NF terpenuhi dalam sebuah tabel jika telah memenuhi bentuk BCNF, dan tabel tersebut tidak boleh memiliki lebih dari satu *multivalued attribute.*

g) *Fifth Normal Form* (5NF)

Bentuk normal 5NF terpenuhi jika tidak dapat memiliki sebuah *lossless decomposition* menjadi tabel-tabel yang lebih kecil.

2.2 *Database Management System* (DBMS)

Database management system merupakan *software* yang membantu pengguna untuk mengakses *database*. Dengan DBMS, seseorang dapat melakukan interaksi dengan *database*, misalnya untuk menambah, mengubah, menghapus, dan melihat data di dalam *database*. Hal ini juga dikemukakan oleh Connolly dan Begg (2010, p.66), DBMS merupakan sebuah sistem *software* yang memungkinkan pengguna untuk menjelaskan, membuat, memelihara dan mengontrol akses dalam *database*.

2.2.1 Arsitektur DMBS

Arsitektur dari aplikasi DBMS terdiri dari beberapa jenis yang digunakan untuk mengimplementasikan DBMS, seperti yang dikatakan oleh Connolly dan Begg, (2010, p.108) Arsitektur dalam DBMS terdiri atas :

- Arsitektur aplikasi *Teleprocessing*

Aplikasi *teleprocessing* adalah aplikasi komputer yang dapat dijalankan hanya pada satu komputer. *Database* dan programnya berada dalam satu komputer itu sendiri.

- Arsitektur aplikasi *File-Server*

Aristektur aplikasi *file-server* didasarkan pada hal sederhana yaitu komputer yang berbeda melakukan tugas yang berbeda juga. Aplikasi dipecah-pecah ke dalam dua komponen utama kemudian bekerjasama untuk mencapai satu tujuan.

- Arsitektur aplikasi *two-tier client-server*

Arsitektur aplikasi yang dijalankan pada sisi *client*. Mengandung kode yang menampilkan data dan berinteraksi dengan *user*.

- Arsitektur aplikasi *three-tier client-server*

Peran *database* disini adalah mengakses dan meng-*update* data. *Client* bertanggung jawab menampilkan data kepada pengguna dan mengirimkan *input* dari pengguna. Tingkat ini merupakan sebuah objek yang berada diantara aplikasi *client* dan *server*.

Pada arsitektur ini, terdapat sebuah komputer yang mana disebut sebagai *server*, dan ada beberapa komputer lagi yang lainnya dimana biasanya disebut sebagai *client*.

- Arsitektur *N-Tier*

Arsitektur *three-tier* dapat diperlebar menjadi *n-tier*. Dengan menambah *tier* dapat menyediakan fleksibilitas dan skalabilitas yang lebih banyak.

2.2.2 Komponen dari Lingkungan DBMS

Untuk dapat membuat dan menjalankan DBMS dengan baik, maka diperlukan komponen-komponen DBMS. Menurut Connolly dan Begg (2010, p.68), kita dapat mengidentifikasi 5 komponen utama dalam lingkungan DBMS :

1) *Hardware*

Untuk menjalankan DBMS dan aplikasi, diperlukan *hardware*. *Hardware* yang diperlukan tergantung pada kebutuhan organisasi. Beberapa DBMS bekerja hanya pada *hardware* atau *operating system* tertentu.

2) *Software*

Untuk menjalankan DBMS, tidak dapat hanya dengan *hardware* namun diperlukan *software* dan aplikasi program untuk beekerja dengan *operating system* agar DBMS dapat dijalankan. *Software* dapat berupa C++, C#, *Visual Basic*, Fortran, dan lain-lain.

3) Data

Salah satu komponen yang paling penting dalam DBMS adalah data. Data adalah apa yang akan dilihat oleh pengguna akhir nantinya. Data akan menjadi jembatan antara komponen manusia dan komponen mesin.

4) Prosedur

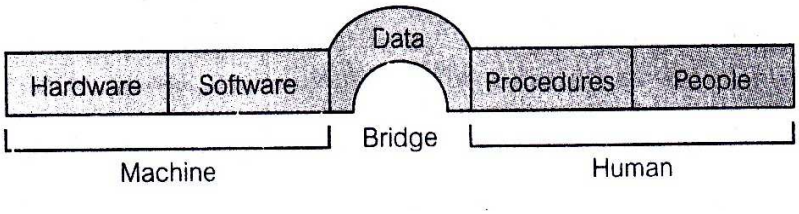
Prosedur mengacu pada instruksi dan aturan yang mengatur desain dan penggunaan *database*. Pengguna dari sistem dan anggota yang mengelola

membutuhkan dokumentasi prosedur tentang cara untuk menjalankan dan menggunakan sistem. Dan berikut langkah dalam menggunakan sistem :

- Masuk ke dalam DBMS
- Gunakan fasilitas atau program aplikasi
- Mulai dan berhenti DBMS
- Membuat cadangan *database*
- Menangani perangkat lunak atau perangkat lunak yang rusak
- Mengubah struktur *table*, mengelola *database* silang, menambah performa, membuat data cadangan.

5) Manusia

Komponen terakhir adalah manusia yang terlibat dengan sistem. Komponen manusia ini terbagi lagi berdasarkan peran mereka dalam lingkungan *database*.



Gambar 2.2 Komponen DBMS

Sumber : Connolly dan Begg (2010, p.68)

2.2.3 Peran dalam Lingkungan *Database*

Dalam menjalankan DBMS, manusia sebagai salah satu komponen adalah yang memiliki peran untuk merancang dan menggunakan DBMS. Berdasarkan peran-peran tersebut, komponen manusia dapat dibagi. Hal ini juga dikatakan oleh Connolly dan Begg (2010, p.71), kita dapat mengidentifikasi 4 tipe dari orang yang berpartisipasi dalam lingkungan DBMS :

1. Data dan *Database Administrator*

   Data dan *database administration* adalah peran yang secara umumnya terhubung dengan pengaturan dan pengontrolan dari DBMS dan datanya. *Data administrator* (DA) bertanggung jawab untuk mengatur sumber data, termasuk perencanaan *databasei; pengembangan* dan perawatan dari standarisasi, kebijakan dan prosedur; desain konseptual/logikal. *Database Administrator* (DBA) bertanggung jawab terhadap realisasi fisikal dari *database*, termasuk desain fisikal dan implementasi, keamanan dan pengontrolan integritas, perawatan sistem operasional, dan memastikan kinerja yang memuaskan dari aplikasi untuk pengguna.

2. *Database designers*

   Kita dapat membedakan *database designers* mwnjadi dua tipe. *Logical database designer* yang berhubungan dengan pengidentifikasian data, relasi antara data, dan batasan pada data yang disimpan dalam *database*. *Physical database designer* yang memutuskan bagaimana desain *database* logikal akan direalisasikan secara fisikal.

3. *Application Developers*

   Application developers bertanggung jawab untuk memastikan program aplikasi dari *database* yang telah diimplementasikan menyediakan kebutuhan fungsional untuk pengguna akhir.

4. *End-User*

   End-User adalah "*clients*" dari *database* yang telah di desain dan dimplementasikan untuk menyediakan informasi yang dibutuhkan. *End-User* dapat diklasifikasikan berdasarkan cara penggunaan sistem. *Naive users* adalah tipe yang tidak menyadari DBMS, mereka mengakses *database*

berdasarkan program aplikasi yang tertulis dan berusaha untuk membuat operasi sesederhana mungkin. Berbeda *Sophisticated users* yang mengenal struktur *database* dan fasilitas yang disediakan oleh DBMS.

2.2.4 Fungsi DBMS

Ketika menggunakan DBMS, kita tentunya mengharapkan fungsi dari DBMS yang dapat kita gunakan DBMS menyediakan beberapa tipe fungsi dan layanan yang dapat kita gunakan. Seperti yang dikatakan oleh Connolly dan Begg, (2010, p.99-104), fungsi daripada DBMS antara lain :

1. Menyimpan, menampilkan, dan mengubah data

   Sebuah DBMS diharuskan memiliki kemampuan untuk menyimpan, menampilkan, dan mengubah data di dalam basis data. Dan DBMS akan menyembunyikan detil dari implementasi fisikal internal DBMS dari pengguna lainnya

2. Sebuah *user-accessible catalog* (Katalog data untuk pengguna)

   Dalam DBMS harus menyediakan katalog yang mana berisi deskripsi dari data yang disimpan didalamnya dan informasi mengenai siapa saja yang diberi hak akses untuk mengakses data didalamnya. Pada dasarnya yang disimpan dalam sistem katalog terdiri dari:

   a. Nama, tipe, dan ukuran data
   b. Nama hubungan
   c. Batasan integritas dalam data
   d. Nama pengguna yang memiliki hak akses ke basis data
   e. Data yang dapat diakses pengguna dan tipe hak akses
   f. Skema eksternal, konseptual, internal serta pemetaan antar skema
   g. Statistik pemakaian

3. Mendukung transaksi

Sebuah DBMS harus menyediakan sebuah mekanisme yang dapat melakukan perubahan terhadap transaksi yang sudah terjadi untuk user.

1) Layanan kontrol konkurensi

DBMS harus dapat membuat sebuah sebuah cara kerja dimana dapat mengatur pengguna sehingga saat pengguna merubah basis data bersamaan tidak terjadi *crash* data.

2) Layanan perbaikan

Dalam satu DBMS, harus ada sebuah mekanisme untuk memperbaiki basis data yang bermasalah dengan menggunakan cara apa pun.

3) Layanan otorisasi

DBMS harus dapat menyediakan validasi untuk dapat memastikan bahwa *user* yang hendak mengaksesnya itu memang sudah memiliki otoritas.

4) Mendukung komunikasi data

Sebuah DBMS harus dapat menghubungkan antar perangkat lunak dimana digunakan untuk komunikasi data.

5) Layanan integritas

Sebuah DBMS harus dapat menyediakan cara untuk memastikan data dalam basis data dan perubahan data tersebut dapat mengikuti peraturan yang telah diatur sebelumnya.

6) Memberikan kemampuan independensi data

Pada DBMS harus memiliki fasilitas dimana mendukung independensi program dari struktur basis data.

7) Layanan utilitas

Sebuah DBMS harus menyediakan sekumpulan layanan utilitas, seperti :

a) Fasilitas *Import* data

b) Fasilitas *monitoring*

c) Program analisis statistik

d) Fasilitas pengaturan ulang indeks

Garbage collection dan realokasi

2.2.5 Fasilitas DBMS

Dari penerapan sebuah DBMS, tentunya akan mendapatkan fasilitas yang akan menguntungkan perusahaan yang menerapkan DBMS itu sendiri. Diantaranya fasilitas yang akan di dapatkan seperti yang dikutip dari Connolly dan Begg, (2010, p.66) fasilitas yang disediakanoleh DBMS antaralain :

a) Data Definition Language (DDL) yang memungkinkan pengguna untuk menspesifikasikan tipe dan struktur data, serta batasan pada data yang disimpan dalam basis data.

b) Data Manipulation Language (DML) yang memungkinkan pengguna untuk memasukkan, mengubah, menghapus, dan menampilkan data dari basis data

2.2.6 Keuntungan dan Kekurangan DBMSs

DBMS menjanjikan banyak keuntungan yang dapat digunakan, namun tentunya DBMS juga memiliki kekurangan yang perlu diperhatikan. Pertama-tama kita perlu melihat keuntungan apa saja yang diberikan oleh DBMS. Menurut Connolly dan Begg, (2010, p.77-80) keuntungan dalam menggunakan DBMS ialah:

a. Mengendalikan pengulangan data (*control of data redudancy*)

Database dimana tempat menampung banyaknya data yang mana dari banyaknya data tersebut memungkinkan adanya pengulangan data. Oleh karena itu dibutuhkannya *Database* dalam mengendalikan pengulangan data dengan cara mengintegrasikan *file* sehingga berbagai data yang sama tidak akan disimpan dalam

database, akan tetapi hal ini tidak menghilangkan semua pengulangan data yang ada di database secara menyeluruh, melainkan hanya membuat jumlah pengulangan data dapat dikontrol.

b. Konsistensi Data

Dengan mengontrol perulangan data, dapat mengurangi resiko terjadinya ketidak konsistensian yang terjadi. Jika data yang disimpan hanya sekali dalam *database*, maka berbagai perubahan bagi nilai data tersebut juga hanya dilakukan satu kali, dan nilai tersebut harus tersedia kepada semua pengguna. Dan jika *item* data yang disimpan lebih dari sekali dan sistem menyadari hal ini, sistem ini dapat memastikan bahwa semua salinan item disimpan secara konsisten.

c. Semakin banyak informasi yang didapat dari data yang sama

Dengan mengintegrasikan data operasional, dapat memungkinkan perusahaan untuk memperoleh informasi tambahan dari data yang sama.

d. Pembagian data (*Sharing of data*)

Database merupakan bagian dari keseluruhan organisasi dan dapat dibagikan ke semua pengguna yang mempunyai wewenang.

e. Meningkatkan Integritas data

Integritas data mengacu pada validitas dan konsistensi data yang disimpan, integritas biasanya diekspresikan dalam istilah batasan, yang berupa aturan konsisten yang tidak boleh dilanggar oleh *database*, dan dapat memungkinkan DBA untuk menjelaskan, dan memungkinkan DBMS untuk membuat batasan integritas.

f. Meningkatkan keamanan data

Meningkatkan keamanan data dimana memproteksi database dari *user* yang tidak memiliki wewenang.

g. Penerapan standarisasi

Integrasi memungkinkan DBA untuk mendefinisikan dan membuat *standard* yang diperlukan. Standarisasi ini termasuk standarisasi departemen, organisasi, nasional, dan internasional dalam hal format data, dan berguna untuk memfasilitasi pertukaran data antara sistem, ketepatan, penamaan standarisasi dokumentasi, prosedur *update*, dan aturan pengaksesan.

h. Pengurangan biaya

Dimana semua data operasional organisasi dipusatkan ke dalam *database* dan membuat aplikasi yang bekerja pada satu sumber data dapat menghasilkan pengurangan biaya. Jadi dengan penyatuan biaya untuk pengembangan dan pemeliharaan sistem pada setiap departemen akan menghasilkan total biaya yang dikeluarkan akan lebih rendah. Sehingga sisa biaya yang merupakan penghematan sebelumnya dapat digunakan untuk hal lain yang dapat meningkatkan performa bagi kebutuhan organisasi.

i. Menyeimbangkan konflik kebutuhan

Setiap pengguna mempunyai kebutuhan yang mungkin berbeda dengan kebutuhan pengguna lain. Oleh karena itu database dikendalikan oleh DBA (*Database Administrator),* DBA juga yang akan membuat keputusan berkaitan dengan perancangan dan penggunaan operasional database yang menyediakan penggunaan terbaik dari sumber daya bagi keseluruhan organisasi.

j. Meningkatkan kemampuan pengaksesan dan respon pada data

Dengan mengintegrasikan data yang melintasi batasan departemen dapat langsung diakses oleh pengguna akhir, hal ini dapat menyediakan sebuah sistem dengan lebih banyak fungsi.

k. Meningkatkan produktivitas

DBMS menyediakan banyak fungsi standar yang biasanya ditulis oleh programmer dalam aplikasi berbasis file. Pada tingkat dasar, DBMS juga menyediakan semua rutinitas tingkat rendah penanganan file yang khas dalam program aplikasi. Fungsi-fungsi ini memungkinkan seorang programmer untuk berkonsentrasi pada fungsi tertentu yang dibutuhkan oleh seorang pengguna.

DBMS banyak juga menyediakan lingkungan generasi keempat, yang digunakan untuk menyederhanakan pengembangan aplikasi database. Hal ini menyebabkan produktivitas pemprogram meningkat dan waktu pengembangan berkurang (dengan penghematan biaya yang terkait).

l. Meningkatkan pemeliharaan melalui data independensi

Dalam sistem berbasis file, deskripsi data dan logika untuk mengakses data yang dibangun ke dalam setiap program aplikasi, membuat program bergantung pada data. Perubahan dengan cara data disimpan pada disk dapat memerlukan perubahan besar untuk program yang dipengaruhi oleh perubahannya. Sebaliknya, DBMS memisahkan deskripsi data dari aplikasi, sehingga membuat aplikasi kebal terhadap perubahan dalam deskripsi data.

m. Meningkatkan konkurensi

DBMS juga digunakan untuk mengelola akses database secara bersamaan dan memastikan bahwa masalah seperti dua atau lebih pengguna yang diijinkan untuk mengakses *file* yang secara bersamaan akan mengakibatkan hilangnya informasi dan integritas.

n. Meningkatkan backup dan perbaikan layanan

Backup data merupakan hal yang perlu diperhatikan karena untuk melindungi data dari kegagalan sistem. Backup harus sering dilakukan seiring dengan proses datanya dan apabila selama pekerjaan telah terjadi kegagalan maka backup dipergunakan.

Dan DBMS modern memberikan fasilitas untuk meminimalkan jumlah pengolahan yang hilang setelah terjadi kegagalan.

Setelah melihat keuntungan yang dimiliki DBMS, kita perlu melihat kerukurangan yang dimiliki oleh DBMS untuk dipertimbangkan. Menurut Connolly dan Begg, (2010, p.80) kerugian dalam menggunakan DBMS ialah :

a. Kompleksitas

Ketentuan dari fungsi yang diharapkan dari DBMS yang baik membuat DBMS menjadi sebuah *software* yang sangat kompleks, perancang dan pengembang *database*, DA, dan DBA, serta pengguna akhir harus memahami funsi tersebut untuk mendapatkan banyak keuntungan dari DBMS tersebut.

b. Ukuran data yang besar

Fungsi yang kompleks dan luas membuat DBMS menjadi *software* yang sangat besar, sehingga memerlukan banyak ruang *harddisk* dan jumlah memori yang digunakan menjadi sangat besar untuk berjalan dengan efisien.

c. Biaya dari DBMS

Biaya DBMS bervariasi tergantung pada lingkungan dan fungsi yang disediakan. Padahal tersebut terdapat biaya pemeliharaan tahunan yang juga dimasukan dalam daftar harga DBMS.

d. Biaya penambahan perangkat keras

Kebutuhan tempat penyimpanan bagi DBMS dan database sangat membutuhkan pembelian tempat penyimpanan tambahan lebih lanjut untuk mencapai performa yang diperlukan, dan akan membutuhkan spesifikasi perangkat keras yang lebih muktahir dan sebagainya yang memerlukan biaya yang tidak sedikit. Tergantung pada spesifikasi perangkat keras yang diperlukan.

e. Biaya konversi

Di dalam situasi tertentu, biaya DBMS dan hardware tambahan mungkin relatif kecil biayanya dibandingkan dengan biaya mengkonversi aplikasi yang ada untuk berjalan di DBMS baru dan perangkat keras. Biaya ini juga termasuk biaya pelatihan karyawan untuk menggunakan sistem baru, dan mungkin kerja dengan karyawan spesialis untuk membantu dengan konversi dan menjalankan sistem. Biaya ini merupakan salah satu alasan utama mengapa beberapa organisasi merasa terikat pada sistem mereka saat ini. dan tidak bisa beralih ke teknologi database yang lebih modern.

f. Kinerja

Kinerja DBMS secara umum sangat baik. Namun, DBMS ditulis lebih umum, untuk melayani banyak aplikasi bukan hanya satu. Hasilnya adalah bahwa beberapa aplikasi mungkin berjalan secepat dulu. Dan biasanya, sistem *file-based* ditulis untuk aplikasi tertentu seperti faktur.

g. Dampak kegagalan lebih besar

Sumber daya terpusat akan meningkatkan kerentanan sistem. Karena semua pengguna dan aplikasi bergantung pada ketersediaan DBMS, sehingga apabila terjadi kegagalan komponen tertentu dapat membawa operasi berhenti.

2.3 *Relational Model*

Dalam melakukan perancangan *database*, kita perlu membuat model relasi untuk mengetahui hubungan-hubungan yang dimiliki dalam *database*. Hal ini akan membantu kita dalam menghadapi redudansi data, konsistensi data, dan manipulasi data. Menurut Connolly dan Begg, (2010, p.142), Tujuan *relational model* dispesifikasikan sebagai berikut :

- Memungkinkan tingkat independensi data yang lebih tinggi.
- Menyediakan alasan yang substansial atau kuat untuk menangani masalah data semantik, konsistensi, dan redudansi.
- Memungkinkan perluasan dari bahasa manipulasi set data orientasi

2.3.1 Struktur Data Relasional

Dalam membuat model relasi, terdapat struktur yang perlu dipahamai terlebih dahulu. Menurut Connolly dan Begg, (2010, p.144) struktur data relational terbagi menjadi beberapa bagian yaitu :

1. Relasi

   Suatu relasional adalah suatu tabel dengan kolom dan baris.

2. Atribut

   Suatu atribut adalah suatu yang dinamakan kolom sebuah relasi.

   1. *Domain*

   Suatu *domain* adalah satuan nilai-nilai yang bisa diijinkan untuk satu atribut atau lebih.

   2. *Tuple*

      Suatu *tuple* adalah suatu barus suatu relasi.

   3. *Degree*

      Degree suatu hubungan adalah jumlah atribut yang dikandungnya.

   4. *Cardinality*

      Cardinality adalah suatu relasi banyaknya tuples yang berisi.

   5. *Relational Database*

      Suatu koleksi dari relasi yang telah dinormalisasi dengan nama relasi yang berbeda.

2.3.2 Sifat-sifat Relasi

Untuk dapat membuat model relasi, perlu dipahami sifat-sifat yang dimiliki oleh relasi. Dengan adanya sifat relasi, penggambaran model relasi akan lebih mudah. Menurut Connolly dan Begg, (2010, p.148-149) relasi mempunyai sifat-sifat sebagai berikut :

1. Relasi memiliki nama yang berbeda satu sama lain dalam skema relasional
2. Setiap sel dari relasi memiliki satu nilai atomik
3. Setiap atribut memiliki nama yang berbeda
4. Nilai satu atribut berasal daru *domain* yang sama
5. Setiap *tuple* pasti berbeda, dan tidak ada duplikasi *tuple*
6. Urutan atribut tidak memiliki makna
7. Urutan *tuple* tidak mempunyai makna, secara teoritis

2.4 *Entity Relationship Modeling*

Entity Relationship Modeling merupakan penyajian data yang berupa penggambaran sebuah entitas dan *relationship* dimana saling berhubungan di dalam suatu organisasi.ER-*Modeling* ini akan membantu kita untuk mengerti dengan lebih tepat mengenai sifat data dan bagaimana data digunakan di dalam perusahaan. Seperti yang disampaikan oleh Connolly dan Begg, (2010, p.371) *Entity Relationship Modeling* merupakan *top-down approach* dalam *database design* dengan mengidentifikasi data yang penting yang disebut *entities* dan *relationships* diantara data yang memberi gambaran dalam bentuk model. Didukung oleh pernyataan Chilton (2006, p.385), bahwa data modeling adalah formulasi struktur data sedemikian rupa sehingga makna dari data yang mengartikan data secara jelas dan mudah dikomunikasikan dan berbagi dengan mereka yang membutuhkan untuk

mengakses. ER-*Modeling* ini penting untuk menggambarkan dan mengkomunikasikan relasi antara *entitas* kepada desainer, programer, dan pengguna untuk mendapatkan pemahaman yang sama.

Entity adalah suatu pengelompokan objek dimana mempunyai karakteristik yang sama dan dapat dibedakan dari *entity* yang lain. Seperti yang disampaikan oleh Connolly dan Begg, (2010, p.372) *Entity Type* merupakan sekumpulan objek dengan properti yang sama, yang diidentifikasikan oleh perusahaan memiliki keberadaan yang independen. *Entity* ini merepresentasikan objek yang ada di dunia nyata sehingga memungkinkan bagi desainer yang berbeda untuk mengidentifikasi entitas dan properti yang berbeda.

2.4.1 Tipe Relasi

Tipe relasi merupakan tipe hubungan antara satu entitas dengan entitas lainnya. Relasi akan membantu untuk memahami bagaimana entitas saling terkait satu sama lain. Hal ini juga dinyatakan oleh Connolly dan Begg, (2010, p.374) tipe relasi adalah satu asosiasi antara satu atau lebih tipe entitas yagn berpartisipasi. Tipe relasi dapat dibagi menjadi 2, menurut Connolly dan Begg (2010, p.374-375) ialah:

- *Relationship type*

Merupakan satu rangkaian asosiasi yang menjelaskan satu atau lebih jenis entitas yang berpartisipasi.

- *Relationship occurrence*

Suatu hubungan yang unik dimana dapat diidentifikasi dengan melihat satu kejadian dari setiap jenis entitas yang berpartisipasi.

2.4.2 Attributes

Atribut merupakan properti yang dimiliki oleh entitas. Setiap entitas memiliki atribut yang merepresentasikan data yang disimpan dalam *database*. Seperti yang dikatakan oleh Connolly dan Begg, (2010, p.379) atribut adalah properti dari entitas atau tipe relasi. Masing-masing atribut akan memiliki nilai yang disebut *domain*. *Attribute domain* merupakan himpunan nilai yang memperbolehkan untuk memiliki satu atau lebih atribut. Hal ini juga dinyatakan oleh Connolly dan Begg, (2010, p.379) *attribute domain* adalah satu set nilai yang diperkenankan untuk satu atau lebih atribut.

Atribut dapat diklasifikasikan menjadi beberapa jenis. Hal ini juga dinyatakan oleh Connolly dan Begg, (2010, p.382) atribut dapat diklasifikasikan menjadi :

1) *Simple and composite attributes*

Simple attribute merupakan sebuah atribut yang terdiri dari komponen tunggal dengan keberadaan yang independen. Nama lainnya *atomic* atribut.

Composite attribute merupakan sebuah atribut yang terdiri dari beberapa komponen dimana masing-masing dengan keberadaan independen.

2) *Single valued and multi valued attributes*

Single-valued attribute merupakan sebuah atribut yang menyimpan nilai tunggal untuk setiap terjadinya suatu entitas.

Multi-valued attribute merupakan sebuah atribut yang menyimpan nilai ganda untuk setiap kemunculan suatu entitas.

3) *Derived attributes*

Sebuah atribut yang mewakili nilai yang diturunkan dari nilai atribut terkait, belum tentu dalam jenis entitas yang sama.

4) Key

- *Candidate key*

  Merupakan set minimal atribut yang secara unik mengidentifikasi setiap kemunculan suatu tipe entitas.

- *Primary key*

  Merupakan sebuah *candidate key* yang dipilih secara unik dengan mengidentifikasi setiap tipe entitas.

- *Composite key*

  Merupakan sebuah *candidate key* yang terdiri dari dua atau lebih atribut.

2.5 Metodologi Perancangan Basis Data

Database design merupakan proses perancangan *database* yang akan mendukung sistem dalam perusahaan. *Database design* diperlukan agar *database* yang dibuat sesuai dengan kebutuhan perusahaan sehingga memiliki guna yang sesuai dan tidak membuang biaya perusahaan dengan sia-sia. Seperti yang juga dikatakan oleh Connolly dan Begg (2010, p.320) *database design* adalah proses membuat sebuah desain yang akan mendukung pernyataan misi dan tujuan misi perusahaan untuk sistem *database* yang diperlukan. Dalam melakukan perancangan, perlu dilakukan mengikuti langkah-langkah yang ada untuk menghasilkan rancangan yang sesuai dengan kebutuhan. Hal ini juga diperkuat oleh Connolly dan Begg (2010, p.322) *database dseign* terdiri dari tiga tahap utama: konseptual, logikal, dan fisikal.

2.5.1 Perancangan Basis Data Konseptual

Perancangan basis data konseptual merupakan langkah pertama yang dilakukan dalam merancang sistem basis data. Pada langkah ini difokuskan pada pemodelan data konseptual perusahaan yang sama sekali tidak bergantung pada detil-

detil implementasi. Perancangan basis data konseptual ini bertujuan untuk membangun model data konseptual dari kebutuhan data dari perusahaan. Model data konseptual terdiri dari : tipe entitas, tipe relasi, atribut dan *domain* atribut, *primary key* dan *alternate key*, dan batasan integritas. Model data konseptual juga didukung oleh dokumentasi, termasuk ER diagram dan kamus data yang dihasilkan melalui pengembangan model data konseptual. Seperti yang disampaikan oleh Woldemichael dan Hashim (2011, p.253), bahwa desain konseptual adalah proses pengetahuan intensif yang memerlukan beragam pengetahuan. Dengan demikian, permodelan harus mencakup sarana untuk menyimpan dan mengambil pengetahuan desain untuk meningkatkan pengetahuan mendesain. Proses desain konseptual dapat dianggap sebagai spesifikasi transformasi desain, yang diberikan sebagai persyaratan menjadi satu atau lebih konsep yang dapat memenuhi persyaratan untuk pengembangan lebih lanjut.

Langkah- langkah yang dijalankan pada tahap ini:

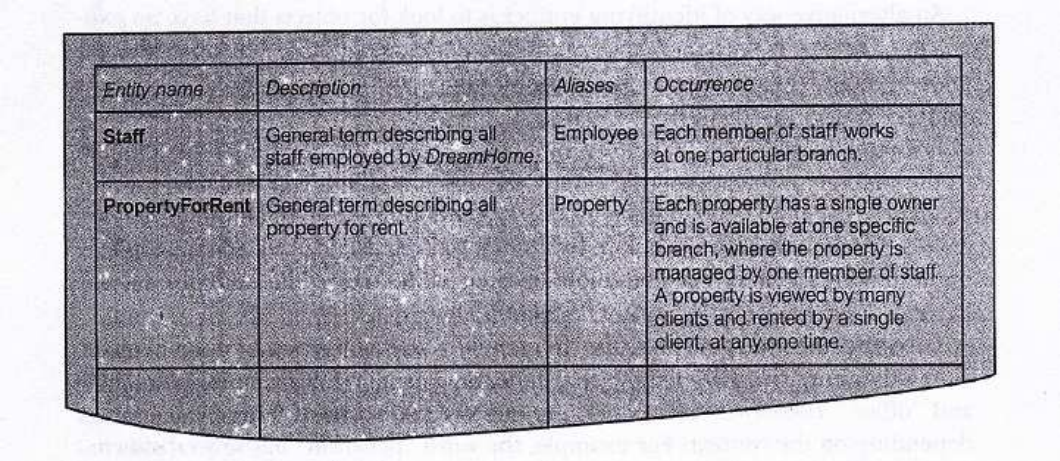
1. Mengidentifikasi tipe entitas

   Pertama-tama dilakukan pengidentifikasian entitas dengan melihat objek-objek yang digunakan oleh *user*. Tujuannya adalah untuk mengidentifikasi entitas yang diperlukan.

2. Mengidentifikasi tipe relasi

   Tujuan identifikasi tipe relasi ini adalah untuk mengetahui relasi penting yang ada antara entitas-entitas. Pada tahap ini, relasi digambarkan menggunakan ER Diagram untuk mempermudah dalam melihat relasi.

   Contoh :



Entity name	Description	Aliases	Occurrence
Staff	General term describing all staff employed by *DreamHome*.	Employee	Each member of staff works at one particular branch.
PropertyForRent	General term describing all property for rent.	Property	Each property has a single owner and is available at one specific branch, where the property is managed by one member of staff. A property is viewed by many clients and rented by a single client, at any one time.

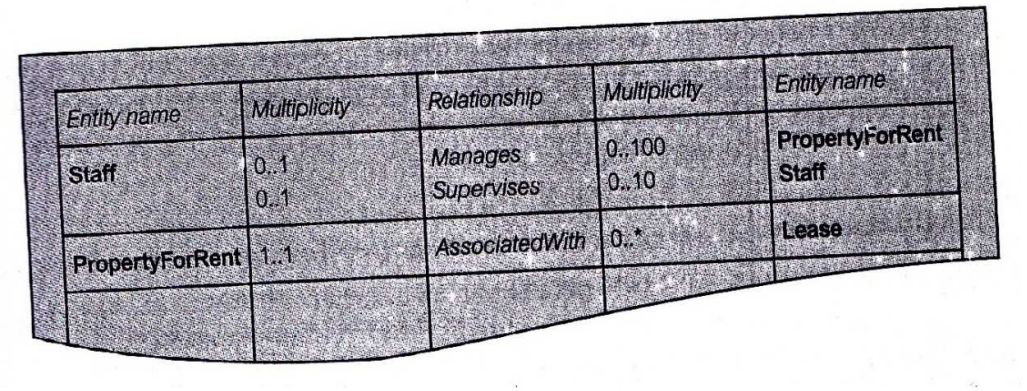
Gambar 2.3 Tipe Relasi

Sumber : Connolly dan Begg (2010, p.472)

3. Mengidetifikasikan dan menghubungkan atribu-atribut dengan entitas atau relasi

   Pada langkah ini, ditentukan tipe-tipe fakta mengenai entitas dan relasi yang akan di masukkan ke dalam basis data. Tujuan langkah ini adalah untuk menghubungkan atribut-atribut dengan entitas atau relasi yang sesuai.

   Contoh :



Entity name	Multiplicity	Relationship	Multiplicity	Entity name
Staff	0..1 0..1	Manages Supervises	0..100 0..10	PropertyForRent Staff
PropertyForRent	1..1	AssociatedWith	0..*	Lease

Gambar 2.4 Asosiasi atribut dengan entitas atau tipe relasi

Sumber : Connolly dan Begg (2010, p.474)

4. Menentukan atribut *domain*

Menurut Connolly dan Begg (2010, p.478) *Domain* adalah sebuah kumpulan nilai-nilai dari satu atau lebih atribut yang diambil nilainya. Tujuan dari langkah ini adalah untuk menentukan *domain* untuk atribut-atribut pada model data konseptual.

5. Menentukan atribut kunci kandidat, *primary*, dan *alternate*

   Tujuan dari langkah ini adalah untuk mengidentifikasi *candidate key* untuk setiap entitas dan, jika ada lebih dari satu *candidate key*, maka akan dipilih satu dari antara semua *candidate key* tersebut untuk menjadi *primary key* sementara sisanya menjadi *alternate key*.

6. Mempertimbangkan penggunaan konsep *enhanced modeling* (*optional*)

   Tujuan langkah ini adalah untuk mempertimbangkan penggunaan *enhanced modeling*, seperti *spezialitation*/*generalization*, *aggregation*, dan *composition*.

7. Melakukan pengecekan redudansi pada model

   Pada tahap ini, tujuannya adalah untuk melakukan pemeriksaan model data konseptual untuk menidentifikasi adanya redudansi dan menghilangkannya.

8. Menvalidasi model data konseptual terhadap transaksi *user*

   Tujuan langkah ini adalah untuk memastikan bahwa model data konseptual yang dibuat mendukung kebutuhan transaksi. Untuk memastikan model data konseptual mendukung kebutuhan transaksi, ada dua pendekatan yang dapat digunakan :

   - Mendeskripsikan transaksi
   - Menggunakan *transaction pathways*

9. Melakukan *review* model data konseptual dengan *user*

   Tujuan dari langkah ini adalah untuk melihat kembali model data konseptual bersama *user* untuk memastikan *user* mempertimbangkan model yang telah dibuat sesuai dengan kebutuhan data perusahaan.

2.5.2 Perancangan Basis Data Logikal

Tujuan utama dari perancangan basis data logikal ini adalah untuk menerjemahkan model data konseptual ke dalam model data logikal yang kemudian divalidasi untuk memeriksa benar tidaknya model data logikal secara struktural dan kemampuannya untuk mendukung kebutuhan transaksi.

Langkah-langkah yang dijalankan pada tahap ini:

1. Mengambil relasi untuk model data logikal

   Tujuan dari langkah ini adalah untuk membuat relasi untuk model data logikal untuk mempresentasikan entitas, relasi, dan atribut yang telah di identifikasi. Dalam mendeskripsikan relasi yang diambil, berikut relasi yang mungkin terjadi:

   - Tipe entitas kuat

     Menurut Connolly dan Begg (2010, p.492), Entitas kuat adalah entitas yang keberadaannya tidak bergantung pada keberadaan entitas lainnya. Dimana pada entitas ini mempunyai karakter bahwa setiap kejadian entitas teridentifikasi secara unik yang memiliki atribut primary key yang dapat membedakan dari entitas yang lain.

   Contoh :

   Ketika ingin mengidentifikasi secara unik setiap siswa dalam sekolah, dapat menggunakan atribut NIS(Nomor Induk Siswa) untuk dapat membedakan.

   Siswa (NIS, nama, alamat, jenis kelamin, agama)

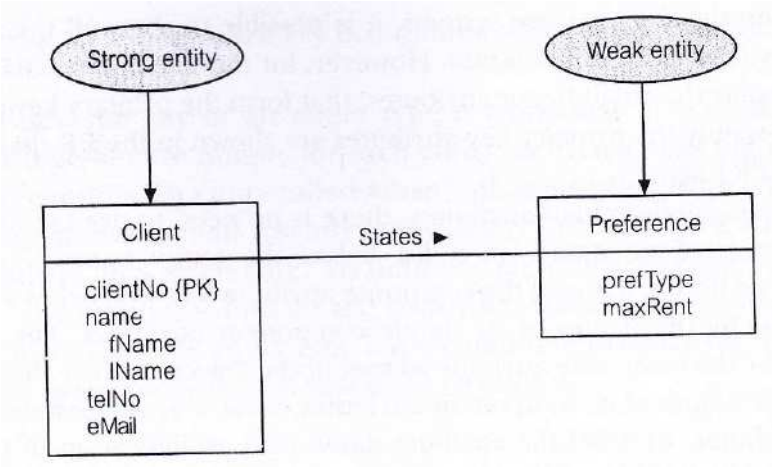
Primary Key : NIS

- Tipe entitas lemah

Sedangkan tipe entitas lemah merupakan entitas yang keberadaanya tergantung oleh beberapa entitas yang lain. Pada entitas ini, karakteristik yang dimilikinya ialah setiap kejadian entitas tidak dapat teridentifikasi secara unik hanya dengan menggunakan atribut entitas tersebut, tidak seperti halnya tipe entitas kuat yang menggunakan primary key.

Contoh:

Pada entitas *preference* hanya memiliki atribut, dimana hanya dapat mengidentifikasikan secara unik entitas *preference* melalui relasi yang dimilikinya dengan entitas *client* yang secara unik mempunyai *primay key* untuk mengidentifikasi entitas *client.*



Gambar 2.5 Contoh entitas kuat dan entitas lemah

Sumber : Connolly dan Begg (2010, p.384)

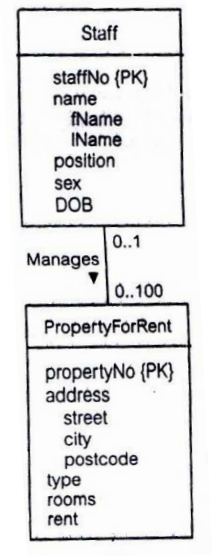
Gambar 2.4 dimana entitas kuat yaitu *Client* dan entitas lemah yaitu *Preference*

- *One-to-many* (1:*) tipe relasi *binary*

  Kondisi ini terjadi apabila tiap anggota pada entitas pertama mempunyai pasangan dengan lebih dari satu anggota entitas kedua.

  Contoh:

  Pada entitas *Staff* dapat berpasangan dengan lebih dari satu anggota dari entitas *PropertyForRent*. Sebaliknya juga pada setiap anggota entitas *PropertyForRent* hanya dapat berpasangan dengan satu anggota entitas *Staff*.

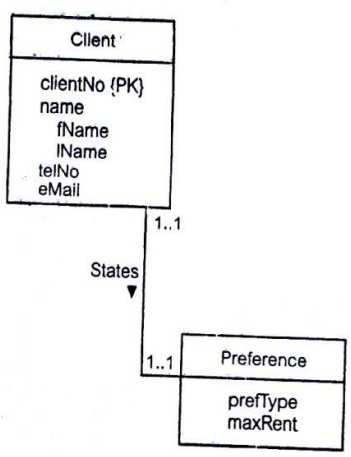


Gambar 2.6 Contoh tipe relasi *binary* dengan kondisi *one to many*

Sumber : Connolly dan Begg (2010, p.491)

- *One-to-one* (1:1) tipe relasi *binary*

  Pada tipe relasi ini dapat terjadi apabila tiap anggota pada entitas *Client* hanya dapat berpasangan dengan satu anggota dari entitas *Preference*. Demikian sebaliknya tiap entitas *Preference* juga hanya dapat berpasangan dengan satu anggota dari entitas *Client.*



Gambar 2.7 Contoh tipe relasi *binary* dengan kondisi *one to one*

Sumber : Connolly dan Begg (2010, p.491)

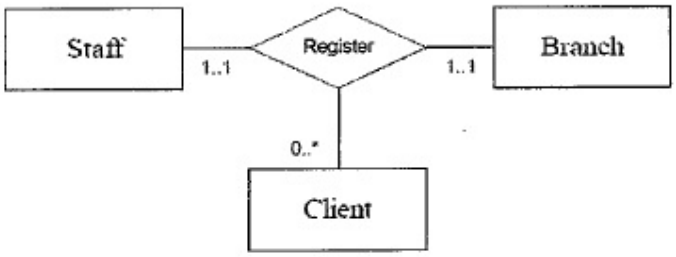
- *One-to-one* (1:1) tipe relasi rekursif
- *Superclass*/*subclass relationship types*
- *Many-to-many* (*:*) tipe relasi *binary*

  Pada hubungan antar entitas ini akan terjadi apabila kedua anggota entitas tersebut dapat berpasangan lebih dari satu anggota entitas.

  Contoh:

  Pada tiap anggota entitas Buku dapat berpasangan lebih dari satu anggota entitas Peminjaman dan demikian juga sebaliknya.
- Tipe Relasi yang komplek

  Relasi yang komplek adalah jumlah dari suatu kejadian yang mungkin dari suatu entitas dalam n-ary *relatioship* ketika nilai entitas yang lain (n-1) diketahui.



Gambar 2.8 Contoh tipe relasi yang komplek dengan *ternary*

Sumber : Connolly dan Begg (2010, p.384)

Contoh *multiplicity* pada *relationship* ternary

Tabel 2.1 Tabel batasan multiplicity

Cara alternatif to menggambarkan batasan pada *multiplicity*	Artinya
0.. 1	Nol atau satu kejadian
1.. 1 (atau hanya 1)	Hanya satu kejadian
0.. * (atau hanya *)	Nol atau banyak kejadian
1.. *	Satu atau banyak kejadian
5.. 10	Minimum 5 dan maksimal 10 kejadian
0, 3, 6-8	Nol atau tiga atau enam, tujuh, atau delapan kejadian

Sumber : Connolly dan Begg (2010, p.391)

- Atribut *multi-valued*

2. Validasi relasi menggunakan normalisasi

   Tujuan langkah ini adalah untuk melakukan validasi model data logikal menggunakan normalisasi. Tujuan dari normalisasi ini adalah untuk

memastikan set dari relasi mempunyai sejumlah atribut yang minimal namun cukup yang diperlukan untuk mendukung kebutuhan data perusahaan.

3. Validasi relasi terhadap transaksi *user*

   Tujuan langkah ini adalah untuk memastikan relasi model data logikal yang dibuat mendukung kebutuhan transaksi, secara detil seperti yang dispesifikasikan pada kebutuhan *user*.

4. Mendefinisikan batasan integritas

   Tujuan langkah ini adalah untuk memeriksa apakah batasan integritas ditampilkan dalam model data logikal. Batasan integritas ini bertujuan untuk melindungi basis data dari ketidak lengkapan, ketidak akuratan, atau ketidak konsistenan. Berikut adalah beberapa tipe batasan integritas :

- *Required data*

  Beberapa atribut harus memiliki isi pada datanya sehingga tidak diperbolehkan menerima *null*.

- *Attribute domain constraints*

  Masing-masing atribut memiliki domain yang merupakan sekumpulan nilai yang sah.

- *Multiplicity*

- *Entity integrity*

  Dimana *primary key* dari sebuah entitas tidak dapat bernilai *null*.

- *Referential integrity*

  Bila *foreign key* mempunyai nilai, maka nilai tersebut haruslah menunjuk pada *tuple* yang ada pada relasi induk. Untuk melakukan itu, *referential integrity* perlu dispesifikasikan *existence constraints* yang mendefinisikan kondisi dimana sebuah *candidate key* atau *foreign key* ditambahkan, diubah,

atau di hapus. Jika terdapat sebuah *tuple* dari relasi induk dihapus, maka *referential integrity* akan hilang apabila adanya *tuple* anak menunjuk ke *tuple* induk yang dihapus.

Ada beberapa strategi yang dapat digunakan :

a. NO ACTION

Strategi yang digunakan ialah mencegah penghapusan dari relasi induk jika terdapat refrensi ke *tuple* anak.

b. CASCADE

Apabila pada *tuple* induk dihapus, maka secara otomatis *tuple* anak akan dihapus juga.

c. SET NULL

Jika pada *tuple* induk dihapus, maka nilai *foreign key* pada semua *tuple* anak otomatis akan terisi nilai *null*.

d. SET DEFAULT

Jika pada *tuple* induk dihapus, maka *foreign key* pada semua *tuple* akan menerima nilai *default.*

e. NO CHECK

Jika *tuple* induk dihapus, maka tidak dilakukan apapun untuk meyakinkan bahwa *referential integrity* terjaga.

- *General constraints*

  Merupakan batasan atau aturan tambahan yang dibuat oleh *user* atau seseorang *database administrator* dari basis data tersebut.

5. Melakukan *review* model data logikal dengan *user*

   Tujuan langkah ini adalah untuk melakukan *review* bersama *user* untuk memastikan *user* mempertimbangkan model yang telah dibuat sesuai dengan kebutuhan data perusahaan.

6. Menggabungkan model data logikal menjadi model *global* (*optional*)

   Tujuan dari langkah ini adalah untuk menggabungkan model data logikal menjadi data logikal *global tunggal* yang menampilkan semua *user views* dari basis data.

7. Melakukan pemeriksaan untuk perkembangan di masa datang

Tujuan langkah ini adalah untuk menentukan apakah ada perubahan yang signifikan di masa mendatang dan untuk menilai apakah model data logikal dapat mengakomodasi perubahan-perubahan tersebut.

2.5.3 Perancangan Basis Data Fisikal

Dalam proses perancangan basis data fisikal, akan dihasilkan deskripsi implementasi dari basis data pada *secondary storage*. Perancangan basis data fisikal akan mendeskripsikan basis relasi, *file* organisasi dan index yang digunakan untuk mencapai akses data yang efisien dan segala batasan integritas yang terkait dan masalah sekuritas.

Langkah-langkah yang dijalankan pada tahap ini:

- Menerjemahkan model data logikal untuk target DBMS

  Tujuan langkah ini adalah untuk menghasilkan skema relasi basis data dari model data logikal yang dapat diimplementasikan pada target DBMS. Untuk menjalankan tahap ini, terbagi menjadi 3 langkah sebagai berikut :

- Merancang relasi dasar

  Pada langkah ini, hal pertama yang dilakukan adalah pengumpulan dan pemahaman informasi tentang relasi yang telah dihasilkan selama perancangan logikal basis data. Setelah itu, menentukan bagaimana mempresentasikan relasi dasar pada target DBMS.

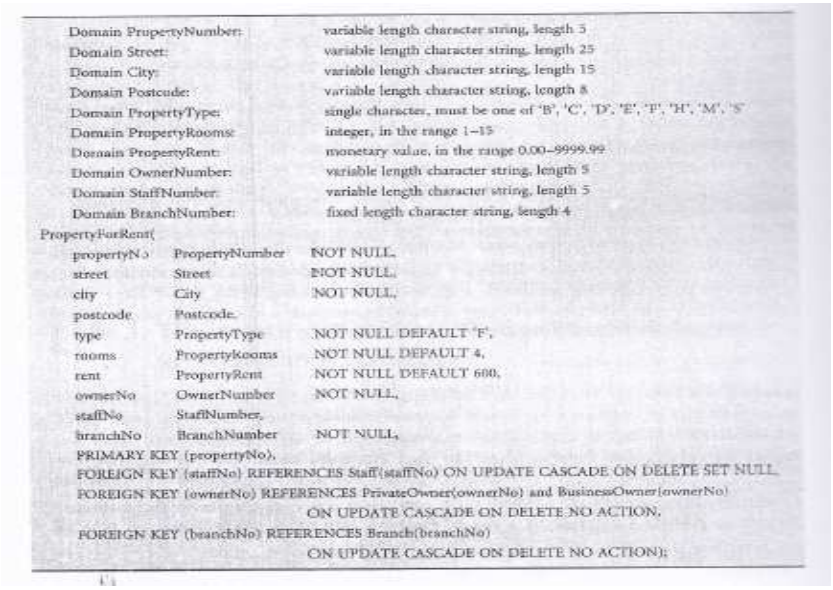
  Setiap relasi yang diidentifikasikan dalam model data logikal memiliki definisi yang terdiri dari :

  1) Nama relasi
  2) Daftar atribut sederhana
  3) *Primary key*, *alternate keys*, dan *foreign keys*
  4) Referensial integritas batasan untuk setiap *foreign keys* yang diidentifikasi

  Dari kamus data, setiap atribut juga memiliki :

  1) Domain atribut yang terdiri dari tipe data, panjang, dan batasa-batasan pada domain
  2) Sebuah *optional* bernilai *default* untuk atribut
  3) Apakah atribut boleh bernilai *null*
  4) Apakah atribut merupakan data yang diambil dan bagaimana data dihitung

Contoh :



```
Domain PropertyNumber:      variable length character string, length 5
Domain Street:              variable length character string, length 25
Domain City:                variable length character string, length 15
Domain Postcode:            variable length character string, length 8
Domain PropertyType:        single character, must be one of 'B', 'C', 'D', 'E', 'F', 'H', 'M', 'S'
Domain PropertyRooms:       integer, in the range 1–15
Domain PropertyRent:        monetary value, in the range 0.00–9999.99
Domain OwnerNumber:         variable length character string, length 5
Domain StaffNumber:         variable length character string, length 5
Domain BranchNumber:        fixed length character string, length 4
PropertyForRent(
    propertyNo      PropertyNumber      NOT NULL,
    street          Street              NOT NULL,
    city            City                NOT NULL,
    postcode        Postcode,
    type            PropertyType        NOT NULL DEFAULT 'F',
    rooms           PropertyRooms       NOT NULL DEFAULT 4,
    rent            PropertyRent        NOT NULL DEFAULT 600,
    ownerNo         OwnerNumber         NOT NULL,
    staffNo         StaffNumber,
    branchNo        BranchNumber        NOT NULL,
    PRIMARY KEY (propertyNo),
    FOREIGN KEY (staffNo) REFERENCES Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY (ownerNo) REFERENCES PrivateOwner(ownerNo) and BusinessOwner(ownerNo)
                                ON UPDATE CASCADE ON DELETE NO ACTION,
    FOREIGN KEY (branchNo) REFERENCES Branch(branchNo)
                                ON UPDATE CASCADE ON DELETE NO ACTION);
```

Gambar 2.9 Contoh Relasi Dasar

Sumber : Connolly dan Begg (2010, P.526)

- Merancang representasi dari data yang diambil

  Tujuan langkah ini adalah untuk menentukan bagaimana mempresentasikan data yang diambil dari data yang ada di dalam model data logikal pada target DBMS.

Contoh :

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Staff

staffNo	fName	lName	branchNo	noOfProperties
SL21	John	White	B005	0
SG37	Ann	Beech	B003	2
SG14	David	Ford	B003	1
SA9	Mary	Howe	B007	1
SG5	Susan	Brand	B003	0
SL41	Julie	Lee	B005	1

Gambar 2.10 Contoh *Derived Data*

Sumber : Connolly dan Begg (2010, P.527)

- Merancang batasan umum

  Tujuan langkah ini adalah untuk merancang batasan umum untuk target DBMS.

  Contoh :

  DreamHome memiliki peraturan yang melarang anggota *staff* mengatur lebih dari 100 properti di waktu yang bersamaan.

```
CONSTRAINT StaffNotHandlingTooMuch
        CHECK (NOT EXIST (SELECT staffNo
                          FROM PropertyForRent
                          GROUP BY staffNo
                          HAVING COUNT (*) > 100)
```

1. Merancang *file* organisasi dan index

   Tujuan langkah ini adalah untuk menentukan *file* organisasi yang optimal untuk menyimpan relasi dasar dan index yang diperlukan untuk mencapai kinerja yang sesuai, yaitu bagaimana jalan dari relasi dan *tuples* dapat disimpan dalam *secondary storage*.

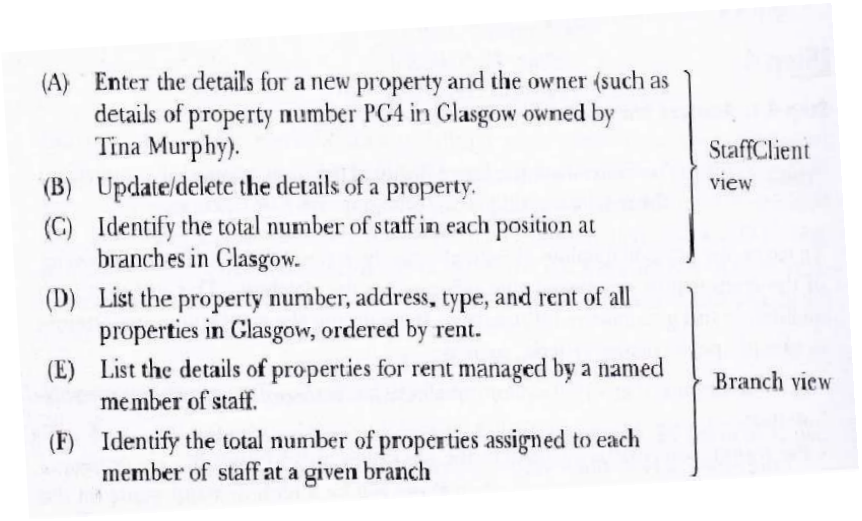
   Tahap ini dibagi menjadi 4 langkah, yaitu:

   - Menganalisa transaksi

Tujuan langkah ini adalah untuk memahami transaksi secara fungsional yang akan dijalankan dalam basis data dan untuk menganalisa transaksi yang penting. Untuk membantu mengidentifikasi transaksi mana yang perlu diteliti, dapat menggunakan *transaction relation cross-refrence matrix* yang menunjukkan relasi setiap akses tranasaksi. Untuk fokus pada area yang mungkin bermasalah, salah satu cara untuk melanjutkan :

1) *Map* seluruh jalan transaksi untuk relasi
2) Tentukan relasi mana yang sering diakses oleh transaksi
3) Analisa penggunaan data dari transaksi yang dipilih yang melibatkan transaksi-transaksi lain.

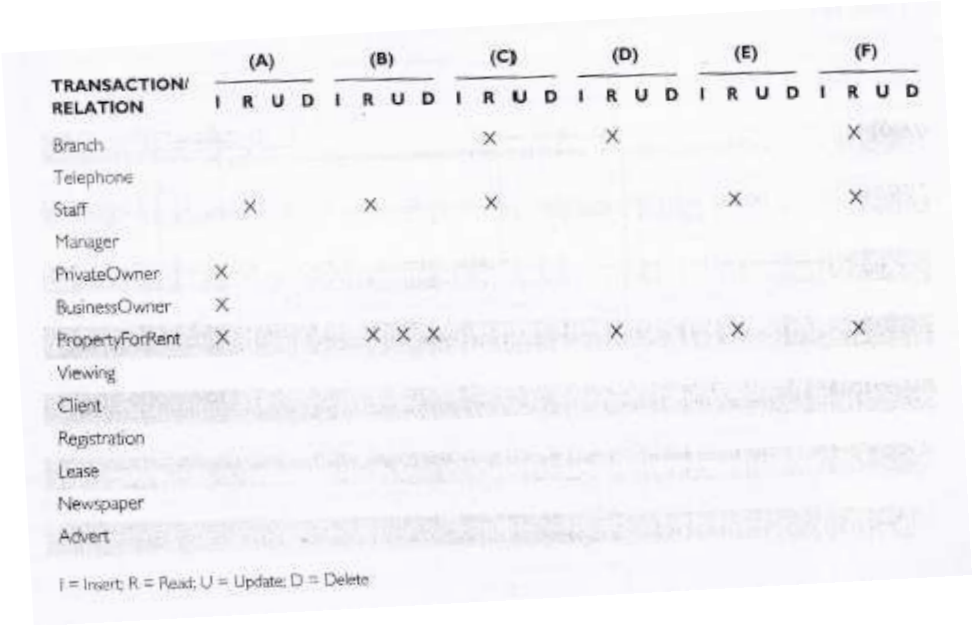
Contoh :



(A)	Enter the details for a new property and the owner (such as details of property number PG4 in Glasgow owned by Tina Murphy).	StaffClient view
(B)	Update/delete the details of a property.	
(C)	Identify the total number of staff in each position at branches in Glasgow.	
(D)	List the property number, address, type, and rent of all properties in Glasgow, ordered by rent.	Branch view
(E)	List the details of properties for rent managed by a named member of staff.	
(F)	Identify the total number of properties assigned to each member of staff at a given branch	

Gambar 2.11 *Map Path*

Sumber : Connolly dan Begg (2010, P.530)



TRANSACTION/ RELATION	(A)				(B)				(C)				(D)				(E)				(F)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Branch										X				X								X		
Telephone																								
Staff		X				X				X								X				X		
Manager																								
PrivateOwner	X																							
BusinessOwner	X																							
PropertyForRent	X					X	X	X						X				X				X		
Viewing																								
Client																								
Registration																								
Lease																								
Newspaper																								
Advert																								

I = Insert; R = Read; U = Update; D = Delete

Gambar 2.12 *Cross-refrencing transactions and relations*

Sumber : Connolly dan Begg (2010, P.531)

- Memilih *file* organisasi

  Tujuan langkah ini adalah untuk menentukan organisasi *file* yang efisien untuk setiap relasi dasar.

- Memilih index

  Tujuan langkah ini adalah untuk menentukan apakah penambahan index akan meningkatkan kinerja dari sistem.

Pemilihan atribut untuk *ordering* atau *clustering* adalah sebagai berikut :

1. Atribut yang banyak digunakan untuk operasi *join*, yang membuat operasi *join* semakin efisien
2. Atribut yang banyak digunakan untuk mengakses tupel dalam sebuah relasi yang memiliki atribut tersebut

   Contoh :

   Untuk membuat index utama pada relasi *PropertyForRent* berdasarkan pada atribut *propertyNo*.

   CREATE UNIQUE INDEX propertyNoInd ON PropertyForRent (propertyNo)

- Memperkirakan kapasitas *disk* yang diperlukan

  Tujuan langkah ini adalah untuk memperkirakan jumlah kapasitas *disk* yang akan diperlukan oleh basis data sehingga dapat diperkirakan penggunaan kapasitas *disk* setiap tahunnya.

2. Merancang *user views*

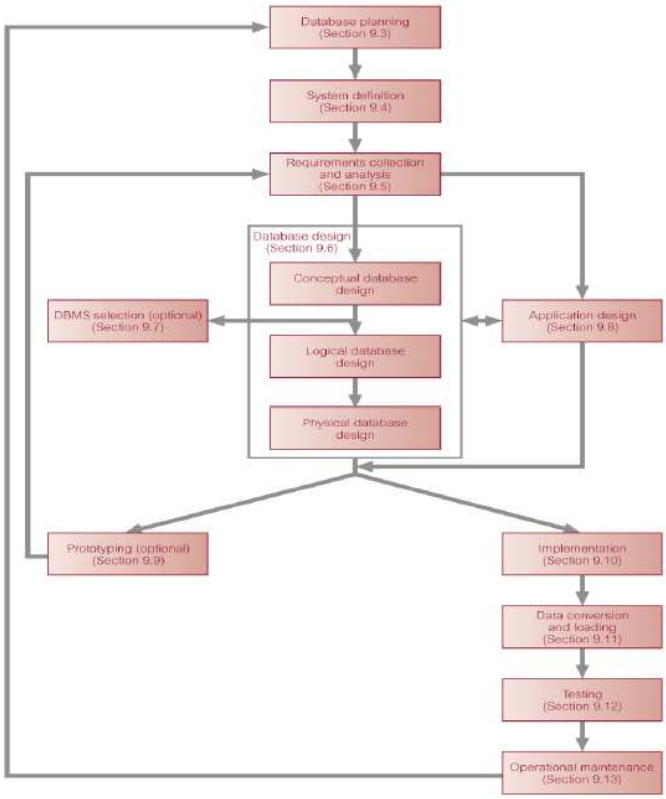
Tujuan langkah ini adalah untuk merancang *user views* yang diidentifikasi selama pengumpulan kebutuhan dan analisa tahap dari pengembangan siklus kehidupan sistem basis data.

3. Merancang mekanisme *security*

Tujuan langkah ini adalah untuk merancang mekanisme *security* untuk basis data seperti yang telah dispesifikasikan oleh *user* selama tahap pengumpulan dan analisa kebutuhan dari pengembangan siklus kehidupan sistem basis data.

2.6 Database Application Lifecycle

Untuk melakukan perancangan basis data, terdapat langkah-langkah yang akan menjadi dasar dalam perancangan. Penting untuk mengetahui dasar dari perancangan agar dapat mengatasi kendala-kendala yang mungkin terjadi selama perancangan. Menurut Connolly dan Begg (2010, P.314) tahap perancangan basis data berdasarkan siklus hidup seperti tergambar di bawah ini :



Gambar 2.13 Siklus Kehidupan Aplikasi Basis Data

Sumber : Connolly dan Begg (2010, p.314)

1) *Database Planning*

Dalam melakukan suatu perancangan, tahap pertama yang perlud dilakukan adalah merencanakan. Menurut Connolly & Begg, (2010, p.313) *database planning* yaitu tahapan perencanaan bagaimana siklus kehidupan dari basis data dapat direalisasikan dengan cara yang paling efektif dan efisien. Langkah awal pada tahap ini adalah menentukan *mission statement*, yaitu menentukan tujuan utama dari sistem basis data yang akan dirancang. Setelah mendefinisikan *mission statement*, berikutnya adalah mengidentifikasi *mission objectives* beserta tugas-tugasnya. Sistem basis data Yang dirancang harus mendukung *mission objective*.

2) System Definition

System definition merupakan tahapan untuk mengspesifikasikan *scope* dan *boundary* dari sistem basis data, termasuk sudut pandang *user*, siapa saja *user*-nya, dan area dari aplikasi.

3) *Requirement Collections and Analysis*

Requirement collections and analysis adalah tahapan pengumpulan dan analisa kebutuhan untuk sistem basis data yang baru.

4) *Database Design*

Database design merupakan tahapan desain untuk perancangan basis data konseptual, logikal, dan fisikal.

5) DBMS *Selection*

DBMS *selection* merupakan tahapan untuk memilih DBMS yang sesuai untuk sistem basis data.

6) *Application Design*

Application design merupakan tahapan untuk mendesain *user interface* dan program aplikasi yang digunakan dan diproses oleh basis data.

7) Prototyping

Prototyoping merupakan tahapan untuk membangun model mengenai cara kerja dari sistem basis data yang dapat membantu desainer ataupun *user* untuk mengvisualisasikan dan mengevaluasi bagaimana sistem akhir akan tampil dan berfungsi.

8) Implementation

Implementation merupakan tahapan untuk membuat definisi basis data fisikal dan program aplikasi.

9) Data Conversion and Loading

Data conversion and loading merupakan tahapan *loading* data dari sistem lama ke sistem baru dan jika mungkin, mengkonversi semua aplikasi yang ada di sistem lama di sistem yang baru.

10) Testing

Testing merupakan tahapan di mana sistem basis data di uji untuk melihat apakah ada *error* dan validasi yang tidak sesuai dengan kebutuhan yang dispesifikasikan oleh *user*.

11) Operational Maintenance

Operational Maintenance merupakan tahapan di mana sistem basis data telah diimplementasikan secara keseluruhan. Sistem dimonitor dan di*maintain* secara berlanjut. Ketika diperlukan, kebutuhan yang baru dimasukkan ke dalam sistem basis data melalui tahap sebelum siklus kehidupan.

2.7 Proses Bisnis

Dalam menjalankan perancangan *database*, terdapat proses bisnis yang berkaitan dengan perancangan untuk mendapatkan kebutuhan yang perlu dipenuhi *database*. Proses tersebut akan terkait dengan pengintegrasian yang biasa disebut *Enterprise*

resource planning (ERP). Proses bisnis biasanya digambarkan dengan beberapa cara, salah satunya adalah *Rich Picture*.

2.7.1 Pendidikan

Pendidikan adalah kebutuhan yang dibutuhkan oleh manusia untuk mengembangkan diri baik. Seperti yang disampaikan Tim Pengembang Ilmu Pendidikan FIP - UPI (2007, p.12), pendidikan adalah kebutuhan manusia yang khusus bersifat manusiawi sesudah kebutuhan makan-minum dan biologis. Seperti dinyatakan sejak dulu oleh Aristoteles hanya manusia yang memerlukan pendidikan mengingat tingkat jiwanya yang bersifat "*anima intellectiva*". Oleh karena itu dapat didefinisikan secara sederhana bahwa pendidikan adalah upaya-upaya dalam proses kegiatan manusia sebagai pribadi-pribadi untuk secara sengaja (*intensional*) membina perkembangan dari pribadi-pribadi sesama manusia sebagai pihak-pihak yang setara dan saling membutuhkan.

2.7.2 Kurikulum

Kurikulum merupakan suatu perencanaan untuk pembelajaran yang dirancang dengan suatu tujuan. Seperti yang dikatakan oleh Null (2011, p.1) Kurikulum merupakan jantung dari edukasi. Alasannya ada dua. Pertama, kurikulum adalah mengenai apa yang harus diajarkan. Kedua, kurikulum mengkombinasikan pikiran, tindakan, dan tujuan. Namun, kurikulum adalah subjek yang spesifik dan nyata yang selalu terikat oleh keputusan yang dibuat di dalam institusi, baik itu sekolah, gereja, agensi non-profit, ataupun program pemerintah.

2.7.3 Administrasi

Administrasi adalah suatu proses dimana diperlukannya suatu pertimbangan dalam menangani segala yang terkait, dengan kebutuhan yang ada dalam suatu organisasi. Seperti yang disampaikan Tim Pengembang Ilmu Pendidikan FIP - UPI

(2007, p.160-161), administrasi adalah suatu proses sosial, yang lebih banyak melihat dan mempertimbangkan segi-segi kemanusiaannya, dan hubungan antar manusia di dalam proses administrasi itu.

Agar perencanaan dan pelaksanaan fungsi administrasi pendidikan berjalan dengan lancar dan wajar, maka proses pelaksanaan hendaknya berdasarkan prinsip-prinsip yang berfungsi sebagai pedoman dalam administrasi pendidikan tersebut.

Prinsip dasar administrasi pendidikan yang demokratis mencakup :

1. Tujuan pendidikan dan perkembangan anak didik harus mendasari semua kegiatan administrasi pendidikan.

   Efektif diukur dengan kepentingan pendidikan dan perkembangan anak didik.
2. Penggunaan waktuu, tenaga, alat secara efektif.

   Administrasi harus berusaha menggunakan waktu, tenaga dan alat seefektif mungkin.
3. Ada koordinasi dalam semua usaha.

   Dapat bekerjasama dengan anggota-anggota lainnya dalam kelompok serta dapat mengkoordinasikan usahanya dengan semua anggota kelompoknya.
4. Partisipasi luas dalam menentukan kebijakan dan program.

   Diharapkan dengan penuh kesadaran dan tanggungjawab dapat melaksanakan program kerja. Dan keputusan hendaknya merupakan hasil keputusan bersama.
5. Pemindahan kekuasaan sesuai dengan tanggungjawab.

   Pemindahan wewenang merupakan hal yang lazim dalam suatu organisasi. Penyerahan tanggungjawab secara hierarki memang perlu, tetapi yang perlu diperhatikan ialah keadilan. Dimana kewajiban dan hak seimbang, demikian juga tanggung jawab dan wewenang.

6. Menghindari *overlapping* fungsi.

   Dimana ada kemungkinan suatu fungsi dilakukan oleh lebih dari satu bagian. Hal ini tidak menguntungkan dan tidak efisien dalam kerja. Maka diperlukannya pembatasan dan perumusan yang tegas masing-masing bagian.

2.7.4 Penilaian dan Evaluasi

Penilaian dan evaluasi adalah bagian yang tidak dapat terpisahkan di dalam dunia pendidikan. Seperti halnya hasil penilaian akan digunakan untuk melakukan sebuah evaluasi. Dan evaluasi juga memerlukan sebuah catatan penilaian dalam jangka waktu tertentu. Penilaian adalah proses mengukur kemampuan hasil belajar dalam jangka waktu tertentu. Sedangkan evaluasi adalah tahapan dimana menuturkan hasil proses pembelajaran berdasarkan nilai yang didapat serta kegiatan yang telah ditempuh. Seperti yang disampaikan oleh Tim Pengembang Ilmu Pendidikan FIP - UPI (2007, p.103), evaluasi pendidikan adalah salah satu tugas penting yang harus dilakukan dalam penyelenggaraan pendidikan. Porsi terbesar dari evaluasi pendidikan adalah pada aspek belajar-mengajar.

Obyek penilaian adalah prilaku siswa, seperti kemampuan hasil belajar, kemampuan bawaan, minat, sikap dan kepribadian. Pelaksanaanya menggunakan tes yang pada umumnya tertulis, dan tes yang digunakan umumnya tes baku yang hasilnya diungkapkan dalam norma kelompok.

2.7.5 Perpustakaan

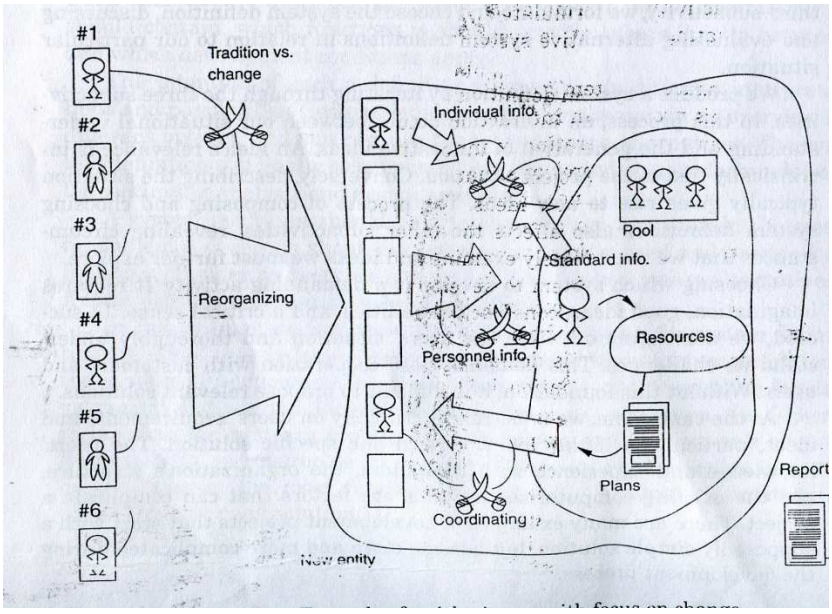
Perpustakaan adalah suatu tempat dimana memperoleh sumber pengetahuan yang berasal dari kumpulan buku yang berguna bagi penggunanya. Pengertian tersebut sesuai dengan buku *Perpustakaan dan Kepustakawanan Indonesia* bahwa perpustakaan adalah suatu unit kerja yang memiliki sumber daya manusia, “ruang

khusus", dan kumpulan koleksi sesuai dengan jenis perpustakaannya (Purwono & Suharmini, 2008, p.12).

2.7.6 Rich Picture

Menurut Mathiassen, Munk-Madsen, Nielsen, & Stage (2000, p.26), *Rich Picture* merupakan penggambaran secara informal yang menampilkan pemahaman illustrator terhadap situasi tertentu. Sebuah *rich picture* fokus pada aspek-aspek penting terhadap situasi yang ditentukan illustrator. *Rich picture* digunakan untuk memberikan gambaran berupa ilustrasi yang terdiri dari *actor* serta *event* simbolik yang menjelaskan proses atau aktivitas yang berjalan.

Contoh penggambaran sebuah *rich picture* sebagai berikut :



Gambar 2.14 Contoh *Rich Picture*

Sumber : (Mathiassen, Munk-Madsen, Nielsen, & Stage, 2000, p.26), *Object Oriented Analysis & Design*

2.7.7 Pengertian *World Wide Web*

World wide web adalah sebuah aplikasi yang membutuhkan internet dan pengaksesannya membutuhkan suatu *client/server*. Seperti yang disampaikan oleh Turban, Rainer, dan Potter, (2005, p.50), bahwa *world wide web* merupakan sebuah aplikasi yang menggunakan fungsi pemindahan dari internet.*Web* merupakan sistem dengan standar yang diterima secara global untuk penyimpanan, pengambilan, pembentukan, dan penampilan informasi melalui suatu arsitektur *client/server*. Demikian juga yang diungkapkan oleh Lee, Cailliau, Groff, dan Pollermann (2010, p.461), bahwa inisiatif *world wide web* merupakan proyek praktis yang dirancang untuk membawa global informasi menjadi ada dengan menggunakan teknologi yang tersedia.

2.7.8 Pengertian *Enterprise Resource Planning*

Enterprise resource planning atau biasa disingkat ERP merupakan suatu proses dimana mengintegrasikan seluruh fungsi yang ada dalam sebuah organisasi dan mengotomatisasikan berbagai proses bisnis di dalam sebuah *software*. Seperti yang disampaikan oleh Satzinger, Jackson, & Burd, (2005, p.17), bahwa *enterprise resource planning* (ERP) adalah sebuah proses dimana suatu organisasi melakukannya dengan menggunakan sekumpulan paket software yang terintegrasi untuk sistem informasi utama.