BAB 2

LANDASAN TEORI

2.1 Teori Umum

2.1.1 Pengertian Sistem

Menurut Connolly dan Begg (2005, p283), sistem adalah suatu cara untuk mengumpulkan, mengatur, mengendalikan, dan menyebarkan informasi ke seluruh organisasi. Menurut O'Brien (2003, p8), sistem adalah kumpulan elemen yang saling terhubung atau berinteraksi membentuk suatu kesatuan atau sekumpulan komponen yang saling terhubung dan bekerja sama untuk mencapai sasaran dengan menerima input dan menghasilkan output dalam sebuah proses transformasi yang terorganisir. Jadi, sistem adalah kumpulan dari semua elemen yang saling berinteraksi, saling mendukung, dan bekerja sama untuk mencapai suatu tujuan tertentu.

2.1.2 Pengertian Data Dan Informasi

Menurut O'Brien (2003, p13), data adalah fakta-fakta atau observasi yang mentah, biasanya mengenai kejadian atau transaksi bisnis. Menurut Elmasri (2000, p4), data adalah fakta-fakta yang dapat disimpan dan memiliki pengertian yang implisit. Jadi, data adalah sesuatu yang masih bersifat mentah mengenai suatu kejadian dan memiliki pengertian yang implisit.

Menurut Turban, Rainer, dan Potter (2003, p15), informasi adalah kumpulan fakta (data) yang disusun dalam beberapa cara sehingga kumpulan fakta tersebut bisa berarti bagi penerimanya.

2.1.3 Pengertian Basis Data

Menurut C.J Date (2000,p5), suatu basis data adalah suatu sistem yang pada dasarnya menyimpan *record – record* di dalam suatu sistem yang dilakukan secara komputerisasi yang tujuannya secara keseluruhan adalah untuk memelihara informasi dan untuk membuat informasi tersebut tersedia berdasarkan permintaan. Menurut O'Brien (2003, p145), basis data adalah sebuah kumpulan yang terintegrasi dari elemen data yang terhubung secara logikal.Elemen data mendeskripsikan entitas-entitas dan hubungan antara entitas-entitas. Menurut Connoly dan Begg (2005,p15), basis data adalah sekumpulan data yang berhubungan secara logikal dan deskripsi mengenai data itu sendiri yang dirancang untuk memenuhi kebutuhan informasi organisasi. Jadi, basis data adalah sebuah tempat penyimpanan data tunggal yang memiliki kapasitas besar yang dapat digunakan secara simultan oleh banyak departemen dan pengguna. Basis data tidak hanya dimiliki oleh sebuah departemen tetapi menjadi sebuah sumber bersama perusahaan. Basis data menyimpan tidak hanya data operasional organisasi tetapi juga deskripsi dari data. Oleh karena itu, basis data juga disebut sebagai *a self-describing collection of integrated records*. Deskripsi data tersebut dikenal sebagai *system catalog* (atau kamus data atau metadata ‘data mengenai

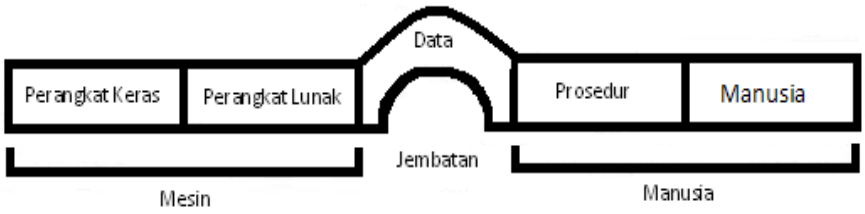
data'). Hal tersebut merupakan sifat dasar basis data yang menyediakan *program-data independence*.

2.1.4 Database *Management System* (DBMS)

Menurut Connolly dan Begg (2005, p16), DBMS adalah suatu sistem perangkat lunak yang memungkinkan user untuk mendefinisikan (*define*), membuat (*create*), memelihara basis data, dan menyediakan kendali dalam mengakses basis data. Menurut Date (1990,p42), DBMS adalah perangkat lunak yang mengatur semua akses menuju basis data. Jadi, DBMS adalah suatu sistem perangkat lunak yang mampu melakukan semua fungsi-fungsi yang diperlukan untuk mengakses basis data.

2.1.4.1 Komponen dari Lingkungan DBMS

Menurut Connoly dan Begg (2005,p18), ada lima komponen dari lingkungan DBMS, yaitu :



Gambar 2.1 Komponen-komponen dalam lingkungan DBMS

1. Perangkat keras (*hardware*)

   DBMS dan aplikasi membutuhkan perangkat keras untuk berjalan. Perangkat keras menjangkau mulai dari komputer personal, *mainframe* tunggal, *minicomputer*, dan sebuah jaringan komputer. Perangkat keras bergantung pada kebutuhan organisasi dan penggunaan DBMS.

2. Perangkat lunak (*software*)

   Komponen perangkat lunak merupakan perangkat lunak DBMS itu sendiri dan program aplikasi, tergabung dengan sistem operasi, termasuk perangkat lunak jaringan apabila DBMS digunakan dalam jaringan.

3. Data

   Komponen paling penting pada lingkungan DBMS, dilihat dari sudut pandang pengguna akhir, adalah data. Data bertindak sebagai jembatan penghubung antara komponen mesin (*hardware* dan *software*) dan komponen manusia (prosedur dan *people*). Basis data mencakup data operasional dan metadata, ‘data mengenai data’.

4. Prosedur

   Prosedur merupakan instruksi dan aturan-aturan yang mengatur perancangan dan penggunaan dari basis data. Pengguna sistem dan staf yang mengatur basis data membutuhkan dokumentasi prosedur tentang bagaimana menggunakan atau menjalankan sistem.

5. Manusia (*People*)

Komponen terakhir dalam lingkungan DBMS adalah manusia (*people*). Ada empat tipe manusia yang berpartisipasi dalam lingkungan DBMS, yaitu :

a. *Data dan Database Administrators*

Data Administrator (DA) bertanggung jawab untuk mengatur sumber data yang mencakup perencanaan basis data, pengembangan dan pemeliharaan, kebijakan dan prosedur, serta perancangan konseptual dan logikal dari basis data. *Database Administrator* bertanggung jawab terhadap realisasi fisikal dari basis data yang mencakup perancangan fisikal dan implementasi, keamanan dan pengaturan integritas, pemeliharaan sistem operasional dan memastikan kepuasan pengguna terhadap performa dari aplikasi.

b. *Database Designers*

Dalam proyek yang besar, kita dapat membedakan dua tipe dari *designer* yaitu *logical database designers* dan *physical database designers*. *Logical database designers* berkonsentrasi pada identifikasi data, hubungan antar data, dan batasan dari data yang disimpan dalam basis data. *Physical database designers* memutuskan bagaimana perancangan logikal dari basis data diubah menjadi realisasi fisikal.

c. *Application Developers*

Ketika basis data diimplementasikan, program aplikasi yang menyediakan fungsi-fungsi yang dibutuhkan oleh pengguna akhir *(end-users)* juga harus diimplementasikan. Hal tersebut merupakan tanggungjawab dari application developers.

d. *End-Users*

End-users merupakan klien dari basis data, yang telah dirancang, diimplementasikan, dan dipelihara untuk menyediakan kebutuhan informasi.

2.1.4.2 Fungsi DBMS

Menurut Date (2000,p43), fungsi DBMS adalah :

1. *Data Definition*

DBMS harus mendukung semua definisi data (skema eksternal, skema konseptual, skema internal) dan melakukan perubahan terhadap skema-skema tersebut ke dalam bentuk objek yang sesuai.

2. *Data Manipulation*

DBMS harus dapat mengatur permintaan dari pengguna untuk mengambil atau mengubah data atau memasukkan data ke dalam basis data.

3. *Data Security and Integrity*

DBMS harus mengawasi permintaan pengguna dan menolak semua hal yang dapat mengganggu keamanan dan pemeriksaan integritas yang telah ditentukan oleh *database administrators*.

4. *Data Recovery and Concurrency*

DBMS atau komponen perangkat lunak lain yang berhubungan (biasanya disebut dengan *transaction manager*) harus menjalankan pengaturan *recovery* dan *concurrency*.

5. *Data Dictionary*

DBMS harus menyediakan fungsi *data dictionary*. *Data dictionary* berisi data tentang data (*metadata*) yang merupakan definisi dari objek-objek dalam sistem. Dalam sebagian kasus, beberapa macam skema dan pemetaan disimpan secara fisikal baik dalam bentuk asli maupun bentuk yang telah disesuaikan.

6. *Performance*

DBMS harus menyediakan semua fungsi yang telah diidentifikasi secara efisien.

2.1.4.3 Keuntungan dan Kerugian DBMS

Menurut Connoly dan Begg (2005, p26), keuntungan dari DBMS adalah :

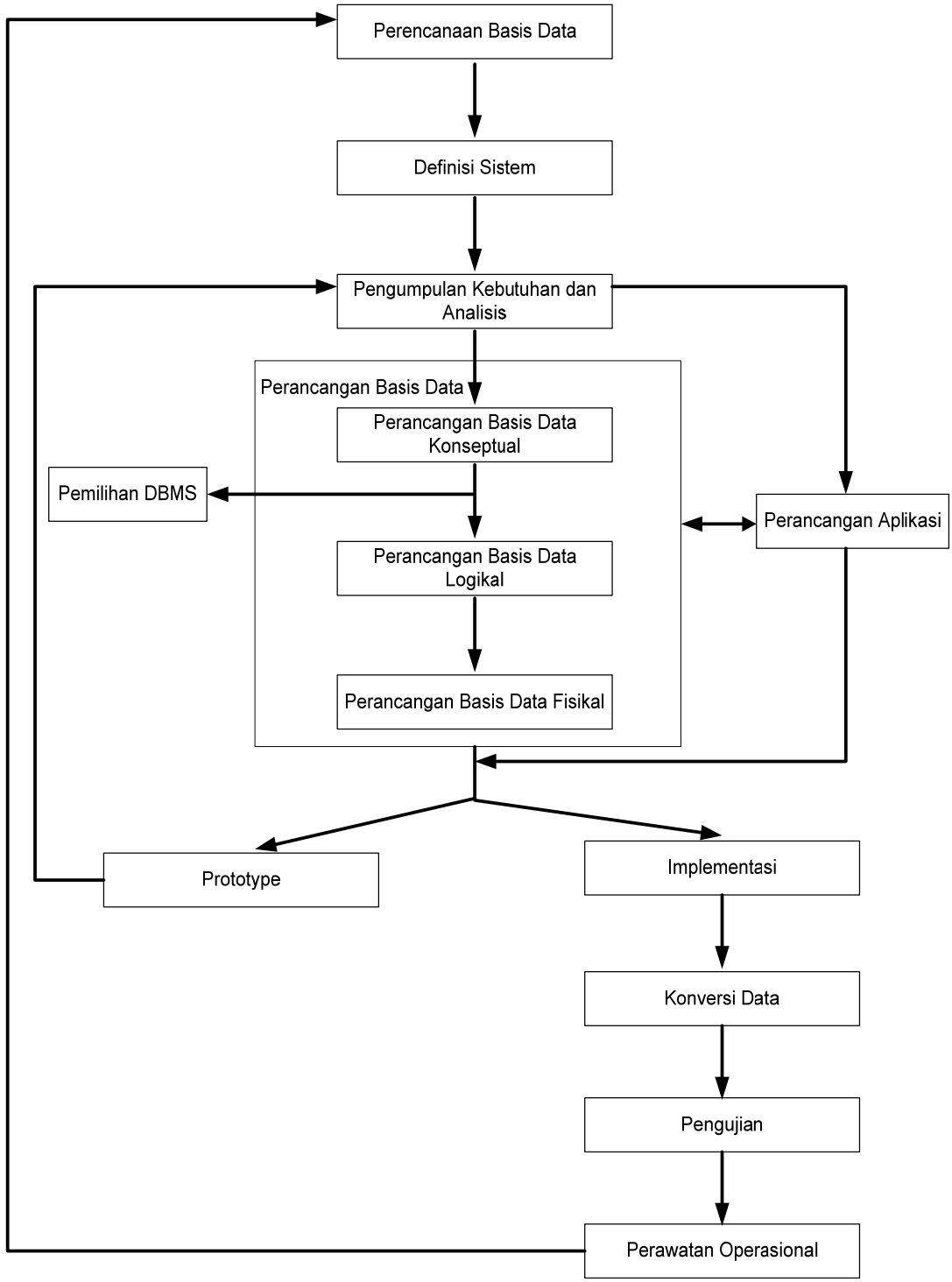
1. Pengaturan redundansi data
2. Konsistensi data
3. Menyediakan lebih banyak informasi dengan jumlah data yang sama
4. *Sharing of data*
5. Meningkatkan integritas data
6. Meningkatkan keamanan
7. Meningkatkan *standard*
8. Skala ekonomi
9. Menyeimbangkan kebutuhan-kebutuhan yang saling bertabrakan
10. Meningkatkan akses dan respon data
11. Meningkatkan produktivitas
12. Meningkatkan pemeliharaan melalui data *independence*
13. Meningkatkan *concurrency*
14. Meningkatkan layanan *backup* dan *recovery*

Sedangkan menurut Connoly dan Begg (2005, p29), kerugian DBMS adalah:

1. Kompleksitas
2. Ukuran
3. Biaya dari DBMS
4. Biaya tambahan perangkat keras
5. Biaya konversi
6. Performa
7. Pengaruh kegagalan yang lebih tinggi

2.1.5 Database Application Lifecycle

Menurut Connoly dan Begg (2005,p284), suatu aplikasi basis data dianalisis dan dirancang dalam tahapan-tahapan berikut :



Gambar 2.2 Tahapan – tahapan aplikasi basis data dianalisis dan dirancang

2.1.5.1 Perencanaan Basis Data

Perencanaan basis data *(database planning)* merupakan aktivitas manajemen yang memungkinkan tahapan-tahapan dari *database application lifecycle* direalisasikan secara efektif dan efisien. Langkah penting dalam tahap perencanaan basis data adalah mendefinisikan secara jelas misi dari proyek basis data. Pernyataan misi *(mission statement)* mendefinisikan tujuan utama dari aplikasi basis data, membantu untuk mengklarifikasi tujuan dari proyek basis data, dan menyediakan jalur yang jelas untuk mencapai pembuatan proyek basis data yang efektif dan efisien.

Setelah pernyataan misi ditentukan, langkah selanjutnya adalah mengidentifikasi misi tujuan *(mission objective).* Setiap misi tujuan harus mengidentifikasi sebagian tugas yang dapat didukung oleh basis data. Perencanaan basis data juga meliputi proses pengembangan standar yang menentukan bagaimana data dikumpulkan, bagaimana format data dispesifikasikan, dokumentasi apa yang diperlukan, dan bagaimana proses desain dan implementasi dilakukan.

2.1.5.2 Definisi Sistem

Definisi sistem *(system definition)* merupakan tahapan untuk menjabarkan ruang lingkup dan batasan dari aplikasi basis data dan sudut pandang user *(user views)* yang utama. Sebelum memasuki tahapan perancangan aplikasi basis data, hal yang kita perlu lakukan terlebih dahulu adalah mengidentifikasi batasan dari sistem dan bagaimana sistem tersebut berhubungan dengan bagian lain dari sistem

informasi organisasi. Hal penting lainnya adalah memasukkan ke dalam batasan sistem tidak hanya *user* dan area aplikasi yang ada sekarang, tetapi juga *user* dan *area* aplikasi di masa mendatang.

2.1.5.3 Pengumpulan dan Analisis Kebutuhan

Pengumpulan dan analisis kebutuhan *(requirement collection and analysis)* merupakan proses pengumpulan dan analisa informasi tentang bagian organisasi yang didukung oleh aplikasi basis data. Informasi yang telah dikumpulkan, kemudian dianalisa untuk mengidentifikasi kebutuhan atau fitur yang akan disertakan dalam aplikasi basis data. Kebutuhan-kebutuhan tersebut dibuat dalam bentuk dokumen dan secara kolektif yang akan berfungsi sebagai spesifikasi kebutuhan untuk aplikasi basis data yang baru. Informasi yang diperoleh mungkin tidak terstruktur dengan baik dan mencakup beberapa permintaan tidak informal. Oleh karena itu, informasi harus dikonversi ke dalam bentuk yang lebih terstruktur jika dibutuhkan.

2.1.5.4 Perancangan Basis data

Perancangan basis data *(database design)* merupakan proses pembuatan desain untuk basis data yang mendukung operasi dan tujuan sebuah *enterprise*. Ada beberapa pendekatan dalam mendesain basis data, yaitu :

1. *Top-Down*

Pendekatan *top-down* diawali dengan pembentukan model data yang berisi beberapa entitas level atas dan hubungan antar entitas tersebut dan kemudian dengan menggunakan pendekatan ini secara berturut-turut untuk mengidentifikasi entitas level bawah, hubungan, dan atribut yang berhubungan. Pendekatan ini digunakan untuk mendesain basis data yang kompleks.

2. *Bottom-Up*

Pendekatan bottom-up dimulai dari level mendasar atribut (sifat sifat entitas dan hubungan) yang dikelompokkan ke dalam suatu relasi yang merepresentasikan tipe dari entitas dan hubungan *(relationship)* antar entitas. Pendekatan ini digunakan untuk mendesain basis data yang sederhana dengan jumlah atribut yang relatif sedikit.

3. *Inside-Out*

Pendekatan ini berhubungan dengan pendekatan *bottom-up*. Perbedaannya adalah pada identifikasi awal entitas utama, kemudian menyebar ke entitas, hubungan, dan atribut yang sebelumnya diidentifikasi.

4. *Mixed Strategy*

Pendekatan *mixed-strategy* menggunakan pendekatan *bottom-up* dan *top-down* untuk bagian yang berbeda dari model data yang pada akhirnya digabungkan sebagai suatu kesatuan model data.

2.1.5.4.1 Tahap dari Perancangan Basis Data

Perancangan basis data terdiri dari tiga tahap utama, yaitu :

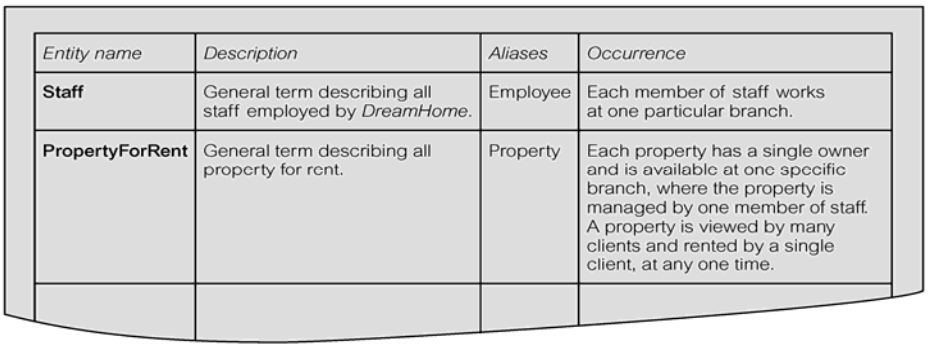
1. Perancangan Konseptual *(Conceptual Basis data Design)*

Perancangan basis data konseptual merupakan proses pembuatan model dari informasi yang digunakan dalam *enterprise*, tidak bergantung pada semua masalah fisik. Tahap ini melibatkan pembuatan konseptual model data perusahaan. Data model konseptual dibuat dengan menggunakan informasi yang didokumentasikan dari spesifikasi kebutuhan user. Perancangan basis data konseptual ini tidak terikat dengan detail-detail seperti target perangkat lunak DBMS, program aplikasi, bahasa pemrograman, *platform*, perangkat keras atau dari perangklat fisik lainnya.

Langkah 1 Membangun model data konseptual untuk setiap view

- Langkah 1.1 Mengidentifikasikan tipe *entity*

Tipe *entity* adalah sekumpulan objek yang diidentifikasikan oleh sebuah perusahaan yang mempunyai sifat-sifat yang sama dan mempunyai keberadaan yang independen (Connoly, 2002, p331). *Entity occurrence* adalah sebuah objek unik yang teridentifikasi dalam tipe *entity* (Connolly dan Begg, 2002, p333).



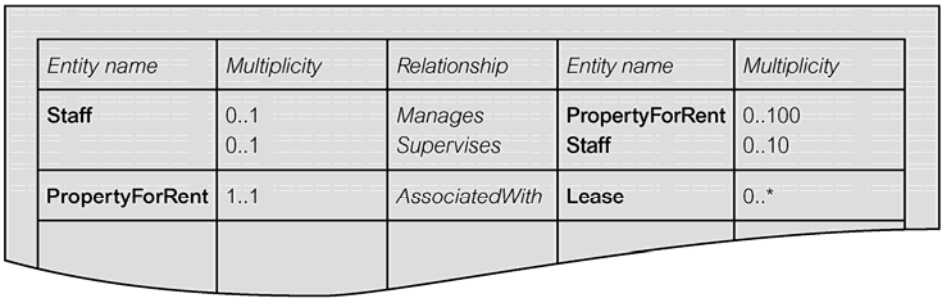
Entity name	*Description*	*Aliases*	*Occurrence*
Staff	General term describing all staff employed by *DreamHome*.	Employee	Each member of staff works at one particular branch.
PropertyForRent	General term describing all property for rent.	Property	Each property has a single owner and is available at 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 Kamus Data *Entity*

(Connolly dan Begg, 2002, p424)

- Langkah 1.2 Mengidentifikasikan tipe *relationship*

Tipe *relationship* adalah sekumpulan hubungan antara satu atau lebih *entity* (Connolly dan Begg, 2002, p334). *Relationship occurrence* adalah sebuah hubungan unik antara satu atau lebih *entity* yang teridentifikasi dalam tipe *relationship.*



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

Gambar 2.4 Kamus Data *Relationship*

(Connolly dan Begg, 2002, p426)

- Langkah 1.3 Mengidentifikasikan dan mengasosiasikan *attributes* dengan *entity* atau tipe *relationship*

Attribute adalah sifat dari sebuah *entity* atau sebuah tipe *relationship* (Connolly dan Begg, 2002, p338). *Attribute* menyimpan nilai dari setiap

entity occurrence dan mewakili bagian utama dari data yang disimpan dalam basis data. Sebagai contoh, tipe *entity Staff* dapat dideskripsikan oleh *attribute staffNo, name, position,* dan *salary*

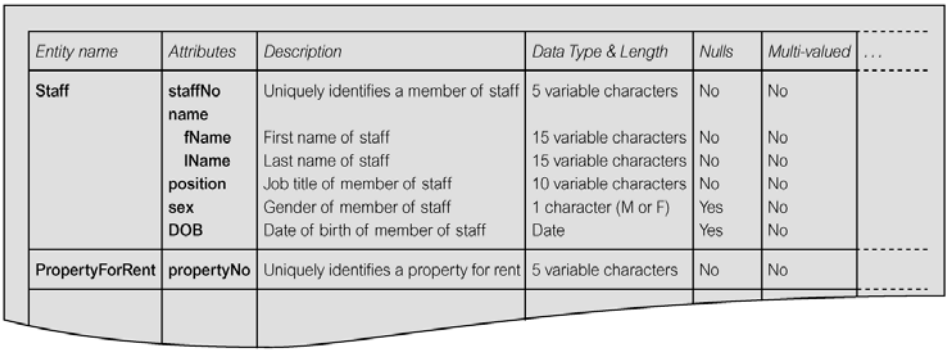
Simple attribute adalah suatu *attribute* yang terdiri atas komponen tunggal dengan keberadaan yang tidak terikat. *Attribute* ini tidak dapat dipecah lagi menjadi menjadi komponen yang lebih kecil. *Simple attribute* terkadang juga disebut sebagai *atomic attributes.*

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

Single-valued attribute adalah *attribute* yang menampung nilai tunggal untuk tiap-tiap kejadian dari suatu tipe *entity.*

Multi-valued attribute adalah *attribute* yang menampung banyak nilai untuk setiap kejadian dari suatu tipe *entity.*

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



Entity name	Attributes	Description	Data Type & Length	Nulls	Multi-valued	...
Staff	staffNo	Uniquely identifies a member of staff	5 variable characters	No	No	
	name					
	fName	First name of staff	15 variable characters	No	No	
	lName	Last name of staff	15 variable characters	No	No	
	position	Job title of member of staff	10 variable characters	No	No	
	sex	Gender of member of staff	1 character (M or F)	Yes	No	
	DOB	Date of birth of member of staff	Date	Yes	No	
PropertyForRent	propertyNo	Uniquely identifies a property for rent	5 variable characters	No	No	

Gambar 2.5 Kamus Data Atribut

(Connolly dan Begg, 2002, p430)

- Langkah 1.4 Menentukan *attribute domain*

  Attribute domain adalah satuan nilai-nilai untuk satu atau beberapa *attribute* (Connolly dan Begg, 2002, p338). *Domain* mendefinisikan nilai-nilai yang dimiliki sebuah *attribute* dan sama dengan konsep *domain* pada *relational model.*

- Langkah 1.5 Menentukan *candidate* dan *primary key attributes*

  Candidate key adalah kumpulan minimal dari *attribute* yang secara unik mengidentifikasikan setiap tipe *entity* (Connolly dan Begg, 2002, p340).

  Primary key adalah *candidate key* yang dipilih secara unik mengidentifikasikan suatu tipe *entity* (Connolly dan Begg, 2002, p341).

- Langkah 1.6 Pertimbangkan untuk menggunakan *konsep enchanced modeling (optional)*

  Mempertimbangkan menggunakan spesialisasi, generalisasi, agregasi dan kompisisi untuk melanjutkan pengembangan dari ER model. Jika menggunakan pendekatan generalisasi, maka kita mencoba untuk menyoroti

perbedaan antara *entity-entity* dengan menjelaskan satu atau lebih *subclasses* dari sebuah *entity superclass* (Connolly dan Begg, 2002, p432).

a) Generalisasi / Spesialisasi

Konsep dari generalisasi dan spesialisasi dihubungkan dengan tipe-tipe *entity* khusus, yaitu *superclass* dan *subclass*, dan proses pewarisan atribut turunan (Connolly dan Begg, 2002, p360). Dimana *superclass* merupakan induk dari beberapa kelompok-kelompok berbeda keberadaannya pada suatu tipe *entity* sedangkan *subclass* merupakan kelompok bagian yang *distinct* dari sebuah tipe *entity.*

Spesialisasi merupakan proses memaksimalkan perbedaan-perbedaan yang ada diantara anggota dari sebuah *entity* dengan mengidentifikasikan perbedaan-perbedaan karakteristik yang ada. Spesialisasi merupakan pendekatan *top-down* untuk mengidentifikasikan sebuah kumpulan dari *superclass* dan hubungannya dengan *subclass-subclass*nya, dimana kumpulan *subclass* dibasiskan pada beberapa perbedaaan karakteristik. Sebagai contoh setiap pekerjaan biasanya mempunyai jabatan atau *job roles* tertentu seperti manajer, sekretaris maupun *sales.* Jadi dapat dianggap manajer, sekretaris dan *sales* merupakan spesialisasi dari pegawai.

Generalisasi merupakan proses meminimalisasi perbedaan-perbedaan antara *entity* yang ada dengan mengidentifikasikan persamaan-persamaan karakteristiknya. Generalisasi merupakan pendekatan *bottom-up.*

Terdapat dua *constraint* yang mugkin digunakan dalam spesialisasi / generalisasi yaitu (Connolly dan Begg, 2002, p336) :

a) *Participation constraint, constraint* ini menentukan apakah setiap anggota dari *superclass* harus berpartisipasi sebagai anggota dari sebuah *subclass*. Terdapat dua kemungkinan yaitu:

- *Mandatory,* dimana setiap anggota *superclass* harus menjadi anggota dari *subclass*.
- *Optional,* dimana tidak setiap anggota *superclass* harus menjadi anggota *subclass.*

b) *Disjoint Constraint, constraint* yang menjelaskan hubungan antar anggota dari *subclass* dan mengidentifikasikan apakah memungkinkan untuk seorang anggota dari *superclass* menjadi anggota dari satu atau lebih dari *subclass.*

Terdapat dua kemungkinan yaitu :

- *Or,* dimana setiap angota *superclass* hanya boleh menjadi salah satu anggota *subclass.*
- *And, dimana* setiap anggota *superclas*s boleh menjadi anggota lebih dari satu *subclass*.

b) Agregasi

Agregasi merupakan representasi dari sebuah relasi mempunyai sebuah (‘*has-a relationship*’) atau merupakan bagian dari *(‘is-part-of relationship’)* diantara tipe-tipe *entity,* dimana salah satu direpresentasikan sebagai keseluruhan *(‘whole’)* dan yang lainnya menjadi bagian *(‘part’).* Dalam agregasi *‘whole’* merupakan representasi dari *entity* yang lebih besar yang mengandung *entity* yang lebih kecil *‘part’*.

c) *Komposisi*

Komposisi merupakan bentuk spesifik dari agregasi yang mempresentasikan sebuah hubungan antar *entity*, dimana terdapat kepemilikan yang kuat selamanya diantara '*whole'* dan '*part'*.

Pada *compotition*, '*whole'* bertanggung jawab atas penempatan dari '*part'*, yang berarti bahwa *compotition* harus mengantur penciptaan dan penghilangan dari '*part*'nya. Dengan kata lain, sebuah objek hanya boleh menjadi bagian dari sebuah *compotition* pada satu waktu.

- Langkah 1.7 Periksa model untuk *redundancy*

  Pada langkah ini, periksa model data konseptual lokal dengan objektif yang spesifik untuk mengidentifikasikan apakah ada terjadi *redundancy* dan menghapus yang sudah ada (Connolly dan Begg, 2002, p434).

  Dua aktifitas dalam langkah ini antara lain :

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

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

  2. Menghapus *relationship* yang *redundant*

     Sebuah *relationship* adalah *redundant* bila informasi yang sama dapat diperolah melalui *relationship* lain.

- Langkah 1.8 Validasikan konseptual model lokal yang bertentangan dengan transaksi *user*

Untuk meyakinkan bahwa model konseptual lokal mendukung transaksi yang dibutuhkan oleh *view* (Connolly dan Begg, 2002, p435). Ada dua pendekatan yang mungkin untuk memastikan bahwa data model konseptual lokal mendukung transaksi :

1. Menyebutkan transaksi.
2. Menggunakan jalur transaksi.

- Langkah 1.9 Mengkaji ulang model data konseptual lokal dengan *user*

  Untuk me*review* model data konseptual lokal dengan *user* untuk meyakinkan bahwa model adalah representasi yang nyata dari *view*. Model data konseptual lokal meliputi diagram ER dan dokumentasi yang mendukung yang menggambarkan model data.

2. Perancangan Logikal *(Logical Basis data Design)*

   Perancangan basis data logikal merupakan proses pembuatan model dari informasi yang digunakan dalam perusahaan berdasarkan model data yang spesifik, tetapi tidak bergantung pada DBMS tertentu dan masalah fisik lainnya.

   Langkah 2 Membangun dan memvalidasi model data logical untuk setiap *view*

- Langkah 2.1 Menghilangkan fitur-fitur yang tidak sesuai dengan model relasional (*optional*)

  Langkah-langkah yang harus dilakukan antara lain :

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

- Hilangkan tipe relasi *recursive many-to-many (* : *)*
- Hilangkan tipe relasi yang kompleks
- Hilangkan atribut *multi-valued*

- Langkah 2.2 Membuat relasi untuk model data logikal lokal

  Tujuan dari langkah ini adalah untuk membuat suatu relasi untuk model data lokal logikal yang merepresentasikan suatu *entity*, relasi, dan juga atribut yang telah diidentifikasi. Adapun pendeskripsian bagaimana relasi dapat diturunkan dari struktur data model yang ada, antara lain :

  - Tipe *strong entity*
  - Tipe *weak entity*
  - Tipe relasi *binary one-to-many* (1: *)
  - Tipe relasi *binary one-to-one* (1: 1)
  - Relasi rekursif *one-to-one* (1:1)
  - Tipe relasi *superclass / subclass*
  - Tipe relasi *binary many-to-many*
  - Tipe relasi kompleks
  - Atribut *multi-valued.*

- Langkah 2.3 Memvalidasi relasi menggunakan normalisasi

  Proses normalisasi melibatkan beberapa langkah, antara lain :

  - *First Normal Form* (1NF), menghilangkan *repeating group*
  - *Second Normal Form* (2NF), menghilangkan *partial dependencies* pada *primary key*

- *Third Normal Form* (3NF), menghilangkan *transitivel dependencies* pada *primary key*
- *Boyce-Codd Normal Form* (BCNF), menghilangkan anomali-anomali yang masih tersisa dalam *functional dependencies*

- Langkah 2.4 Memvalidasi relasi pada transaksi-transaksi *user*

  Langkah ini dilakukan dengan tujuan untuk memastikan bahwa relasi dalam model data logikal lokal mendukung transaksi-transaksi yang diperlukan dalam penggambaran (*view*).

- Langkah 2.5 Mengdefinisikan *integrity constraints*

  Ada lima tipe *integrity constraints*, antara lain :

  - *Required data*

    Beberapa atribut harus selalu berisi data yang sah sehingga atribut tersebut tidak diperbolehkan menerima *null.*

  - *Attribute domain constraints*

    Setiap atribut mempunyai *domain* yang merupakan sekumpulan nilai yang sah.

  - *Entity Integrity*

    Primary key dari sebuah *entity* tidak dapat menerima *null*.

  - *Referential Integrity*

    Jika *foreign key* berisi nilai maka nilai tersebut harus menunjuk pada *tuple* yang ada pada relasi induk.

Untuk meyakinkan *referential integrity* perlu dispesifikasikan *existence constraint* yang mendefinisikan kondisi dimana *candidate key* atau *foreign key* ditambahkan, diubah atau dihapus.

Jika sebuah *tuple* dari relasi induk dihapus, *referential integrity* hilang jika ada *tuple* anak menunjuk ke *tuple* induk yang dihapus.

Ada beberapa strategi yang dapat digunakan :

a) NO ACTION. Mencegah penghapusan dari relasi induk jika terdapat referensi ke *tuple* anak.

b) CASCADE. Jika *tuple* induk dihapus maka secara otomatis *tuple* anak akan dihapus.

c) SET NULL. Jika *tuple* induk dihapus, maka *foreign key* dari tuple akan menjadi *null*.

d) SET DEFAULT. Jika *tuple* induk dihapus, maka *foreign key* pada semua *tuple* anak akan diberikan nilai *default*.

e) NO CHECK. Jika *tuple* induk dihapus, maka tidak dilakukan apapun untuk meyakinkan bahwa *referential integrity* terjaga.

- *Enterprise constraints*

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

- Langkah 2.6 Meninjau ulang model data logikal lokal dengan *user*

  Langkah ini dilakukan dengan tujuan untuk memastikan model data logikal lokal dan dokumentasi pendukung menggambarkan model yang merupakan representasi nyata dari *view* tersebut.

Langkah 3 Membangun dan Memvalidasi model data logikal global

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

  Hal-hal yang perlu dilakukan antara lain :

  - Meninjau ulang nama dan isi dari tiap *entities* / *relations* dari *candidate key*
  - Meninjau ulang nama dan isi dari *relationships foreign keys*
  - Menggabungkan *entities* / *relations* dari model data lokal
  - Memasukkan (tanpa menggabungkan) *entities* / *relations* yang unik ke dalam setiap model data
  - Menggabungkan *relationships* / *foreign keys* dari model data lokal
  - Memasukkan (tanpa menggabungkan) *relationships* / *foreign keys* yang unik ke dalam setiap model data
  - Mengecek apakah ada *entities* / *relations* dan *relationship / forign keys* yang tertinggal
  - Mengecek *foreign keys*
  - Mengecek *integrity constraints*
  - Menggambarkan diagram ER / relasi global
  - Meng-*update* dokumentasi

- Langkah 3.2 Memvalidasi model data logikal global

  Tujuan dari langkah ini adalah untuk memvalidasi relasi yang dibuat dari model data logikal global dengan menggunakan teknik dari normalisasi dan juga memastikan bahwa relasi yang dibuat mendukung transaksi.

- Langkah 3.3 Mengecak pertumbuhan masa depan

  Untuk menentukan apakah ada perubahan signifikan yang mungkin terjadi di masa depan dan untuk menafsir apakah model data logikal global dapat mengakomodasikan segala perubahan tersebut.

- Langkah 3.4 Meninjau ulang model data logikal global dengn *users*

  Untuk memastikan bahwa model data logikal global merupakan representasi nyata perusahaan.

3. Perancangan Fisikal *(Physical Basis data Design)*

   Perancangan basis data fisikal merupakan proses memproduksi deskripsi dari implementasi basis data pada penyimpanan sekunder. Tahap ini menjelaskan relasi dasar, organisasi *file* dan *index* yang digunakan untuk mencapai efisiensi pengaksesan data dan *integrity constraints* lain yang berhubungan dan ukuran keamanan.

   Langkah 4 Menerjemahkan model data logikal ke dalam target DBMS

- Langkah 4.1 Merancang relasi dasar

  Tujuan dari langkah ini adalah untuk memutuskan bagaimana merepresentasikan relasi dasar yang diidentifikasi dalam model data logikal global pada DBMS yang dipakai. Untuk memulai proses perancangan sistem basisdata fisikal, pertama-tama mengumpulkan dan mengasimilasikan suatu informasi tentang relasi yang dirancang selama perancangan sistem basis data logikal. Informasi yang diperlukan bisa berasal dari kamus data dan definisi dari relasi yang didefinisikan menggunakan *Database Design*

Lenguage (DBDL). Untuk setiap relasi yang diidentifikasi pada model data logikal global, definisinya terdiri dari:

- Nama relasi
- Suatu *list* untuk atribut yang *simple*
- *Primary key, alternate key, dan foreign key*
- Suatu daftar dari atribut turunan dan bagaimana pembuatannya
- Batasan integrasi untuk setiap integrasi untuk setiap *foreign key* yang diidentifikasi.

Dari kamus data, dari setiap atributnya dapat diketahui:

- Domain atribut tersebut yang terdiri dari tipe data, panjang dan berbagai batasan pada *domain.*
- Suatu *optional* nilai *default* untuk atribut.
- Apakah atribut boleh bernilai *null.*

- Langkah 4.2 Merancang representasi dari data turunan

Tujuan dari langkah ini adalah untuk memutuskan bagaimana merepresentasikan suatu data turunan pada model data logikal global pada DBMS yang dipakai.

- Langkah 4.3 Merancang batasan perusahaan

Tujuan dari langkah ini adalah untuk merancang batasan perusahaan untuk DBMS yang dipakai.

Langkah 5 Merancang representasi fisikal

- Langkah 5.1 Analisis transaksi

Tujuan dari langkah ini adalah untuk mengerti fungsi dari suatu transaksi yang mana akan dijalankan pada basis data dan untuk menganalisa transaksi yang penting.

Untuk membuat perancangan sistem basis data fisikal efektif maka perlu dimiliki pengetahuan mengenai transaksi atau *query* yang akan dijalankan di dalam basis data. Hal ini termasuk kuantitatif atau kualitatif informasi. Dalam menganalisa transaksi, dapat diidentifikasi kriteria performasi sebagai berikut:

- Transaksi yang sering digunakan akan berdampak besar terhadap performasi keseluruhan.
- Transaksi yang merupakan transaksi bisnis yang kritis.
- Durasi waktu dalam harian atau mingguan yang akan mendapatkan banyak permintaan pada basis data

- Langkah 5.2 Memilih organisasi *file*

Tujuan dari langkah ini adalah untuk menentukan organisasi file yang efisien untuk setiap relasional data.

Dalam banyak kasus yang ada, suatu relasional DBMS akan memberikan sedikit bahan tanpa pilihan dalam memilih organisasi file, walaupun beberapa akan mempunyai indeks yang spesifik. Bagaimanapun, berikut ini beberapa organisasi file yang ada adalah sebagai berikut:

- *Heap.*
- *Hash.*
- *Indexed Sequential Access Method (ISAM).*

- B^+*-tree*.
- *Clusters*.

- Langkah 5.3 Memilih indeks

  Tujuan dari langkah ini adalah untuk menentukan apakah penambahan indeks akan meningkatkan performasi dari suatu sistem.

  Biasanya pemilihan atribut untuk indeks adalah sebagai berikut:

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

- Langkah 5.4 Memperkirakan kapasitas *disk* yang dibutuhkan untuk menyimpan basis data.

  Tujuan dari langkah ini adalah untuk mengestimasi ukuran kapasitas *disk* yang diperlukan untuk sistem basis data.

Langkah 6 Merancang *user views*

Tujuan dari langkah ini adalah untuk merancang tampilan *user* yang diidentifikasi selama pengumpulan informasi dan analisis dari siklus hidup aplikasi sistem basis data.

Langkah 7 Merancang mekanisme keamanan

Tujuan dari langkah ini adalah untuk merancang ukuran keamanan untuk basis data yang telah dispesifikasi oleh *user*.

2.1.5.5 Seleksi DBMS

Seleksi DBMS *(DBMS Selection)* merupakan proses memilih DBMS yang diperlukan untuk mendukung aplikasi basis data. Tahap-tahap dalam melakukan seleksi DBMS yaitu :

1. Menentukan terminologi dari studi referensi.

   Terminologi dari studi referensi untuk seleksi DBMS dibuat , bersamaan dengan tujuan dan ruang lingkup pembelajaran dan tugas yang perlu dilakukan. Dokumen ini juga mencakup deskripsi dari kriteria (berdasarkan spesifikasi kebutuhan pengguna) yang digunakan untuk mengevaluasi produk DBMS.

2. Mendaftarkan dua atau tiga produk.

3. Mengevaluasi produk.

   Ada banyak fitur yang dapat digunakan untuk mengevaluasi produk DBMS. Untuk tujuan evaluasi, fitur-fitur tersebut dapat dilihat sebagai kelompok atau individual.

4. Merekomendasikan pillihan dan membuat laporan.

   Tahap terakhir dari seleksi DBMS adalah mendokumentasikan proses dan menyediakan pernyataan mengenai kesimpulan dan rekomendasi terhadap salah satu produk DBMS.

2.1.5.6 Perancangan Aplikasi

Perancangan aplikasi *(application design)* adalah proses merancang antar muka dan program aplikasi yang menggunakan dan melakukan proses terhadap basis data. Perancangan basis data dan aplikasi merupakan serangkaian aktifitas paralel dalam *database application lifecycle*. Dalam kebanyakan kasus, hal yang tidak mungkin terjadi adalah menyelesaikan perancangan aplikasi sebelum proses desain basis data selesai. Di sisilain, basis data yang ada untuk mendukung aplikasi dan oleh karena itu harus ada aliran informasi antara desain aplikasi dan desain basis data.

Ada dua aspek dalam desain aplikasi yaitu perancangan transaksi *(transaction design)* dan perancangan antar muka *(user interface design).* Transaksi adalah sebuah aksi atau serangkaian aksi yang dibawa oleh pemakai tunggal atau program aplikasi, yang mengakses dan mengubah isi dari basis data. Sebuah transaksi dapat terdiri dari beberapa operasi. Tujuan dari desain transaksi adalah mendefinisikan dan mendokumentasikan karakteristik level atas dari transaksi yang dibutuhkan oleh basis data, yang meliputi :

1. Data yang digunakan oleh transaksi.
2. Karakteristik fungsional dari transaksi.
3. Hasil *(output)* dari transaksi.
4. Kegunaannya untuk pemakai.
5. Perkiraan jumlah penggunaan.

Aktifitas ini harus dilakukan pada awal dari proses desain untuk memastikan bahwa implementasi basis data dapat mendukung semua transaksi yang dibutuhkan. Ada tiga tipe transaksi yaitu :

1. *Retrieval Transactions* digunakan untuk mendapatkan data untuk ditampilkan pada layar atau memproduksi laporan *(report).*
2. *Update Transactions* digunakan untuk memasukkan *record* baru, menghapus *record* lama, atau melakukan modifikasi pada *record* yang terdapatpada basis data.
3. *Mixed Transactions* mencakup *retrieval transactions* dan *update transactions.*

2.1.5.7 *Prototyping*

Prototyping merupakan proses membuat model kerja dari aplikasi *database*. Sebuah *prototype* merupakan model kerja yang secara normal tidak memiliki semua fitur yang dibutuhkan atau menyediakan semua fungsi dari *final system*. Tujuan utama dalam membangun *prototype* dari aplikasi basis data adalah memungkinkan *user* untuk menggunakan *prototype* untuk mengidentifikasi fitur-fitur dari sistem mana yang bekerja dengan baik, tidak cukup, atau dimungkinkan untuk menyarankan pengembangan atau fitur baru pada aplikasi basis data.

Ada dua strategi *prototyping* yang umum digunakan sekarang, yaitu *requirement prototyping* dan *evolutionary protyping*. *Requirement prototyping* adalah *prototype* untuk menetapkan kebutuhan dari tujuan aplikasi basis data dan ketika kebutuhan sudah terpenuhi, *prototype* tidak digunakan lagi atau dibuang

(discard). Evolutionary prototype menggunakan tujuan selanjutnya dikembangkan menjadi aplikasi basis data yang bekerja.

2.1.5.8 Implementasi

Implementasi merupakan realisasi fisikal dari desain basis data dan aplikasi. Implementasi basis data dilakukan dengan menggunakan *Data Definition Language* (DDL) dari DBMS yang dipilih atau *Graphical User Interface* (GUI). Bagian dari aplikasi program dan transaksi basis data, yang diimplementasikan menggunakan *Data Manipulation Language* (DML).

2.1.5.9 Konversi dan Proses Memuat Data

Konversi dan proses memuat data *(data conversion and loading)* melakukan pemindahan data yang ada ke dalam basis data yang baru dan melakukan konversi aplikasi yang ada untuk dijalankan pada basis data yang baru. Tahap ini diperlukan ketika sistem basis data yang baru menggantikan sistem yang lama.

2.1.5.10 Pengujian

Pengujian *(testing)* merupakan proses mengeksekusi program aplikasi dengan tujuan untuk menemukan kesalahan *(error).* Tahap ini dilakukan dengan menggunakan strategi perencanaan tes secara hati-hati dan data yang realistik.

2.1.5.11 Pemeliharaan Operasional

Pemeliharaan operasional *(operational maintenace)* merupakan proses mengawasi dan memelihara sistem setelah instalasi. Tahap ini meliputi aktifitas-aktifitas berikut :

1. Mengawasi performa dari sistem.
2. Memelihara dan mengembangkan aplikasi basis data (jika diperlukan).

2.1.6 *Structured Query Language* (SQL)

Secara ideal, bahasa basis data harus mengijinkan pengguna untuk membuat basis data dan struktur hubungan, menampilkan tugas manajemen basis data (*insert*, modifikasi, penghapusan data) dan menyediakan modifikasi, penghapusan data dan menyediakan *query* yang sederhana maupun kompleks. SQL merupakan bahasa basis data yang memenuhi persyaratan di atas.

SQL merupakan contoh dari *transform-oriented language* atau bahasa yang dirancang untuk menggunakan relasi dalam mengubah input menjadi *output* yang diinginkan.

2.1.6.1 *Data Definition Language (DDL)*

Data Definition Language (DDL) merupakan bahasa yang mengijinkan DBA atau pengguna untuk menggambarkan dan memberikan nama terhadap entitas, atribut, dan hubungan yang dibutuhkan untuk aplikasi, bersama-sama dengan batasan-batasan integritas dan keamanan yang dibutuhkan. Skema basis

data dispesifikasikan oleh sekumpulan definisi yang diekspresikan dengan tujuan untuk bahasa tertentu yang disebut *Data Definition Language.*

DDL digunakan untuk mendefinisikan sebuah skema atau untuk memodifikasi skema yang sudah ada. DDL tidak bisa digunakan untuk memanipulasi data.

Hasil pengkompilasian pernyataan DDL adalah sekumpulan tabel yang disimpan dalam *file-file* khusus secara bersama-sama yang disebut *system catalog*. Sistem *catalog* mengintegrasikan *metadata*, yaitu data yang menggambarkan objek-objek dalam basis data dan membuat objek-objek tersebut lebih mudah diakses atau dimanipulasi. *Metadata* mengandung definisi *record, item data,* dan objek-objek lain yang menarik bagi pengguna atau dibutuhkan oleh DBMS. Istilah kamus data dan direktori data juga bisa digunakan untuk menggambarkan *system catalog*.

2.1.6.2 *Data Manipulation Language (DML)*

DML adalah sebuah bahasa yang menyediakan sekumpulan operasi untuk mendukung operasi manipulasi data dasar terhadap data yang disimpan di dalam basis data. Operasi manipulasi data biasanya mencakup hal-hal berikut ini :

- Memasukkan data baru ke dalam basis data
- Modifikasi data yang disimpan di dalam basis data
- Memperoleh data yang ada di dalam basis data
- Menghapus data dari basis data

Bagian dari DML yang melibatkan perolehan data disebut *query language*. *Query language* dapat didefinisikan sebagai bahasa level tinggi bertujuan khusus yang digunakan untuk memenuhi permintaan yang beraneka ragam untuk data-data yang disimpan di dalam basis data.

DML dapat dibedakan menjadi dua tipe yaitu DML procedural dan DML non-procedural. DML procedural adalah sebuah bahasa yang mengijinkan pengguna memberitahukan sistem data apa yang dibutuhkan dan secara tepat bagaimana memperoleh data. DML non-procedural adalah sebuah bahasa yang mengijinkan pengguna untuk menentukan data apa yang dibutuhkan tanpa memperhatikan bagaimana data diperoleh.

2.1.7 Normalisasi

Menurut Connoly dan Begg (2005, p388), normalisasi adalah sebuah teknik untuk menghasilkan sekumpulan relasi dengan properti-properti yang sesuai dengan persyaratan data yang diberikan sebuah perusahaan. Tujuan dari normalisasi adalah untuk mengidentifikasi sekumpulan relasi yang mendukung persyaratan data sebuah perusahaan. Karakteristik sekumpulan relasi yang sesuai meliputi berikut ini :

- Jumlah minimal atribut yang diperlukan untuk mendukung persyaratan data perusahaan;
- Atribut-atribut dengan relasi logikal yang dekat (digambarkan sebagai ketergantungan fungsional) ditemukan di dalam relasi yang sama;

- Redundansi minimal dengan setiap atribut direpresentasikan hanya sekali dengan pengecualian penting atribut yang membentuk sebagian atau seluruh *foreign keys*, yang esensial untuk menggabungkan relasi-relasi yang berhubungan.

Manfaat menggunakan basis data yang mempunyai sekumpulan relasi yang sesuai adalah bahwa basis data tersebut akan lebih mudah diakses dan dipelihara datanya oleh pengguna, dan membutuhkan ruang penyimpanan minimal pada komputer.

2.1.7.1 Proses Normalisasi

Normalisasi adalah sebuah teknik formal untuk menganalisis relasi berdasarkan primary key relasi-relasi tersebut (atau *candidate key*) dan ketergantungan fungsional. Teknik itu melibatkan rangkaian aturan yang bisa digunakan untuk menguji relasi individual sehingga basis data bisa dinormalisasikan ke derajat apapun. Ketika persyaratan tidak ditemukan, relasi yang tidak sesuai persyaratan harus didekomposisi menjadi relasi-relasi yang secara individual memenuhi persyaratan normalisasi.

Tiga bentuk normal yang pada awalnya diusulkan disebut *First Normal Form* (1NF), *Second Normal Form* (2NF), dan *Third Normal Form* (3NF). Normalisasi sering dieksekusi sebagai rangkaian langkah.

Setiap langkah berkorespondensi ke sebuah normal form tertentu yang memiliki properti yang diketahui. Untuk model data relasional, perlu diketahui bahwa hanya *First Normal Form* (1NF) yang penting dalam membuat relasi.

Semua normal form selanjutnya adalah langkah opsional. Tetapi untuk menghindari *update anomaly*, direkomendasikan bahwa kita melanjutkannya setidaknya sampai *Third Normal Form* (3NF).

1. *First Normal Form (1NF)*

Unnormalized Form (UNF) adalah sebuah tabel yang mengandung satu atau lebih repeating groups. *First Normal Form* (1NF) adalah sebuah relasi di dalam mana titik potong setiap baris dan kolom mengandung satu dan hanya satu nilai.

Proses awal dari normalisasi adalah memindahkan data dari sumber ke dalam bentuk tabel dengan format baris dan kolom. Dalam bentuk format ini, tabel ini merupakan UNF dan mengarah sebagai *unnormalized table*. Untuk mengubah *unnormalized table* menjadi *first normal form* (1NF), kita harus mengidentifikasi dan menghilangkan *repeating groups* dari *table*. *Repeating groups* merupakan sebuah atribut atau kelompok atribut dalam tabel yang mengakibatkan munculnya *multiple values* untuk sebuah kejadian tunggal dari atribut *nominated key* untuk tabel tersebut. Ada dua pendekatan umum untuk menghilangkan kelompok-kelompok yang berulang dari *unnormalized table* yaitu :

a) Menghilangkan *repeating groups* dengan memasukkan data yang sesuai dalam kolom kosong di mana baris-barisnya mengandung *repeating data*.
b) Menghilangkan *repeating groups* dengan mengganti *repeating data* bersamaan dengan duplikasi dari kunci atribut yang asli.

2. Second Normal Form (2NF)

Second Normal Form (2NF) dibuat berdasarkan konsep *full fuctional dependency*. *Second Normal Form* (2NF) adalah sebuah relasi yang berada dalam *first normal form* dan setiap atribut *non-primary-key* secara penuh bergantung secara fungsional pada *primary key.*

Second Normal Form berlaku pada relasi-relasi dengan *composite keys*, yaitu, relasi-relasi dengan *primary key* yang dibentuk dari dua atau lebih atribut. Relasi yang memiliki sebuah *primary key* tunggal secara langsung sudah merupakan 2NF. Relasi yang tidak berada dalam 2NF bisa mengalami *update anomaly*. Normalisasi relasi 1NF ke 2NF melibatkan penghilangan ketergantungan parsial. Jika terdapat ketergantungan parsial, kita menghilangkan *functionally* dependent attributes dari relasi dengan menempatkannya ke dalam relasi yang baru bersama dengan duplikasi determinannya.

3. *Third Normal Form (3NF)*

Third Normal Form (3NF) adalah sebuah relasi dalam bentuk 1NF dan 2NF dimana tidak ada atribut *non-primary-key* yang bergantung secara transitif terhadap *primary key*. Proses normalisasi dari relasi 2NF ke 3NF melibatkan penghapusan ketergantungan transitif. Jika terdapat ketergantungan transitif, kita menghilangkan atribut yang bergantung secara transitif dari relasi dengan menggantikan atribut dalam relasi yang baru bersamaan dengan duplikasi dari determinannya.

2.1.8 *Entity Relationship Modelling* (*ER Modelling*)

Menurut Conolly dan Begg (2005,p342), *ER Modelling* adalah model onseptual yang menjabarkan hubungan antara penyimpan data dan hubungan data. ER modeling merupakan sebuah pendekatan *top-down* pada desain basis data yang dimulai dengan mengidentifikasi data penting yang disebut entitas (*entity*) dan hubungan (*relationship*) ntara data yang harus direpresentasikan dalam model. Kemudian itambahkan detil lebih sebagai informasi yang kita ingin simpan tentang entitas dan hubungannya yang disebut atribut (*attributes*) dan segala pada *entities, relationship, dan attributes*.

1. Entity Types

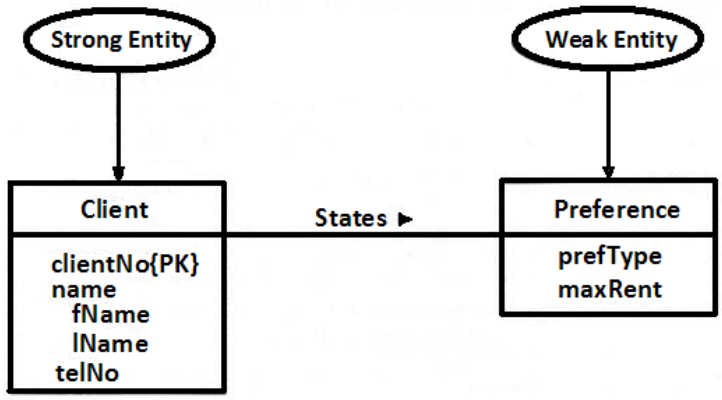
Entity type adalah kumpulan dari objek-objek dengan sifat *(property)* yang sama, yang diidentifikasi oleh *enterprise* sebagai eksistensi yang independen. Keberadaannya dapat berupa fisik maupun abstrak *Entity Occurrence* yaitu pengidentifikasian objek yang unik dari sebuah entitas diidentifikasi dan disertakan *property*-nya. Entitas dapat diklasifikasikan sebagai berikut:

- *Strong / Parent / Owner / Dominant entity type*

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

- *Weak / Child / Subordinate / Dependent entity type*

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

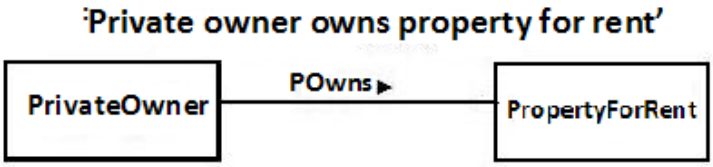


Gambar 2.6 *Strong Entity dan Weak Entity (Connoly dan Begg, 2005, p355)*

2. Relationship Types

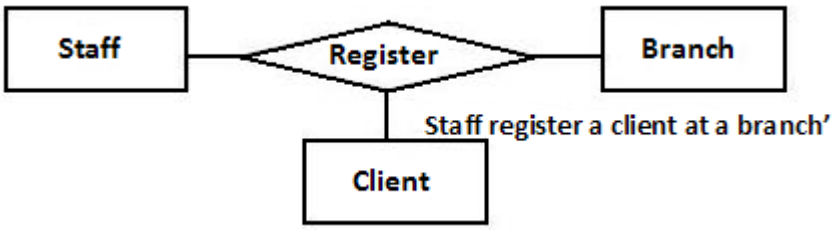
Relationship Type merupakan sebuah set asosiasi yang mempunyai arti antar *entity* dan diberi nama sesuai dengan fungsinya. *Relationship Occurrence*, yaitu suatu gabungan yang diidentifikasi secara unik yang meliputi suatu kejadian dari tiap *entity type* yang berpartisipasi. *Degree of a relationship type* adalah jumlah entitas yang berpartisipasi dalam suatu *relationship*. Derajat *relationship* terdiri dari :

- *Binary relationship*, keterhubungan antar dua tipe entitas. Contoh : *binary relationship* antara *PrivateOwner* dengan *PropertyForRent* yang disebut *POwns*.



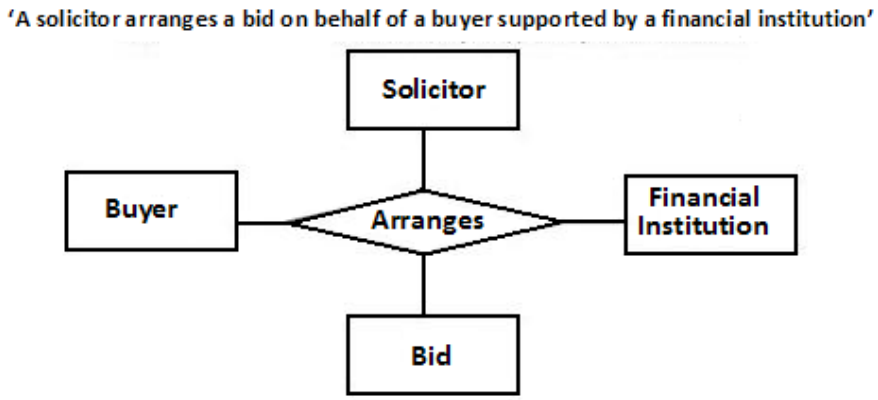
Gambar 2.7 *Binary Relationship (Connoly dan Begg, 2005, p348)*

- *Ternary relationship*, keterhubungan antar tiga tipe entitas. Contoh *ternary relationship* yang dinamakan *Registers*. Relasi ini melibatkan tiga tipe entitas yaitu *Staff*, *Branch*, dan *Client*. Relasi ini menggambarkan *staff* mendaftarkan *client* pada *branch*.



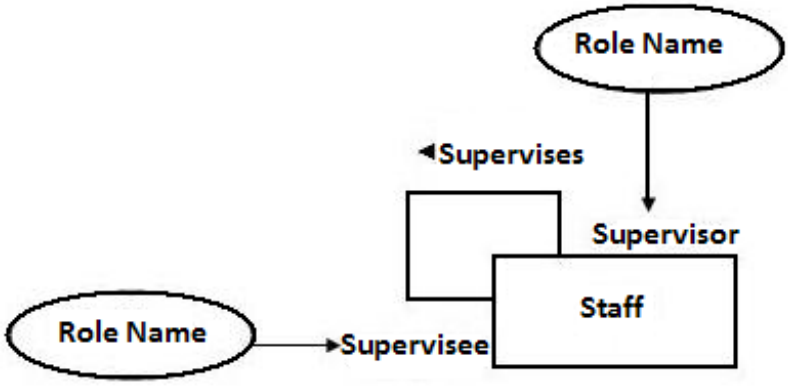
Gambar 2.8 *Ternary Relationship (Connoly dan Begg, 2005, p348)*

- *Quaternary relationship*, keterhubungan antar empat tipe entitas. Contoh *quarternary relationship* yang dinamakan *Arranges*. Relasi ini melibatkan 4 tipe entitas yaitu *Buyer*, *Solicitor*, *Financial Institution*, dan *Bid*. Relasi ini menggambarkan *buyer* diberi masukan oleh *Solicitor*, dan didukung oleh *Financial Institution*, dan melakukan penawaran (*bid*).



Gambar 2.9 *Quaternary Relationship (Connoly dan Begg, 2005, p349)*

- *Unary relationship*, keterhubungan antar satu tipe entitas, di mana tipe entitas tersebut berpartisipasi lebih dari satu kali dengan peran yang berbeda. Kadang disebut juga *recursive relationship*. *Relationship* dapat diberikan *role* names untuk mengidentifikasi keterkaitan tipe entitas dalam *relationship*. Contohnya adalah *Staff* yang berperan menjadi *supervisor* dan *Staff* yang diawasi.



Gambar 2.10 *Hubungan Rekursif (Connoly dan Begg, 2005, p349)*

3. Attributes

Merupakan sifat-sifat *(property)* dari sebuah *entity* atau *relationship*. Contohnya : sebuah *entity Staff* digambarkan oleh atribut *staffNo*, *name*, *position*, and *salary*. Atribut-atribut tersebut mempunyai nilai yang menggambarkan *entity ocurrence* dan merepresentasikan bagian penting dari data yang tersimpan dalam basis data. Atribut *Domain* adalah himpunan nilai yang diperbolehkan untuk satu atau lebih atribut.Macam-macam atribut :

- *Simple Attribute*, yaitu atribut yang terdiri dari satu komponen tunggal dengan keberadaan yang independen dan tidak dapat dibagi menjadi bagian yang lebih kecil.
- *Composite Attribute*, yaitu atribut yang terdiri dari beberapa komponen, di mana masing-masing komponen memiliki keberadaan yang independen. Misalkan atribut *Address* dapat terdiri dari *Street*, *City*, *PostCode*.
- *Single-valued Attribute*, yaitu atribut yang mempunyai nilai tunggal untuk setiap kejadian.
- *Multi-valued Attribute*, yaitu atribut yang mempunyai nilai untuk setiap kejadian.
- *Derived Attribute*, yaitu atribut yang memiliki nilai yang dihasilkan dari satu atau beberapa atribut lainnya.

4. Key

Candidate Key, yaitu jumlah minimal atribut-atribut yang dapat mengidentifikasikan setiap kejadian/*record* secara unik. *Primary key*, yaitu

Candidate Key yang dipilih untuk mengidentifikasikan setiap kejadian/*record* dari suatu entitas secara unik. *Composite Key*, yaitu *Candidate Key* yang terdiri dari dua atau lebih atribut.

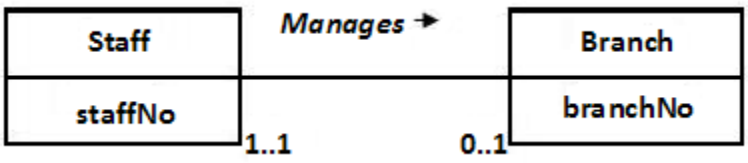
5. Structural Constraints

Constraints harus merefleksikan kendala relasi seperti anggapan '*real word*'. Contohnya seperti *constraints* yang menyertakan kebutuhan *PropertyForRent* yang harus mempunyai *owner* dan tiap *branch* harus mempunyai *staff*. Jenis *constraint* utama pada relasi disebut *multiplicity*.

Batasan utama pada *relationship* disebut *multiplicity*, yaitu jumlah (atau *range*) dari kejadian yang mungkin terjadi pada suatu entitasyang terhubung ke satu kejadian dari entitas lain yang berhubungan melalui suatu relationship. *Relationship* yang paling umum adalah *binary relationship*. Macam-macam *binary relationship*:

- *One to One*

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



Gambar 2.11 Contoh *One to One Relationship (Connoly dan Begg, 2005, p358)*

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

- *One to Many / Many to One*

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

Gambar 2.12 Contoh *One to Many Relationship*

(Connoly dan Begg, 2005, p359)

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

- *Many to Many*

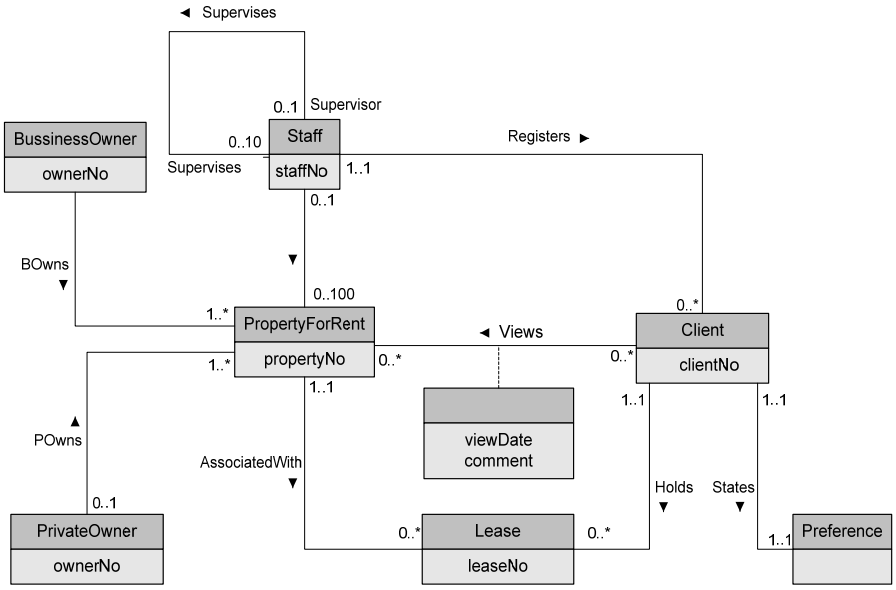
  Contoh : hubungan antara *Newspaper* dan *PropertyForRent* yang memiliki asosiasi *Advertises*.

Gambar 2.13 Contoh *Many to Many Relationship*

(Connoly dan Begg, 2005, p360)

Dari diagram di atas maka arti dari 0..* adalah tiap *PropertyForRent* diiklankan dalam paling sedikit 0 atau lebih dari satu koran. Sedangkan arti dari 1..* adalah tiap koran mengiklankan paling sedikit 1 atau lebih dari satu *PropertyForRent*. *Multiplicity* sebenarnya terdiri dari dua *contraint* yang terpisah yaitu *cardinality* dan *participation*.

- *Cardinality*, menjelaskan jumlah maksimal *ocurrence relationship* yang diperbolehkan untuk satu entitas yang berpartisipasi dalam satu tipe relasi.
- *Participation*, menentukan apakah seluruh atau sebagian *entity ocurrence* yang berpartisipasi dalam suatu relasi.



Gambar 2.14 Contoh ERD

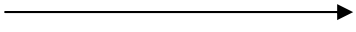
2.1.9 *State Transition Diagram* (STD)

Menurut Whitten, Bentley, Dittman (2004, p636), *State Transition Diagram* (STD) digunakan untuk menggabmarkan urutan dan variasi *screen* yang dapat terjadi selama satu sesi pengguna. *State Transition Diagram* (STD) merupakan suatu diagram yang menggambarkan bagaimana *state* dihubungkan dengan *state* yang lain pada satu waktu. *State Transition Diagram* (STD) menunjukkan bagaimana sistem bekerja sebagai akibat dari kejadian eksternal. Untuk melakukan hal itu, *State Transition Diagram* (STD) menampilkan model yang bermacam-macam dari tindakan sebuah sistem dan dibuat dari *state* ke *state*.

Komponen dasar dari STD adalah:

1. 

   Menyatakan *state* atau kondisi dari suatu *system*. *state* terdiri dari dua macam, yaitu *initial state* / *state* awal dan *final state*/ *state* akhir. *Final state* dapat terdiri dari beberapa *state*, tetapi *initial state* tidak boleh lebih dari satu

2. 

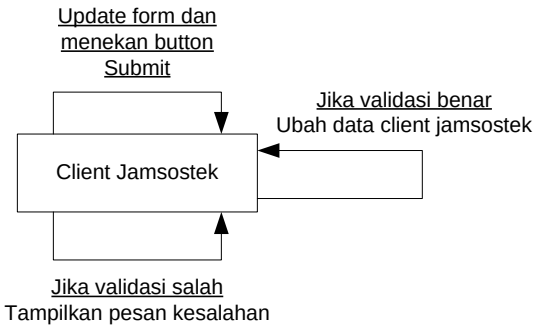
   Menyatakan perubahan kondisi dari suatu *system*. Digambarkan untuk menghubungkan keadaan *system* yang berkaitan.

3. Kondisi dan aksi

   Kondisi: menyatakan suatu kejadian pada lingkungan eksternal yang

dapat dideteksi oleh suatu system. Misalnya : suatu sinyal/ data.

Aksi: menyatakan sesuatu yang dilakukan oleh *system* apabila terjadi perubahan *state* / merupakan suatu reaksi terhadap kondisi. Aksi akan menghasilkan *output, message display* pada *monitor* dan menghasilkan kalikulasi.



Gambar 2.15 Contoh STD

2.2 Teori Khusus

2.2.1 *Outsource*

2.2.1.1 Pengertian *Outsourcing* (Alih Daya)

Outsourcing (Alih Daya) diartikan sebagai pemindahan atau pendelegasian beberapa proses bisnis kepada suatu badan penyedia jasa, dimana badan penyedia jasa tersebut melakukan proses administrasi dan manajemen berdasarkan definisi serta kriteria yang telah disepakati oleh para pihak.

Outsourcing (Alih Daya) dalam hukum ketenagakerjaan di Indonesia diartikan sebagai pemborongan pekerjaan dan penyediaan jasa tenaga kerja.

Pengertian *outsourcing* (Alih Daya) secara khusus didefinisikan oleh Maurice F Greaver II, pada bukunya *Strategic Outsourcing, A Structured*

Approach to Outsourcing: Decisions and Initiatives, dijabarkan sebagai *"Strategic use of outside parties to perform activities, traditionally handled by internal staff and respurces."* Yang memandang *Outsourcing* (Alih Daya) sebagai tindakan mengalihkan beberapa aktivitas perusahaan dan hak pengambilan keputusannya kepada pihak lain (*outside provider*), dimana tindakan ini terikat dalam suatu kontrak kerjasama.

Muzni Tambusai, Direktur Jenderal Pembinaan Hubungan Industrial Departemen Tenaga Kerja dan Transmigrasi mendefinisikan pengertian *outsourcing* (Alih Daya) sebagai memborongkan satu bagian atau beberapa bagian kegiatan perusahaan yang tadinya dikelola sendiri kepada perusahaan lain yang kemudian disebut sebagai penerima pekerjaan.

(http://panmohamadfaiz.com/category/outsourcing/)

2.2.2 Gaji

2.2.2.1 Pengertian Gaji

Menurut Mulyadi (2001, p373), "Gaji umumnya merupakan pembayaran atas penyerahan jasa yang dilakukan oleh karyawan, yang umumnya dibayarkan secara tetap per bulan. Sedangkan upah umumnya merupakan pembayaran atas penyerahan jasa yang dilakukan oleh karyawan pelaksana (buruh), yang dibayarkan berdasarkan hari kerja, jam kerja, atau jumlah satuan produk yang dihasilkan oleh karyawan."

Menurut Warren *et al* (2005,p552), dalam akuntansi, istilah gaji diartikan sebagai jumlah tertentu yang dibayarkan kepada karyawan untuk jasa yang

diberikan selama periode tertentu. Gaji merupakan hal yang penting karena beberapa alasan yaitu:

- Para karyawan sangat sensitif terhadap kesalahan atau ketidakwajaran dalam gaji.
- Gaji merupakan hal yang diatur dengan berbagai peraturan pemerintah federal dan negara bagian
- Gaji dan pajak yang terkait dengan gaji mempunyai efek yang signifikan terhadap laba bersih pada sebagian besar usaha.

Menurut Romney dan Steinbart (2006, p495), terdapat 7 (tujuh) aktivitas utama dalam siklus penggajian, yaitu :

a) Meng-*update master file* penggajian

Aktivitas pertama dalam siklus penggajian meliputi peng-*update*-an *master file* penggajian untuk mencerminkan beberapa tipe perubahan dalam penggajian seperti mempekerjakan karyawan baru, pemecatan,perubahan dalam gaji atau perubahan dalam pemotongan gaji.

b) Meng-*update* tarif dan pengurangan pajak

Departemen penggajian membuat perubahan ini, tetapi jarang terjadi. Perubahan dilakukan apabila departemen panggajian menerima perubahan baru dalam tarif pajak atau pengurangan gaji lainnya dari berbagai unit pemerintahan dan perusahaan asuransi.

c) Memvalidasi waktu dan data kehadiran

Untuk karyawan yang digaji berdasarkan jam kerja biasanya perusahaan menggunakan *time card* untuk mencatat data absensi karyawan. Sedangkan untuk karyawan yang memiliki gaji tetap biasanya mempunyai tambahan gaji dari komisi, insentif maupun bonus. Waktu absensi bagi karyawan dengan gaji tetap digunakan oleh *supervisor* untuk me-*monitor* kinerja dari karyawan tersebut.

d) Menyiapkan gaji

Jumlah gaji yang akan dibayarkan didapat dari *master file* penggajian. Untuk karyawan yang berdasarkan jam kerja, jumlah jam kerja dikalikan dengan tarif upah dan ditambahkan bonus atau lembur jika ada. Untuk karyawan gaji tetap akan diberikan gaji per bulan ditambah lembur maupun bonus jika ada. Kemudian seluruh gaji kotor akan dikurangkan dengan pajak dan pengurangan sukarela seperti biaya asuransi, dana pensiun dan lain-lain. Semua pengurangan ini di-*update* ke *master file* penggajian dengan alasan untuk mengetahui kapan pengurangan pajak dilakukan dan untuk memastikan jumlah pajak dan pengurangan lainnya yang dibayarkan ke perusahaan asuransi dan lembaga pemerintahan. Pada akhirnya *payroll register* yang digunakan mengotorisasi pemindahan dana ke rekening penggajian perusahaan dan cek pembayaran gaji karyawan dicetak.

e) Membayar gaji

Langkah selanjutnya adalah pembayaran gaji kepada karyawan baik melalui cek maupun *transfer* ke rekening bank dengan prosedur berikut: *Payroll register* dikirim ke departemen hutang untuk di*review* dan disetujui. *Voucher* pembayaran gaji disiapkan utnuk mengotorisasi pemindahan dana dari rekening perusahaan ke rekening penggajian. Kemudian *payroll register* dan *voucher* pembayaran gaji dikirim ke kasir untuk menyiapkan dan menandatangani cek pemindahan dana ke rekening penggajian.

f) Menghitung pajak dan keuntungan yang dibayarkan kepada karyawan

Menyediakan jasa tambahan untuk perhitungan pajak, dan pensiun, asuransi dari masing-masing karyawan perusahaan.

g) Membayar pajak dari gaji dan pengurangan lain-lain

Aktivitas terakhir dari proses penggajian adalah membayar hutang pajak penggajian dan pengurangan sukarela lainnya dari setiap karyawan. Perusahaan harus secara periodik menyiapkan cek atau pemindahan dana elektronik untuk membayar utang pajak yang ada sesuai dengan jangka waktu yang ditetapkan pemerintah.

2.2.3 Lembur

Peraturan yang dipakai untuk menghitung upah lembur adalah Kepmen 102 tahun 2004. Menghitung upah lembur agak rumit bagi yang belum biasa menghitungnya, berikut rinciannya:

1. Untuk menghitung upah sejam adalah: 1/173 X upah sebulan
2. Apabila kerja lembur dilakukan pada hari kerja biasa:
    a. Untuk jam lembur pertama dibayar sebesar 1,5 x upah sejam.
    b. Untuk jam lembur selebihnya dibayar sebesar 2 x upah sejam
3. Apabila kerja lembur dilakukan pada hari istirahat mingguan dan/atau hari libur resmi untuk waktu kerja 6 hari kerja seminggu maka:
    a. Perhitungan upah kerja lembur untuk 7 jam pertama dibayar 2 kali upah sejam
    b. Jam kedelapan dibayar 3 kali upah sejam
    c. Jam lembur kesembilan dan kesepuluh dibayar 4 kali upah sejam.
    d. Apabila hari libur resmi jatuh pada hari kerja terpendek, perhitungan upah lembur 5 jam pertama dibayar 2 kali upah sejam, jam keenam 3 kali upah sejam, dan jam lembur ketujuh dan kedelapan 4 kali upah sejam.

4. Apabila kerja lembur dilakukan pada hari istirahat mingguan dan/atau hari libur resmi untuk waktu kerja 5 hari kerja seminggu maka:
    a. Perhitungan upah kerja lembur untuk 8 jam pertama dibayar 2 kali upah sejam
    b. Jam kesembilan dibayar 3 kali upah sejam
    c. Jam kesepuluh dan kesebelas 4 kali upah sejam.

2.2.4 Pajak

2.2.4.1 Pengertian Pajak

Pajak adalah iuran rakyat kepada kas negara berdasarkan undang-undang sehingga dapat dipaksakan dengan tiada mendapat balas jasa secara langsung. Pajak dipungut penguasa berdasarkan norma-norma hukum untuk menutup biaya produksi barang-barang dan jasa kolektif untuk mencapai kesejahteraan umum.

Terdapat bermacam-macam batasan atau definisi tentang "pajak" yang dikemukakan oleh para ahli diantaranya adalah :

- Prof. Dr. P. J. A. Adriani

  Pajak adalah iuran masyarakat kepada negara (yang dapat dipaksakan) yang terutang oleh yang wajib membayarnya menurut peraturan-peraturan umum (undang-undang) dengan tidak mendapat prestasi kembali yang langsung dapat ditunjuk dan yang gunanya adalah untuk membiayai pengeluaran-pengeluaran umum berhubung tugas negara untuk menyelenggarakan pemerintahan.

- Prof. Dr. H. Rochmat Soemitro SH

  Pajak adalah iuran rakyat kepada Kas Negara berdasarkan undang-undang (yang dapat dipaksakan) dengan tiada mendapat jasa timbal (kontra prestasi) yang langsung dapat ditunjukkan dan yang digunakan untuk membayar pengeluaran umum. Definisi tersebut kemudian dikoreksinya yang berbunyi sebagai berikut: Pajak adalah peralihan kekayaan dari pihak rakyat kepada Kas Negara untuk membiayai pengeluaran rutin dan surplusnya digunakan untuk *public saving* yang merupakan sumber utama untuk membiayai *public investment*.

- Sommerfeld Ray M., Anderson Herschel M., & Brock Horace R

  Pajak adalah suatu pengalihan sumber dari sektor swasta ke sektor pemerintah, bukan akibat pelanggaran hukum, namun wajib dilaksanakan, berdasarkan ketentuan yang ditetapkan lebih dahulu, tanpa mendapat imbalan yang langsung dan proporsional, agar pemerintah dapat melaksanakan tugas-tugasnya untuk menjalankan pemerintahan.

  (http://id.wikipedia.org/wiki/Pajak)

2.2.4.2 Pajak Pertambahan Nilai (PPN)

Indonesia menganut sistem tarif tunggal untuk PPN, yaitu sebesar 10 persen. Dasar hukum utama yang digunakan untuk penerapan PPN di Indonesia adalah Undang-Undang No. 8/1983 berikut revisinya, yaitu Undang-Undang No. 11/1994 dan Undang-Undang No. 18/2000.

2.2.5 Jamsostek

Jaminan Sosial Tenaga Kerja adalah program publik yang memberikan perlindungan bagi tenaga kerja untuk mengatasi resiko sosial ekonomi tertentu yang penyelenggarannya menggunakan mekanisme asuransi sosial.

2.2.5.1 Dasar Hukum

Program JAMSOSTEK kepesertaannya diatus secara wajib melalui Undang-Undang No. 3 tahun 1992 tentang Jaminan Sosial Tenaga Kerja, sedangkan pelaksanaannya dituangkan dalam Peraturan Pemerintah No. 14 tahun 1993, Keputusan Presdien No. 22 tahun 1993 dan Peraturan Menteri Tenaga Kerja No. Per.05/MEN/1993.

2.2.5.2 Jenis Program Jamsostek

Undang-Undang No. 3 Tahun 1992 baru mengatur jenis program Jaminan Kecelakaan Kerja, Jaminan Hari Tua, Jaminan Kematian dan Jaminan Pemeliharaan Kesehatan.

2.2.5.3 Program Jaminan Hari Tua

Program Jaminan Hari Tua ditujukan sebagai pengganti terputusnya penghasilan tenaga kerja karena meninggal, cacat, atau hari tua dan diselenggarakan dengan sistem tabungan hari tua. Program Jaminan Hari Tua

memberikan kepastian penerimaan penghasilan yang dibayarkan pada saat tenaga kerja mencapai usia 55 tahun atau telah memenuhi persyaratan tertentu.

Iuran Program Jaminan Hari Tua:

- Ditanggung Perusahaan = 3,7%
- Ditanggung Tenaga Kerja = 2%

2.2.5.4 Jaminan Kecelakaan Kerja

Kecelakaan kerja termasuk penyakit akibat kerja merupakan risiko yang harus dihadapi oleh tenaga kerja dalam melakukan pekerjaannya. Untuk menanggulangi hilangnya sebagian atau seluruh penghasilan yang diakibatkan oleh adanya risiko-risiko sosial seperti kematian atau cacat karena kecelakaan kerja baik fisik maupun mental, maka diperlukan adanya jaminan kecelakaan kerja. Kesehatan dan keselamatan tenaga kerja merupakan tanggung jawab pengusaha sehingga pengusaha memiliki kewajiban untuk membayar iuran jaminan kecelakaan kerja yang berkisar antara 0,24% - 1,74% sesuai kelompok jenis usaha.

2.2.5.5 Jaminan Kematian (JK)

Jaminan Kematian diperuntukkan bagi ahli waris dari peserta program Jamsostek yang meninggal bukan karena kecelakaan kerja. Jaminan Kematian diperlukan sebagai upaya meringankan beban keluarga baik dalam bentuk biaya

pemakaman maupun santunan berupa uang. Pengusaha wajib menanggung iuran Program Jaminan Kematian sebesar 0,3% dengan jaminan kematian yang diberikan adalah Rp 12 Juta terdiri dari Rp 10 juta santunan kematian dan Rp 2 juta biaya pemakaman* dan santunan berkala.

2.2.5.6 Jaminan Pemeliharaan Kesehatan

Pemeliharaan kesehatan adalah hak tenaga kerja. JPK adalah salah satu program Jamsostek yang membantu tenaga kerja dan keluarganya mengatasi masalah kesehatan. Mulai dari pencegahan, pelayanan di klinik kesehatan, rumah sakit, kebutuhan alat bantu peningkatan fungsi organ tubuh, dan pengobatan, secara efektif dan efisien. Setiap tenaga kerja yang telah mengikuti program JPK akan diberikan KPK (Kartu Pemeliharaan Kesehatan) sebagai bukti diri untuk mendapatkan pelayanan kesehatan.

- Jumlah iuran yang harus dibayarkan:

  Iuran JPK dibayar oleh perusahaan dengan perhitungan sebagai berikut:

- Tiga persen (3%) dari upah tenaga kerja (maks Rp 1 juta) untuk tenaga kerja lajang
- Enam persen (6%) dari upah tenaga kerja (maks Rp 1 juta) untuk tenaga kerja berkeluarga
- Dasar perhitungan persentase iuran dari upah setinggi-tingginya Rp 1.000.000,-

2.2.6 Mutasi Dan Terminasi

2.2.6.1 Pengertian Mutasi

Mutasi merupakan perpindahan pegawai dari satu divisi/bagian ke divisi/bagian lain yang lebih tinggi.

2.2.6.2 Pengertian Terminasi / Pemutusan Hubungan Kerja

Terminasi / Pemutusan Hubungan Kerja (PHK) adalah apabila ikatan formal antara organisasi selaku pemakai tenaga kerja dan pegawainya terputus. Banyak faktor yang dapat menjadi penyebab terjadinya pemutusan hubungan kerja seperti:

- Alasan pribadi pegawai tertentu
- Pegawai dikenakan sanksi disiplin yang sifatnyaberat
- Faktor ekonomi seperti depresi atau resesi

Adanya kebijaksanaan organisasi untuk mengurangi kegiatannya yang pada gilirannya menimbulkan keharusan untuk mengurangi jumlah pegawai yang dibutuhkan oleh organisasi.

2.2.7 Insentif

2.2.7.1 Pengertian Insentif

Menurut Anwar Prabu Mangkungara (2002:89), mangatakan pengertian insentif adalah “Suatu penghargaan dalam bentuk uang yang diberikan oleh pihak pemimpin organisasi kepada karyawan agar mereka bekerja dengan motivasi yang tunggi dan berprestasi dalam pencapaian tujuan-tujuan organisasi”.

(http://makalahdanskripsi.blogspot.com/2008/09/pengertian-insentif.html)

Tujuan dari pemberian insentif, yaitu :

1. Bagi perusahaan

   Tujuan pelaksanaan pemberian insentif kepada karyawan dimaksudkan untuk meningkatkan produksi dengan cara mendorong mereka agar bekerja disiplin dan semangat yang lebih tinggi dengan tujuan menghasilkan kualitas produksi yang lebih baik serta dapat bekerja dengan menggunakan faktor produksi seefektif dan seefisien mungkin.

2. Bagi karyawan

   Dengan pemberian insentif dari perusahaan maka diharapkan karyawan memperoleh banyak keuntungan, seperti misalnya mendapatkan upah atau gaji yang lebih besar, mendapat dorongan untuk mengembangkan dirinya dan berusaha bekerja dengan sebaik – sebaiknya.

   (http://jurnal-sdm.blogspot.com/2009/05/pengupahan-insentif-definisi-tujuan-dan_05.html)

2.2.8 *Invoice*

2.2.8.1 Pengertian *Invoice*

Invoice/Faktur penjualan yang selanjutnya kita sebut *invoice* adalah dokumen yang digunakan sebagai pernyataan tagihan yang harus dibayar oleh customer. Dalam bentuk sederhana dikenal dengan nama BON. Pada transaksi yang nominalnya relatif kecil, *invoice* digunakan langsung sebagai dokumen

tagihan sedangkan pada perusahaan yang nominal transaksinya besar, biasanya dilengkapi dengan surat tagihan atau kwitansi.

2.2.9 Teori Web-*Database*

2.2.9.1 Pengertian *Web*

Menurut Eaglestone, *web* menggunakan internet dalam memberikan fasilitas kepada penggunanya untuk (2001, p190) :

a. Menyimpan dan menyampaikan informasi.
b. Mencari dan menemukan informasi
c. Memasukkan dan memanipulasi informasi

2.2.9.2 *Web Database*

Menurut Eaglestone, *web database* sistem adalah sistem dimana kedua teknologi *web* dan *database* digunakan (2001, p38). *Web database* memiliki beberapa keuntungan, diantaranya (2001, p36) :

a. *Database* dapat diakses oleh pengguna luas di seluruh dunia
b. Sistem terdistribusi

Data dapat diberikan dimana ia dibutuhkan dan aplikasi dapat diletakkan dimana ada aktivitas yang mendukung dan membutuhkan aplikasi itu.

c. *Web database* memberikan fasilitas yang menguntungkan untuk *query* data, manipulasi data dan administrasi data.

2.2.9.3 Perancangan Web *Database*

Disamping perancangan konvensional *database*, ada 2 hal yang harus dipertimbangkan dalam merancang *web database*, Eaglestone (2001, p262).

2.2.9.3.1 Perancangan halaman web

1. Menampilkan *web* data – mengambil dari *database* atau masukkan dari *user*.
2. Kumpulan *web* data – perancangan hubungan untuk petunjuk didalam maupun diantara halaman *web*.
3. Perancangan *user interface* web – perancangan fitur halaman *web*.

2.2.9.3.2 Perancangan hubungan antara halaman *web* dan basis data

1. *Web database logical mapping* – definisi *mapping* antara data yang ditampilkan di halaman *web* dan data yang disimpan dalam *database*.
2. *Web database physical mapping* – implementasi mekanisme dimana data dilakukan diantara halaman *web* dan *database*. Kinerja cepat dan bebas kesalahan.

2.2.9.4 *Web Databases Connectivity*

Menurut Eaglestone, web memiliki fasilitas untuk menyimpan, memberikan dan mengakses informasi. Informasi dapat diakses dari web menggunakan *browser* (2001, p209).

Sistem *web database* dibagi atas 2 bagian (Eaglestone, 2001, p349) :

2.2.9.4.1 *Client Systems*

Sistem ini akan menampilkan halaman web dengan *user interface* ke sistem web *database*. Halaman *web* akan ditampilkan menggunakan *browser*.

2.2.9.4.2 *Web Server Systems*

Sistem ini akan menyimpan dokumen-dokumen *web,* data-data, dan program. Pada sistem ini, data-data dan dokumen dokumen *web* dapat ditambah, diperbaharui, dan di manipulasi kedalam *database*.

2.2.9.4.2.1 Pendekatan *Client Side*

Pendekatan ini dilihar dari sisi komputer yang digunakan oleh pengguna. (Eaglestone, 2001, p350)

1. *Browser Extension*

   Browser secara luas memberikan fungsi-fungsi tambahan. Dapat digunakan menggunakan *plug-ins* (pada *Netscape* dan *Internet Explorer* (IE)), *ActiveX controls* (pada IE), atau dapat juga menggunakan *java applets* dan *javascript*.

2. *External Applications*

   Ini adalah tipe yang ada di *database client* di dalam *client system*, dan ditampilkan oleh *web browser* untuk memberikan tambahan layanan.

Aplikasi ini, serupa dengan menggunakan jembatan *server side* untuk mengeksekusi program lain.

2.2.9.4.2.2 Pendekatan *Server Side*

Menurut Eaglestone (2001, p351), pendekatan ini digunakan untuk mengeksekusi proses-proses aplikasi didalam sistem server komputer. Misalnya ketika kita ingin mengakses sistem *database,* harus dari server.

Prosedur-prosedur yang harus dilakukan dalam pendekatan ini :

1. Data (permintaan) harus dapat terkirim dari *client* ke *server*.
2. Data harus dapat di proses di *server*.
3. Hasilnya harus dapat dikirim dari *server* ke *client*.