BAB 2

LANDASAN TEORI

2.1 Basisdata

Sebelum aplikasi basisdata (DBMS) dikenal, biasanya proses penyimpanan data disimpan di dalam sebuah *file*. Menurut **Connoly (2002, p12)**, bahwa setiap program mendefinisikan dan mengatur datanya masing-masing, kemudian akhirnya ditemui banyaknya keterbatasan dari pendekatan dengan menggunakan *file*, antara lain:

1. Data yang terpisah dan terisolasi
2. Duplikasi data
3. Ketergantungan data
4. Format data yang tidak kompatibel
5. *Query* yang tetap atau tidak dapat mengantisipasi perkembangan program aplikasi

Akan tetapi, saat ini aplikasi basisdata merupakan aplikasi yang sudah umum dan sering digunakan dalam kehidupan sehari-hari.

Menurut **Connoly (2002, p7)**, *File –Based System* adalah kumpulan program aplikasi yang memberikan pelayanan kepada user untuk membuat suatu laporan, dimana setiap program mendefinisikan dan mengatur datanya masing - masing. Oleh karena itu, saat ini proses penyimpanan data telah dilakukan dengan menggunakan pendekatan basisdata atau *Database Management System* (DBMS).

2.1.1 Pengertian Basisdata dan Sistem Basisdata

Menurut **Connolly (2002, p14)**, basisdata adalah suatu koleksi data yang saling berhubungan secara logikal, dan sebuah deskripsi data yang dirancang untuk memenuhi kebutuhan informasi suatu organisasi. Dapat dikatakan juga basisdata adalah kumpulan *file* yang saling berhubungan, hubungan tersebut biasa ditunjukkan dengan kunci dari tiap *file* yang ada. Suatu basisdata menunjukkan satu kumpulan data yang dipakai dalam satu lingkup organisasi.

Menurut **Whitten (2004, p58)**, basisdata juga dapat diartikan sebagai *file* yang saling berhubungan di mana setiap baris pada suatu basisdata juga harus saling terhubung dengan baris pada lain file. Dapat disimpulkan bahwa basisdata menyimpan data yang saling berhubungan yang dibutuhkan oleh suatu organisasi untuk menyediakan informasi-informasi yang berguna.

Dengan basisdata dapat mempercepat proses pemenuhan suatu kebutuhan informasi suatu organisasi di mana kebutuhan informasi tersebut berbeda-beda pada tiap-tiap organisasi.

Menurut **Date (2000, p5)**, sistem basisdata adalah suatu *record* yang terkomputasi yang bertujuan menyajikan informasi pada saat yang dibutuhkan.

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

Dan menurut **Connolly (2002, p21)**, selain membutuhkan sebuah perangkat lunak untuk mengatur basisdata yang ada, sebuah organisasi juga membutuhkan

seseorang yang bertanggung jawab terhadap basisdata yang ada. Orang yang bertanggung jawab terhadap realisasi fisikal dari suatu basisdata, yang meliputi perancangan basisdata fisikal, implementasi, kontrol keamanan dan integritas, pemeliharaan operasional sistem, dan memastikan kepuasan pengguna terhadap unjuk kerja aplikasi dinamakan dengan *database administrator*, disingkat DBA.

2.1.2 *Database Management System* (DBMS)

Menurut **Whitten (2004, p554)**, sebuah basisdata yang besar, memerlukan sebuah perangkat lunak untuk mengatur basisdata tersebut secara keseluruhan. Perangkat lunak yang dapat digunakan untuk membuat, mengakses, mengontrol, dan mengatur suatu basisdata dinamakan sistem manajemen basisdata, biasa disebut dengan *database management system*, disingkat DBMS.

Sedangkan menurut **Connolly (2002, p16)**, *database management system* adalah suatu sistemperangkat lunak yang membantu user untuk mendefinisikan, membuat, memelihara dan mengontrol akses ke basisdata. Dalam proses penggunaannya, menurut **Connolly (2002, p25)**, database management system memiliki keuntungan, kekurangan, serta fasilitas yang mendukungnya.

Keuntungan dari DBMS, antara lain:

- Kontrol terhadap pengulangan data.
- Data yang dihasilkan konsisten.
- Dari data yang sama dapat diperoleh informasi beragam.
- Data dapat dipakai secara bersama-sama.
- Meningkatkan integritas data.
- Meningkatkan keamanan.
- Penetapan standarisasi.

- Perbandingan skala ekonomi.
- Mengatasi konflik kebutuhan.
- Memperbaiki pengaksesan data.
- Meningkatkan produktifitas.
- Memperbaiki pemeliharaan data melalui data yang tidak tergantung dengan data yang lain.
- Memperbaiki pengaksesan data secara bersama-sama.
- Meningkatkan layanan *backup* dan perbaikan data.

Kerugian penggunaan DBMS, antara lain:

- Memiliki sistem yang rumit.
- Dengan sistem yang rumit, mengakibatkan DBMS memiliki ukuran yang semakin besar.
- DBMS memiliki harga yang bervariasi tergantung dari fungsi dan kebutuhan.
- Penambahan biaya untuk perangkat keras yang dibutuhkan.
- Penambahan biaya konversi.
- Karena DBMS dirancang untuk mengakses lebih dari satu aplikasi sehingga performansinya menurun.
- Kegagalan DBMS mengakibatkan operasi tidak dapat berjalan.

Fasilitas yang disediakan DBMS, antara lain:

a. DDL *(Data Definition Language).*

Menurut Connolly (2002, p40), DDL merupakan sebuah fasilitas yang digunakan untuk mendefinisikan skema basisdata atau memodifikasi basisdata yang telah ada. DDL adalah suatu bahasa yang memperbolehkan DBA *(Database Administrator)* atau user untuk mendeskripsikan dan memberi nama *entity-entity*, atribut-atribut, dan

relationship-relationship yang dibutuhkan untuk aplikasi, bersama dengan batasan integritas dan keamanan yang berhubungan.

b. DML *(Data Manipulation Language)*.

Menurut **Connolly(2002, p40-41),** DML adalah fasilitas yang memperbolehkan user untuk membaca dan meng-*update* atau memanipulasi data, dimana manipulasi data meliputi: insert data baru ke dalam basisdata, memodifikasi data yang disimpan di dalam basisdata (*update, edit*), mencari suatu data yang berada pada basisdata, menghapus data dari basisdata. Dengan kata lain, *DML* adalah suatu bahasa yang menyediakan serangkaian operasi untuk mendukung operasi dasar manipulasi data yang tersimpan di dalam basisdata.

c. SQL *(Structured Query Language)*.

SQL adalah sebuah fasilitas yang digunakan untuk melayani pengaksesan data. Bahasa *query* yang paling baik adalah yang secara *de facto* merupakan standar bagi DBMS.

2.1.3 Entity-Relationship Modeling (E-R Modeling)

Menurut **Connolly (2002, p330)**, model *Entity-Relationship* merupakan salah satu model yang dapat memastikan pemahaman yang tepat terhadap data dan bagaimana penggunaannya di dalam suatu organisasi. Model ini dimulai dengan identifikasi entitas dan *relationship* antardata yang harus direpresentasikan di dalam model, dan kemudian ditambahkan atribut dan setiap *constraint* pada entitas, *relationship*, dan atributnya.

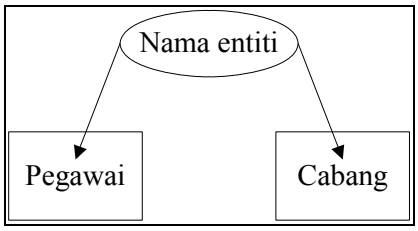
2.1.3.1 Konsep Dasar Model E-R

Beberapa konsep dasar dalam model E-R, yaitu:

A. Tipe Entitas

Menurut **Connolly (2002, p331)**, tipe entitas adalah sekumpulan objek yang memiliki properti yang sama, yang diidentifikasikan di dalam organisasi karena keberadaannya yang bebas (*independent existence*). Sedangkan *entity occurrence* menurut **Connolly (2002, p333)**, adalah sebuah objek dari satu tipe entitas yang dapat diidentifikasi secara unik. Keberadaan objek-objeknya secara fisik/nyata (*physical existence*), seperti entitas Pegawai, Rumah, dan Pelanggan, atau secara konseptual/abstrak (*conceptual existence*), seperti entitas Inspeksi, Penjualan, dan Peninjauan.

Setiap tipe entitas dilambangkan dengan sebuah persegi panjang yang diberi nama dari entitas tersebut. Nama tipe entitas biasanya adalah kata benda tunggal. Huruf pertama dari setiap kata pada nama tipe entitas ditulis dengan huruf besar. Representasi diagram tipe entitas terlihat pada gambar 2.1.



Gambar 2-1 Representasi Diagram dari Tipe Entitas Pegawai dan Cabang, Connolly (2002, p333)

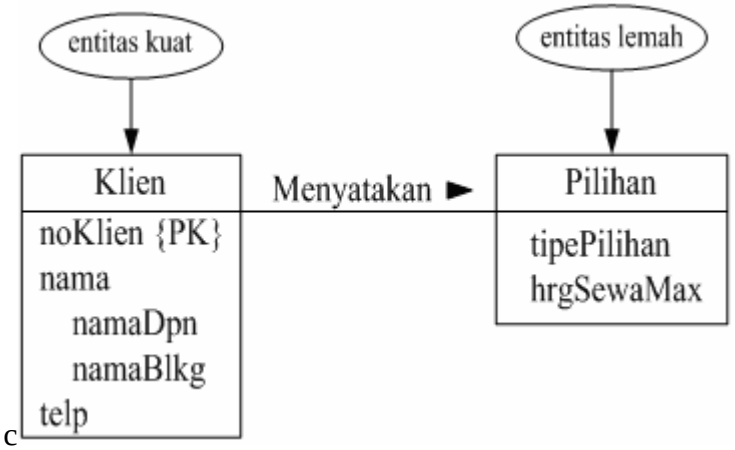
Tipe entitas dapat diklasifikasikan menjadi:

- Tipe Entitas Kuat

  Menurut **Connolly (2002, p342)**, yaitu tipe entitas yang keberadaannya tidak bergantung pada tipe entitas lainnya.

- Tipe Entitas Lemah

  Menurut **Connolly (2002, p343)**, yaitu tipe entitas yang keberadaannya bergantung pada tipe entitas lainnya.



c

Gambar 2-2 Representasi Diagram Tipe Entitas Kuat dan Tipe Entitas Lemah, Connolly (2002, p343)

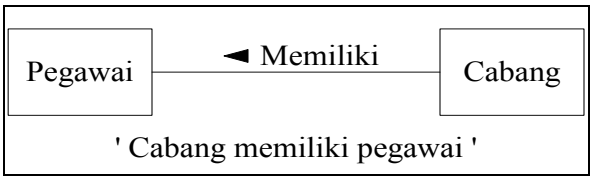
B. Tipe *Relationship*

Menurut **Connolly (2002, p334)**, tipe *relationship* adalah sekumpulan hubungan antartipe entitas yang memiliki arti. Sedangkan *relationship occurrence,* menurut **Connolly (2002, p334)**, adalah sebuah hubungan yang dapat diidentifikasikan secara unik, yang meliputi sebuah kejadian (*occurrence*) dari setiap tipe entitas di dalam *relationship*.

Tipe *relationship* digambarkan dengan sebuah garis yang menghubungkan tipe entitas-tipe entitas yang saling berhubungan. Garis tersebut diberi nama sesuai dengan nama hubungannya dan diberi tanda panah satu arah di samping nama hubungannya.

Biasanya sebuah *relationship* dinamakan dengan menggunakan kata kerja, seperti *Mengatur*, atau dengan sebuah frase singkat yang meliputi sebuah kata kerja, seperti *DisewaOleh*. Sedangkan tanda panah ditempatkan di samping nama *relationship* yang

mengindikasikan arah bagi pembaca untuk mengartikan nama dari suatu *relationship*. Huruf pertama dari setiap kata pada nama *relationship* ditulis dengan huruf besar. Representasi diagram dari suatu tipe *relationship* terlihat pada gambar 2.3.



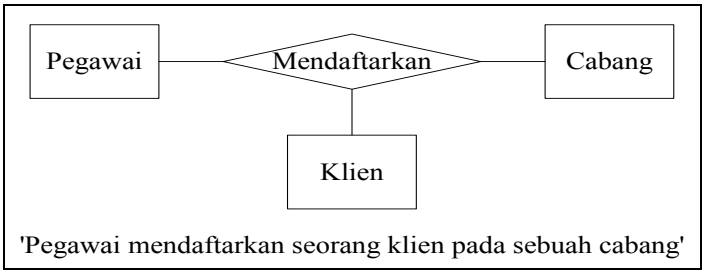
Gambar 2-3 Representasi Diagram dari Tipe *relationship*, Connolly (2002, p335)

- **Derajat dari Tipe *Relationship***

  Menurut **Connolly (2002, p335)**, derajat dari tipe *relationship* adalah jumlah tipe entitas yang ikut serta dalam sebuah *relationship*.

  Menurut **Connolly (2002, p445)**, c*omplex relationship types* adalah sebuah *relationship* antara tiga atau lebih tipe entitas. Masih menurut **Connolly (2002, p336)**, sebuah *relationship* yang memiliki derajat dua dinamakan *binary*. Gambar 2.3 juga merepresentasikan diagram *relationship* derajat dua. Sedangkan sebuah *relationship* derajat tiga dinamakan *ternary*, dan jika sebuah *relationship* memiliki derajat empat dinamakan *quarternary*.

  Lambang belah ketupat merepresentasikan *relationship* yang memiliki derajat lebih dari dua. Nama dari *relationship* tersebut ditampilkan di dalam lambang belah ketupat. Panah yang biasanya terdapat di samping nama suatu *relationship* dihilangkan. Representasi diagram derajat tiga dari suatu tipe *relationship* terlihat pada gambar 2.4.



Gambar 2-4 Representasi Diagram Derajat Tiga dari Suatu Tipe *Relationship*

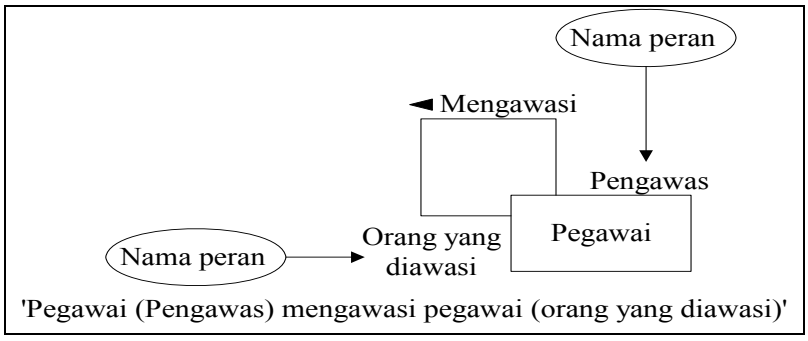
Connolly (2002, p336)

- ***Recursive Relationship***

Menurut **(Connolly, 2002, p337)**, r*ecursive relationship* adalah sebuah tipe *relationship* dimana tipe entitas yang sama ikut serta lebih dari sekali pada peran yang berbeda.

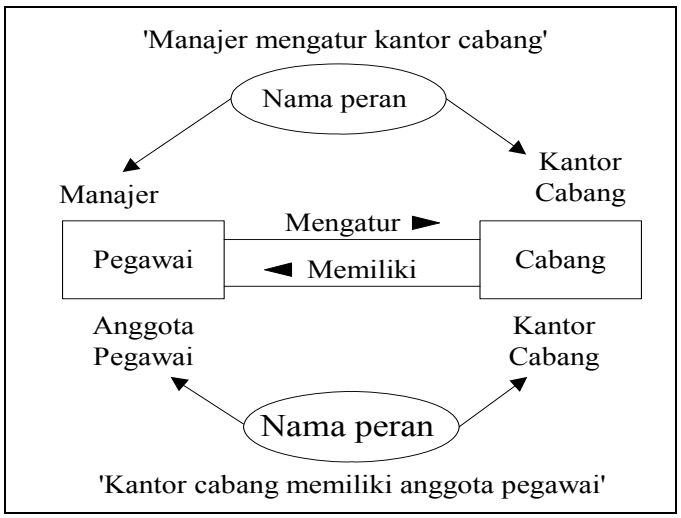
Relationship dapat diberikan nama peran untuk menentukan fungsi dari setiap entitas yang terlibat dalam *relationship* tersebut. Representasi diagram *recursive relationship* beserta nama perannya terlihat pada gambar 2.5.

Nama peran juga dapat digunakan jika dua buah entitas dihubungkan melalui lebih dari satu *relationship*. Representasi diagram nama peran yang digunakan pada dua buah entitas terlihat pada gambar 2.6.



Gambar 2-5 Representasi Diagram *Recursive Relationship* dan Nama Peran,

Connolly (2002, p337)



Gambar 2-6 Representasi Diagram Entitas dengan Dua *Relationship* Berbeda Beserta Nama Peran, Connolly (2002, p338)

C. Atribut

Menurut **Connolly (2002, p338)**, atribut adalah properti sebuah entitas atau *relationship*. Menurut **Jeffery L. Whitten (2004, p295)**, Atribut merupakan properti deskriptif atau karakteristik dari sebuah entitas. Atribut menampung nilai yang menjelaskan setiap *entity occurrence* dan menggambarkan bagian utama dari data yang disimpan di dalam basis data.

Menurut **Connolly (2002, p338)**, atribut *domain* adalah sekumpulan nilai yang dibolehkan bagi satu atau lebih atribut. Atribut dapat diklasifikasikan menjadi:

1) *Simple attribute,*

   Menurut **Connolly (2002, p339)**, *simple attribute* adalah atribut yang terdiri dari komponen tunggal denga keberadaaannya yang bebas.

2) *Composit attribute,*

   Menurut **Connolly (2002, p339)**, *composit attribute* adalah atribut yang terdiri dari beberapa komponen, dan keberadaan setiap komponen tersebut bebas.

3) *Single-valued attribute,*

   Menurut Connolly (2002, p339), *single-valued attribute* adalah atribut yang hanya memiliki sebuah nilai untuk setiap *occurrence* dari sebuah entitas.

4) *Multi-valued attribute*

   Menurut **Connolly (2002, p340)**, *multi-valued attribute* adalah sebuah atribut yang memiliki banyak nilai untuk setiap *occurrence* dari sebuah tipe entitas.

5) *Derived attribute*

   Menurut **Connolly (2002, p340)**, *derived attribute* adalah atribut yang merepresentasikan sebuah nilai yang diturunkan dari atribut lain yang berhubungan atau kumpulan dari atribut.

2.1.4 Keys

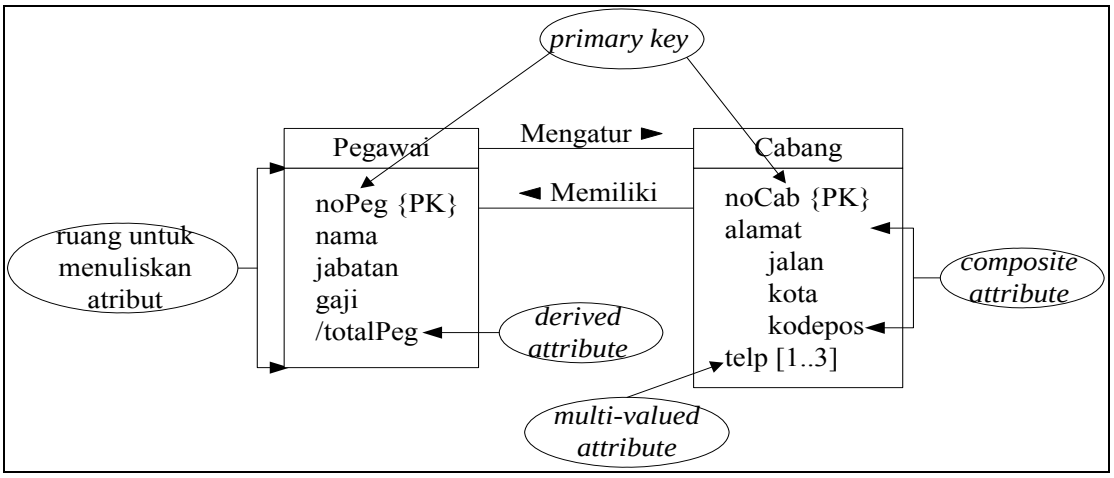
Menurut **Connolly (2002, p340)**, *candidate key* adalah himpunan atribut yang minimal yang secara unik mengidentifikasikan setiap *occurrence* dari sebuah tipe entitas.

Menurut **Connolly (2002, p341)**, *composite key* adalah sebuah *candidate key* yang terdiri atas dua atau lebih atribut.

Menurut **Connolly (2002, p341)**, p*rimary key* adalah *candidate key* yang terpilih untuk mengidentifikasikan secara unik setiap *occurrence* dari sebuah tipe entitas. Pada sebuah tipe entitas biasanya terdapat lebih dari satu *candidate key* yang salah satunya harus dipilih untuk menjadi *primary key*. Pemilihan *primary key* didasarkan pada panjang atribut, jumlah minimal atribut yang diperlukan, dan keunikannya.

Sedangkan menurut **Whitten (2004, p298)**, a*lternate key* adalah setiap *candidate key* yang tidak terpilih menjadi *primary key*, atau biasa disebut dengan *secondary key*.

Menurut **Whitten (2004, p301)**, f*oreign key* adalah sebuah *primary key* pada sebuah entitas yang digunakan pada entitas lainnya untuk mengidentifikasikan sebuah *relationship*.



Gambar 2-7 Representasi Diagram Entitas Pegawai dan Cabang Beserta Atribut dan *Primary Key*nya, Connolly (2002, p342)

2.1.5 Batasan Struktural (Structural Constraints)

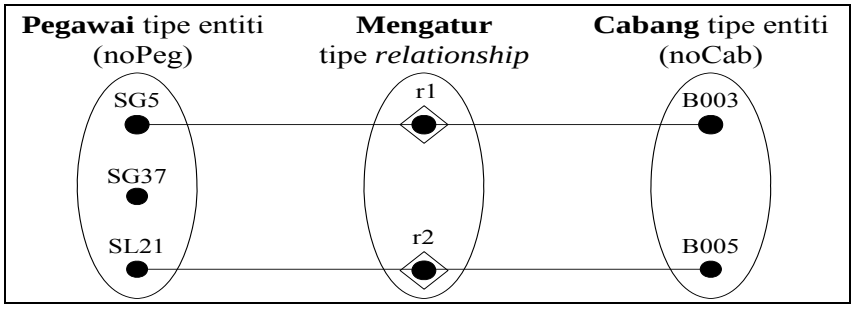
Menurut **Connolly (2002, p344)**, batasan-batasan yang menggambarkan pembatasan pada *relationship* seperti yang ada pada '*real world*' harus diterapkan pada tipe entitas yang ikut serta pada sebuah *relationship*. Jenis utama dari batasan pada suatu *relationship* dinamakan *multiplicity*.

Sedangkan *Multiplicity* menurut **Connolly (2002, p344)**, adalah jumlah *occurrence* yang mungkin terjadi pada sebuah tipe entitas yang berhubungan ke sebuah *occurrence* dari tipe entitas lain pada suatu *relationship*.

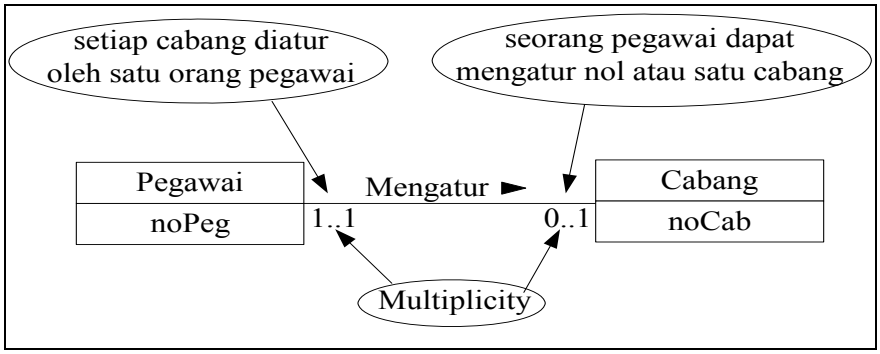
Derajat yang biasanya digunakan pada suatu *relationship* adalah *binary relationship*, yang terdiri atas:

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

Setiap *relationship* menggambarkan hubungan antara sebuah *entity occurrence* pada entitas yang satu dengan sebuah *entity occurrence* pada entitas lainnya yang ikut serta dalam *relationship* tersebut.



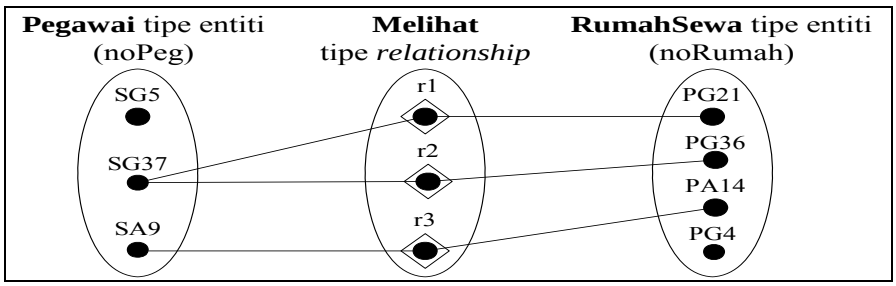
Gambar 2-8 *Semantic Net* Menunjukkan Dua *Occurrence* dari *Relationship* Pegawai Mengatur Cabang, Connolly (2002, p345)



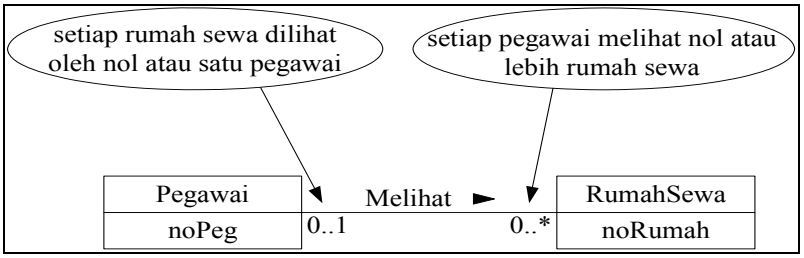
Gambar 2-9 *Multiplicity* dari *Relationship one-to-one* (1:1), Connolly (2002, p346)

- ***One-to-many* (1:*) *Relationship***

Setiap *relationship* menggambarkan hubungan antara sebuah *entity occurrence* pada entitas yang satu dengan satu atau lebih *entity occurrence* pada entitas lainnya yang ikut serta dalam *relationship* tersebut.



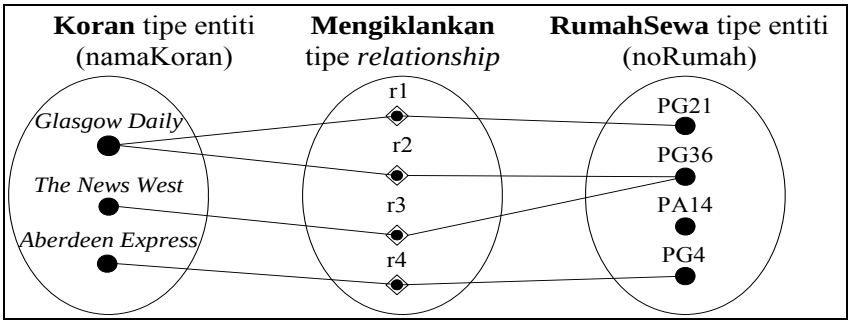
Gambar 2-10 *Semantic Net* Menunjukkan Tiga *Occurrence* Dari *Relationship* Pegawai Melihat RumahSewa, Connolly (2002, p346)



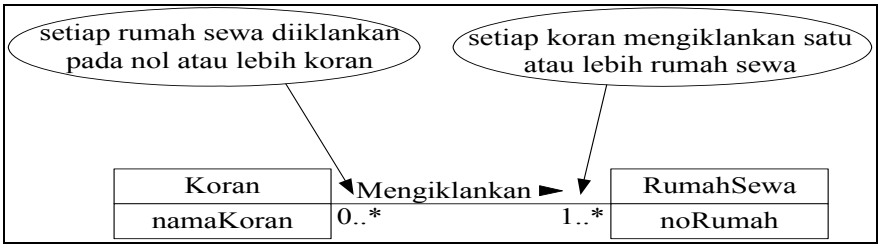
Gambar 2-11 *Multiplicity* dari *Relationship one-to-many* (1:*), Connolly (2002, p347)

- ***Many-to-many* (*:*) *Relationship***

Setiap *relationship* menggambarkan hubungan antara satu atau lebih *entity occurrence* pada entitas yang satu dengan satu atau lebih *entity occurrence* pada entitas lainnya yang ikut serta dalam *relationship* tersebut.



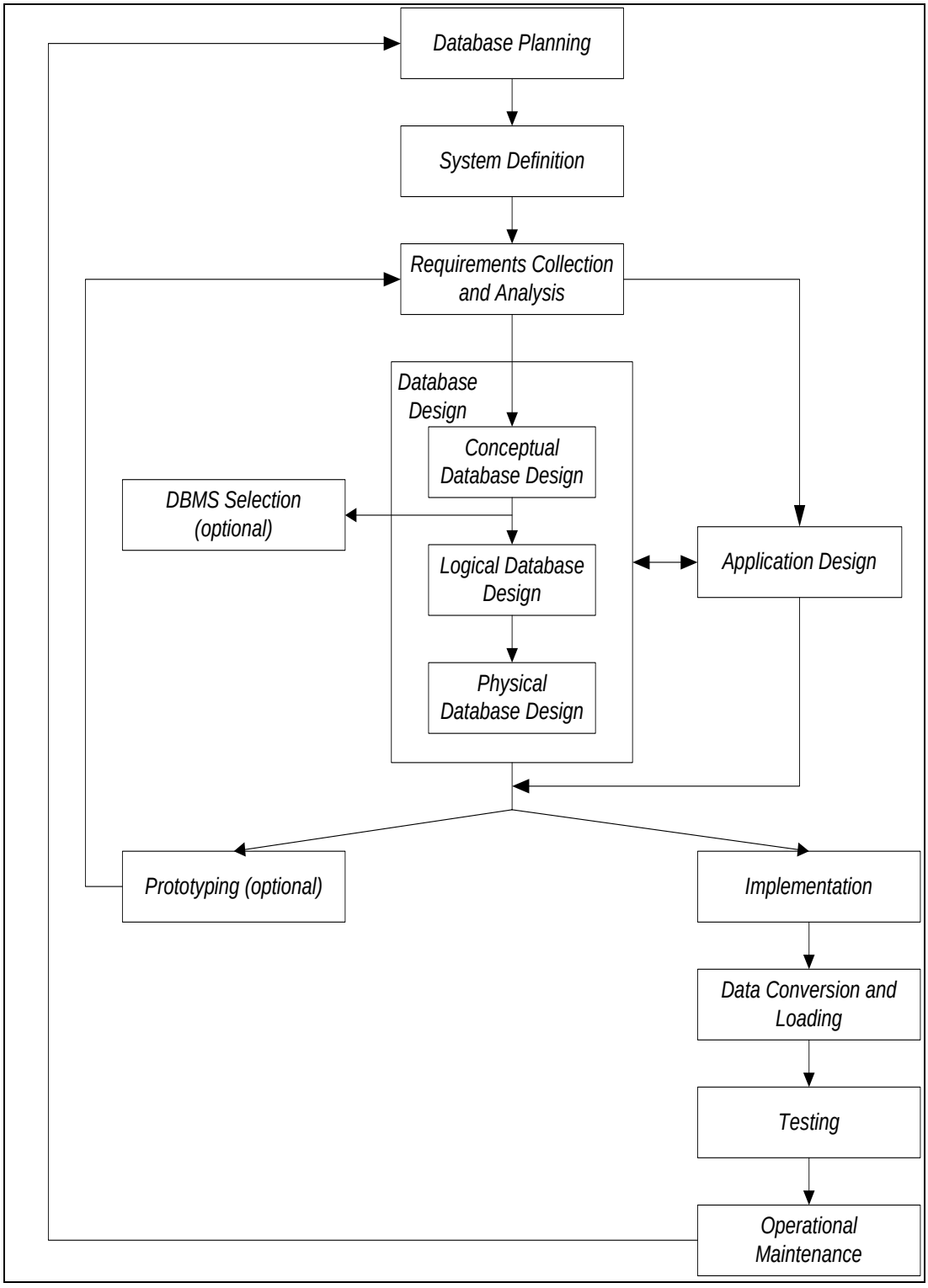
Gambar 2-12 *Semantic Net* Menunjukkan Empat *Occurrence* dari *Relationship* Koran Mengiklankan RumahSewa, Connolly (2002, p348)



Gambar 2-13 *Multiplicity* dari *relationship many-to-many* (*:*), Connolly (2002, p348)

2.1.6 Database Application Lifecycle (DBLC)

Untuk merancang aplikasi sistem basisdata diperlukan tahapan-tahapan terstruktur yang harus diikuti yang dinamakan dengan Siklus Hidup Aplikasi Basisdata (*Database Application Lifecycle*) atau disingkat dengan *DBLC*. Perlu diingat bahwa tahapan dalam *DBLC* tidak harus berurutan, namun juga melibatkan beberapa pengulangan ke tahapan sebelumnya melalui putaran balik (*feedback loops*). Tahapan-tahapan tersebut terlihat pada gambar 2.14.



Gambar 2-14 *Database Application Lifecycle,* Connolly (2002, p272)

- *Database Planning*

Perencanaan basisdata adalah kegiatan pengaturan yang memungkinkan tahap-tahap dalam aplikasi basisdata dapat diwujudkan secara efisien dan secara efektif mungkin, hal ini disebutkan oleh **Connolly (2002, p273)**. Tahap perencanaan basisdata juga harus menjelaskan :

- ***Mission statement*** dari proyek basisdata. Menurut **Connolly (2002, p274)**, *mission statement* ini menjelaskan tujuan utama aplikasi basisdata, juga membantu menjelaskan tujuan proyek basis data, dan menyediakan maksud yang lebih jelas dalam pembuatan aplikasi basisdata secara efektif dan efisien. Dengan merumuskan apa sebenarnya yang menjadi tujuan dari proyek basis data ini diharapkan dapat lebih memfokuskan pekerjaan pada tahap selanjutnya.
- ***Mission objectives*.** Selain merumuskan tujuan dari sebuah proyek basisdata, harus diperhatikan juga mengenai tugas apa saja yang harus didukung oleh basisdata tersebut. Menurut **Connolly (2002, p274)**, setiap *mission objective* akan menjelaskan tugas tertentu yang harus didukung oleh basisdata, dengan asumsi jika basisdata mendukung *mission objectives*, maka *mission statement*nya juga akan sesuai.

Perencanaan basisdata harus terintegrasi dengan keseluruhan strategi sistem informasi dari organisasi. Ada 3 masalah pokok dilibatkan dalam perumusan suatu strategi sistem informasi, yaitu:

- Identifikasi rencana dan tujuan perusahaan dengan penentuan kebutuhan sistem informasi.

- Evaluasi tentang sistem informasi untuk mengetahui kelebihan dan kekurangannya.
- Penilaian kesempatan *information technology* yang mungkin menghasilkan manfaat yang kompetitif.

- *System Definition*

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

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

- *Requirement Collection and Analysis*

  Tahap selanjutnya yang dilakukan setelah Pendefinisian Sistem adalah tahap pengumpulan kebutuhan dan analisis. Menurut **Connolly (2002, p276)**, dalam tahap ini dilakukan proses pengumpulan dan analisa informasi tentang bagian organisasi yang akan didukung oleh aplikasi basis data, dan menggunakan informasi ini untuk mengidentifikasi kebutuhan pengguna terhadap sistem yang baru.

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

1. **Memeriksa dokumentasi**

   Pemahaman terhadap jalannya sistem akan cepat diperoleh dengan memeriksa dokumen-dokumen, formulir, laporan, dan berkas yang terkait dengan sistem yang sedang berjalan pada perusahaan. Dengan pemeriksaan ini diharapkan dapat mengetahui data-data apa saja yang akan disimpan di dalam basis data.

2. **Mengamati operasional perusahaan**

   Pengamatan ini memungkinkan untuk ikut serta atau mengamati seseorang melakukan kegiatan untuk mempelajari sistem. Salah satu faktor pengamatan dapat berhasil adalah dengan mencari informasi sebanyak mungkin tentang aktivitas yang akan diamati serta orang yang melakukan aktivitas tersebut.

   Keuntungan menggunakan teknik ini antara lain :

   - Validitas fakta dan data dapat diperiksa
   - Pengamat dapat melihat dengan jelas apa yang dikerjakan
   - Pengamat juga dapat memperoleh data yang menjelaskan lingkungan fisik dari tugas yang diberikan
   - Relatif murah
   - Pengamat dapat membuat pengukuran kerja

Kerugian teknik ini yaitu:

- Sangat memakan waktu dan biaya, sehingga menjadi tidak praktis
- Dapat terlewat dalam mengamati tugas-tugas yang melibatkan tingkat kesulitan yang lain
- Beberapa tugas tidak selalu dilakukan dengan cara seperti pada saat pengamatan

3. Penelitian

Selain melakukan penelitian yang berasal dari dalam organisasi itu sendiri, dapat juga dilakukan pengumpulan informasi yang berasal dari luar organisasi tersebut. Beberapa contoh sumber informasi tersebut antara lain jurnal komputer, buku-buku referensi, dan internet. Sumber informasi tersebut juga dapat digunakan untuk memecahkan masalah serupa.

Keuntungan menggunakan teknik ini antara lain :

- Dapat menghemat waktu jika solusinya telah tersedia
- Peneliti dapat mengamati cara orang lain memecahkan masalah yang sama atau menemui kebutuhan yang serupa
- Membuat para peneliti selalu *up-to-date* dengan perkembangan baru

Kerugian teknik ini yaitu :

- Dapat menjadi sangat memakan waktu
- Membutuhkan akses ke sumber informasi yang tepat
- Dapat saja tidak membantu memecahkan masalah karena tidak didokumentasikan

- *Database Design*

Menurut **Connolly (2002, p279)**, perancangan basis data (*database design*) adalah proses pembuatan sebuah rancangan untuk basis data yang akan mendukung operasi dan tujuan perusahaan. Dalam perancangannya diperlukan metodologi yang benar sehingga perancangan dapat berjalan sesuai dengan konsep yang ada. Menurut **Connolly (2002, p418)**, metodologi perancangan (*design methodology*) adalah pendekatan terstruktur yang menggunakan prosedur-prosedur, teknik-teknik, peralatan, dan dokumentasi untuk mendukung dan memudahkan proses perancangan. Teknik ini digunakan untuk membantu merencanakan, mengatur, mengontrol, dan mengevaluasi proyek pengembangan basis data. Tahapan dalam metodologi perancangan ada tiga, yaitu:

1. **Perancangan basisdata konseptual *(Conceptual Database Design)***

Menurut **Connolly (2002, p419)**, *conceptual database design* adalah proses membangun model informasi yang digunakan organisasi, bebas dari semua pertimbangan fisik. Pertimbangan fisik yang dimaksud meliputi DBMS yang akan digunakan, program aplikasi, bahasa pemrograman, *platform* perangkat keras, unjuk kerja, dan pertimbangan fisik lainnya. Langkah-langkah dalam metodologi *conceptual database design* yaitu :

Langkah 1 - Membangun *Local Conceptual Data Model* untuk setiap pandangan pengguna

Menurut **Connolly (2002, p421)**, bertujuan untuk memecah rancangan menjadi tugas-tugas yang dapat diatur dengan memeriksa sudut pandang yang berbeda dari pengguna di dalam organisasi. Hasil dari langkah ini berupa pembuatan satu atau lebih *local conceptual data model* yang

merupakan penggambaran yang tepat dan lengkap dari suatu organisasi dilihat dari para pengguna yang berbeda. Tugas-tugas yang dilakukan pada langkah ini terdiri dari :

Langkah 1.1 - Mengidentifikasi tipe entitas

Tipe entitas dapat dikenali dengan mengidentifikasikan kata benda atau frase kata benda pada spesifikasi kebutuhan pengguna, objek besar seperti orang (*people*), tempat (*place*), benda (*thing*) atau konsep (*concept*). Alternatif lain adalah dengan mencari obyek yang keberadaannya bebas.

Langkah 1.2 - Mengidentifikasi tipe hubungan antar entitas

Bertujuan untuk mengidentifikasi *relationship* yang penting yang ada antara tipe entitas-tipe entitas yang telah diidentifikasi sebelumnya. Tipe *relationship* diidentifikasikan dengan mencari kata kerja atau suatu kata yang berhubungan dengan kata kerja.

Langkah 1.3 - Mengidentifikasi dan menghubungkan atribut dengan entitas atau hubungan

Tujuannya untuk menghubungkan atribut dengan entitas dan tipe *relationship* yang tepat. Atribut yang dimiliki oleh setiap entitas dan *relationship* harus memenuhi karakteristik atribut yaitu s*imple*/*composite attribute, single*/*multi-valued attribute,* dan *derived attribute.*

Langkah 1.4 - Menentukan *attribute domains*

Menurut **Connolly (2002, p430)**, *domain* adalah sekumpulan nilai dimana satu atau lebih atribut memperoleh nilainya. Contoh menentukan

domain pada atribut JenisKelamin di entitas Mahasiswa adalah dengan ‘L’ atau ’P’.

Langkah 1.5 - Menentukan *Candidate Key* dan *Primary Key*

Tujuannya untuk mengidentifikasi *candidate key* setiap tipe entitas, dan jika terdapat lebih dari satu *candidate key* maka terpilih satu sebagai *primary key*.

Langkah 1.6 - Pertimbangkan penggunaan *enhance modelling concepts* (pilihan)

Maksud dari langkah ini adalah untuk menentukan *specialization*, *generalization*, *aggregation*, *composition.*

Menurut **Connolly (2002, p362)**, s*pecialization* merupakan suatu proses memaksimalkan perbedaan-perbedaan antara anggota-anggota sebuah entitas dengan cara mengidentifikasi karakteristik yang membedakan entitas tersebut.

Menurut **Connolly (2002, p363)**, *generalization* merupakan suatu proses meminimalkan perbedaan-perbedaan antara entitas-entitas dengan cara mengidentifikasi sifat umum entitas.

Menurut **Connolly (2002, p371)**, *aggregation* menggambarkan *relationship* ‘*has-a*’ atau ‘*is-part-of*’ antara tipe entitas dimana yang satunya mewakili ‘*whole*’ (seluruhnya) dan yang satunya lagi mewakili ‘*part*’ (bagian).

Langkah 1.7 - Memeriksa model akan redudansi

Bertujuan memeriksa *conceptual model* untuk menghindari adanya redundansi atau pengulangan data dalam model. Ada dua kegiatan yang dapat dilakukan pada tahap ini:

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

Kemungkinan ada dua entitas yang menggambarkan objek yang sama dalam organisasi. Oleh karena itu, kedua entitas tersebut harus digabungkan.

b. Menghilangkan relasi yang redundan

Suatu *relationship* menjadi redundan jika informasi yang sama dihasilkan melalui *relationship* yang lainnya. Untuk meminimalkan *data model* maka *relationship* yang redundan harus dihilangkan.

Langkah 1.8 - Validasi model konseptual lokal terhadap transaksi pengguna

Menurut **Connolly (2002, p435)**, bertujuan untuk memastikan *local conceptual data model* mendukung transaksi yang dibutuhkan oleh pandangan pengguna. Dua pendekatan untuk memastikan *local conceptual data model* mendukung kebutuhan transaksi:

a. Menggambarkan transaksi (*describing the transaction*)

Memeriksa semua informasi (entitas, *relationship*, dan atributnya) yang dibutuhkan setiap transaksi yang disediakan oleh model seperti yang disebutkan **Connolly (2002, p435)**.

b. Menggunakan *transaction pathways*

Memvalidasi *data model* terhadap kebutuhan transaksi dengan menggambar diagram yang mewakili *pathway* yang diambil oleh setiap transaksi secara langsung pada *E-R diagram* seperti yang disebutkan **Connolly (2002, p435)**.

Langkah 1.9 - Melihat kembali *conceptual data model* dengan pengguna

Langkah ini dilakukan dengan tujuan untuk memastikan bahwa *data model* merupakan representasi yang benar bagi setiap pandangan.

2. Perencanaan basisdata logikal *(Logical Database Design)*

Menurut **Connolly (2002, p441)**, desain basis data logikal adalah proses membangun model informasi yang digunakan organisasi berdasarkan model data tertentu, tetapi tidak tergantung dari *Database Management System (DBMS)* dan pertimbangan fisik lainnya. Langkah-langkah dalam metodologi *logical database design* yaitu:

Langkah 2 - Membangun dan validasi *local logical data model* bagi setiap pandangan pengguna.

Tujuannya untuk membangun sebuah *local logical data model* dari sebuah *local conceptual data model* yang mewakili pandangan tertentu dari organisasi dan kemudian memvalidasi model ini untuk memastikan bahwa strukturnya benar (dengan menggunakan teknik normalisasi) dan untuk memastikan dukungannya terhadap transaksi-transaksi yang dibutuhkan. Kegiatan yang dilakukan pada langkah ini meliputi:

Langkah 2.1 - Menghilangkan fitur-fitur yang tidak *compatible* dengan model *relational* (langkah pilihan)

Bertujuan untuk menyaring *local conceptual data model s*ehingga fitur-fitur yang tidak sesuai dengan model relasional dihilangkan. Langkah-langkahnya antara lain:

a. Menghilangkan *many-to-many (*:*) binary relationship*

Dengan memecah *relationship* yang mengandung *many-to-many (*:*)* untuk mengidentifikasikan sebuah entitas tengah (*intermediate entity*) sehingga *relationship* ini digantikan dengan dua buah *one-to-many (1:*) relationship*, dengan entitas tengah berada di antara dua buah entitas yang lama.

b.* Menghilangkan *many-to-many (*:*) recursive relationship types

Jika *recursive relationship* ada pada *conceptual data model*, *relationship* tersebut harus dipecah untuk mengidentifikasikan sebuah entitas tengah dengan cara menganggap entitas yang terlibat pada *relationship* ini merupakan dua buah entitas dengan jenis *relationship many-to-many (*:*) binary* sehingga penyelesaiannya sama dengan penyelesaian pada *relationship many-to-many (*:*) binary*.

c. Menghilangkan *complex relationship types*

Dihilangkan dengan memecah *relationship* ini untuk mengidentifikasikan entitas tengah (*intermediate entity*). Kemudian *complex relationship* ini akan digantikan dengan beberapa *one-to-many (1:*) binary relationship*.

d. Menghilangkan *multi-valued attributes*

Cara menghilangkannya adalah dengan memecah atribut ini untuk mengidentifikasikan sebuah entitas.

Langkah 2.2 - Membuat relasi untuk model data *logical*

Yang ditentukan pertama kali adalah nama relasi diikuti daftar *simple attribute* yang disertai dengan tanda kurung, *primary key* beserta *alternate key* dan atau *foreign key* dari relasi. *Relationship* antara satu entitas dengan entitas lainnya digambarkan dengan mekanisme *primary key* atau *foreign key*.

Langkah 2.3 - Validasi relasi dengan normalisasi

Normalisasi menurut **Connolly (2002, p376)**, adalah suatu teknik untuk menghasilkan himpunan relasi dengan properti yang diinginkan berdasarkan kebutuhan-kebutuhan data suatu organisasi.

Tujuan dari normalisasi adalah sebagai berikut:

- Menghilangkan kumpulan relasi dari update anomalli
- Menghilangkan duplikasi data
- Membuat model relasional lebih informatif

Proses normalisasi dimulai dengan memindahkan data sumber ke bentuk tabel dengan format baris dan kolom. Yang menurut **Connolly (2002, p388)**, tabel ini berbentuk tidak normal dan disebut dengan *unnormalized table*.

Unnormalized form (*UNF*), dipaparkan oleh **Connolly (2002, p387)**, adalah suatu tabel yang terdiri dari satu atau lebih kelompok yang berulang (*repeating group*). Sedangkan r*epeating group* menurut **Connolly (2002, p388)**, adalah sebuah atribut atau himpunan atribut di dalam tabel yang memiliki lebih dari satu nilai (*multiple value*) untuk sebuah *primary key* pada tabel tersebut.

Proses atau tahapan yang terjadi pada normalisasi adalah:

a. *First Normal Form* (*1NF*)

Menurut **Connolly (2002, p388)**, suatu relasi dikatakan *1NF* jika titik temu tiap baris dan kolom pada relasi tersebut mengandung satu dan hanya satu nilai.

Sebuah relasi akan berada dalam bentuk *1NF* jika *repeating group*nya sudah hilang. Ada dua pendekatan untuk menghilangkan *repeating group* pada tabel yang tidak normal, yaitu:

- Dengan memasukkan data yang sesuai ke dalam kolom yang kosong dari baris yang mengandung data berulang.
- Dengan menempatkan data yang berulang bersama salinan atribut kunci pada relasi yang terpisah. Sebuah *primary key* diidentifikasikan ke dalam relasi yang baru.

b. *Second Normal Form* (*2NF*)

Menurut **Connolly (2002, p392),** relasi dikatakan *2NF* jika relasi tersebut berada pada *1NF* dan setiap atribut yang bukan *primary key* bergantung penuh (*fully functionally dependent*) terhadap *primary key*.

Full functional dependency, dikatakan **Connolly (2002, p391)**, terjadi jika A dan B merupakan atribut dari suatu relasi, dan B dikatakan bergantung penuh terhadap A (A→B), jika B bergantung terhadap A, namun bukan subset dari A.

Untuk menghasilkan relasi dalam bentuk *2NF* melibatkan penghilangan ketergantungan sebagian (*partial dependency*) dan

menempatkannya pada relasi yang baru bersama salinan atribut penentunya (*determinant attribute*).

c. *Third Normal Form* (*3NF*)

Menurut **Connolly (2002, p394)**, suatu relasi dikatakan *3NF* jika relasi tersebut berada dalam bentuk *1NF* dan *2NF*, dan tidak ada atribut bukan *primary key* bergantung secara transitif (*transitively dependent*) terhadap *primary key*.

Transitive dependency menurut **Connolly (2002, p394),** ialah sebuah kondisi dimana A, B, dan C merupakan atribut dari relasi yang jika A→B dan B→C maka C disebut bergantung secara transitif (*transitively dependent*) terhadap A melalui B (A tidak *functionally dependent* terhadap B atau C).

d. *Boyce-Codd Normal Form (BCNF)*

Menurut **Connolly (2002, p398)**, relasi dikatakan *BCNF* jika dan hanya jika setiap determinan adalah *candidate key*.

Candidate key menurut **Connolly (2002, p398)**, adalah atribut, atau kumpulan dari atribut, dimana beberapa atribut lain sepenuhnya bergantung secara fungsional.

Perbedaan antara *3NF* dengan *BCNF* adalah bahwa untuk *functional dependency*, *3NF* mengijinkan adanya ketergantungan dalam relasi A → B, jika B merupakan atribut *primary key* dan A bukan merupakan *candidate key*.

e. *Fourth Normal Form(4NF)*

Menurut **Connolly (2002, p408)**, sebuah relasi dikatakan 4NF saat relasi tersebut dalam BCNF dan tidak lagi memiliki *non-trivial Multi Valued Dependencies* (MVD). MVD sendiri adalah dependensi antara atribut-atribut dalam relasi meskipun relasi tersebut telah BCNF sehingga menyebabkan redundansi data.

Langkah 2.4 - Validasi relasi terhadap transaksi pengguna

Bertujuan untuk memastikan bahwa relasi-relasi pada *local logical data model* mendukung transaksi-transaksi yang dibutuhkan oleh pengguna, seperti terinci pada spesifikasi kebutuhan pengguna.

Langkah 2.5 - Mendefinisikan *integrity constraint*

Menurut **Connolly (2002, p457)**, *integrity constraint* adalah batasan-batasan yang harus ditentukan untuk melindungi basis data agar tetap konsisten. Ada lima jenis *integrity constraint*, yaitu:

a. Required data

Beberapa atribut harus selalu berisi nilai yang benar (*valid*), tidak dapat bernilai null. *Constraint* ini harus diidentifikasikan pada saat mendokumentasikan atribut-atribut pada kamus data (langkah 1.3).

b. Attribute domain constraint

Menurut **Connolly (2002, p457)**, setiap atribut memiliki *domain*, yaitu himpunan nilai yang dibolehkan. *Constraint* ini harus diidentifikasikan pada saat pemilihan *attribute domain* untuk *data model* (langkah 1.4).

c. Entity integrity

Primary key dari sebuah entitas tidak boleh bernilai null. *Constraint* ini harus dipertimbangkan pada saat penentuan *primary key* bagi setiap tipe entitas (langkah 1.5).

d. Referential integrity

Jika suatu *foreign key* memiliki nilai, maka nilai tersebut harus menunjuk ke sebuah baris yang ada pada relasi '*parent*'.

e. Enterprise constraint (business rules)

Kegiatan *update* entitas dibatasi oleh peraturan atau kebijakan organisasi yang mengatur transaksi yang diwakilkan oleh *update* yang dilakukan.

Langkah 2.6 - Meninjau kembali *local logical data model* yang dibuat dengan pengguna

Tujuan yang ingin dicapai adalah untuk memastikan bahwa *local logical data model* dan dokumentasi pendukung yang menggambarkan model merupakan perwakilan yang benar dari pandangan pengguna.

Langkah 3 - Membangun dan validasi *global logical data model*

Bertujuan menggabungkan masing-masing *local logical data model* menjadi sebuah *global logical data model* yang menggambarkan organisasi dengan menyatukan masing-masing *local logical data model* bagi setiap pandangan pengguna. Kegiatan yang dilakukan pada langkah ini meliputi :

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

Untuk setiap *local logical data model* telah dihasilkan sebuah diagram *Entity-Relationship* (*ER diagram*), skema relasional, kamus data, dan dokumentasi pendukung yang menggambarkan batasan-batasan model. Komponen-komponen tersebut digunakan untuk mengidentifikasikan persamaan dan perbedaan antara model-model dan oleh karena itu akan membantu menyatukan model-model tersebut.

Langkah 3.2 - Validasi *global logical data model*

Bertujuan untuk memvalidasi relasi-relasi yang dibuat dari *global logical data model* dengan menggunakan teknik normalisasi dan untuk memastikan bahwa model tersebut mendukung transaksi-transaksi yang dibutuhkan.

Langkah 3.3 - Memeriksa kemungkinan perkembangan di masa yang akan datang

Bertujuan untuk menentukan kemungkinan adanya perubahan yang berarti pada waktu mendatang dan untuk memperkirakan apakah *global logical data model* yang ada dapat menyesuaikan dengan perubahan tersebut.

Langkah 3.4 - Meninjau kembali *global logical data model* dengan para pengguna

Bertujuan untuk memastikan bahwa *global logical data model* merupakan representasi yang benar dari organisasi.

3. Perencanaan basisdata fisikal *(Physical Database Design)*

Menurut **Connolly (2002, p282)**, perancangan basis data fisikal (*physical database design*) adalah proses untuk menghasilkan penjelasan dari

pengimplementasian suatu basis data pada media penyimpanan kedua, juga menjelaskan *base relation*, pengaturan file, dan indeks yang digunakan untuk mencapai akses data yang efisien, *integrity constraint*, serta ukuran keamanan. Langkah-langkah metodologi perancangan basis data fisikal terdiri dari:

Langkah 4 - Menterjemahkan global logical data model untuk target DBMS

Bertujuan untuk menghasilkan skema basis data relasional bagi *global logical data model* yang dapat diimplementasikan pada target *DBMS*.

Langkah 4.1 - Merancang Relasi Dasar

Untuk setiap relasi yang diidentifikasikan pada *global logical data model*, definisinya terdiri dari nama relasi, daftar *simple attribute* diikuti tanda kurung, *primary key* berserta *alternate key* dan *foreign key* jika ada, dan *referential integrity constraint* bagi *foreign key* yang teridentifikasi.

Definisi setiap atribut pada kamus data terdiri dari *domain*nya yang terdiri atas tipe data, panjang data, setiap *constraint* pada *domain*, nilai *default*nya jika ada, dan keterangan apakah atribut dapat memiliki nilai *null*.

Langkah 4.2 - Merancang representasi *derived data*

Bertujuan untuk menentukan cara untuk merepresentasikan *derived data* yang ada dalam *global logical data model* ke dalam target DBMS.

Biasanya *derived attribute* tidak terlihat pada *logical data model* namun didokumentasikan di dalam kamus data. Untuk setiap *derived attribute* yang ada, tanda ‘/’ digunakan untuk menandakan atribut tersebut adalah *derived attribute*.

Langkah 4.3 - Merancang *enterprise constraints*

Bertujuan untuk merancang batasan-batasan organisasi untuk target DBMS. *Update* terhadap relasi dibatasi oleh peraturan organisasi yang mengatur transaksi '*real world*' yang diwakili oleh *update* tersebut.

Langkah 5 - Membuat representasi fisikal

Bertujuan untuk menentukan organisasi file yang optimal untuk menyimpan *base relation* dan indeks yang diperlukan untuk mencapai unjuk kerja yang sesuai, dengan cara penentuan penyimpanan relasi dan baris-baris pada tempat penyimpanan kedua. Ada tiga faktor untuk mengukur efisiensi penyimpanan data yaitu:

- ***Transaction throughput*:** menurut **Connolly (2002, p484)**, adalah jumlah transaksi yang dapat diproses pada rentang waktu yang diberikan.
- ***Response Time***: menurut **Connolly (2002, p484)**, merupakan waktu yang diperlukan untuk menyelesaikan sebuah transaksi.
- ***Disk storage*:** menurut **Connolly (2002, p484)**, merupakan besarnya ukuran penyimpanan untuk menyimpan basis data.

Untuk meningkatkan unjuk kerja, cara interaksi antara empat komponen dasar perangkat keras harus diperhatikan karena berpengaruh terhadap unjuk kerja sistem. Komponen-komponen itu antara lain:

- ***Main memory***

  Semakin besar memori utama yang disediakan untuk DBMS dan aplikasinya, semakin cepat aplikasi akan berjalan.

- ***CPU***

  CPU mengontrol tugas-tugas dari sumber daya sistem lain dan mengeksekusi proses yang diminta oleh pengguna. Komponen ini harus dijaga agar tidak terjadi *bottleneck*.

- ***Disk I/O***

  Penyimpanannya harus didistribusikan secara merata ke semua *drive* yang ada untuk mengurangi masalah pada untuk kerja sistem. Prinsip dasar dalam pendistribusian data yaitu:

  - File sistem operasi harus dipisahkan dari file basis data
  - File utama basis data harus dipisahkan dari file indeks
  - File *log recovery* harus dipisahkan dari basis data yang sedang tidak digunakan

- **Network**

  Ketika jumlah lalu-lintas data menjadi besar, atau ketika jumlah tabrakan di dalam jaringan menjadi besar, akan terjadi *bottleneck* di dalam jaringan.

Langkah 5.1 - Analisis transaksi

Bertujuan untuk memahami fungsi dari transaksi yang dijalankan pada basis data dan menganalisis transaksi-transaksi yang penting. Dalam menganalisis transaksi, kriteria unjuk kerja yang harus diidentifikasi seperti:

- Transaksi yang sering digunakan dan yang memiliki dampak yang signifikan pada unjuk kerja
- Transaksi yang penting bagi kegiatan operasional bisnis

- *Peak load*, dimana menurut **Connolly (2002, p486)** merupakan saat-saat pada hari atau minggu dimana akan ada permintaan yang tinggi terhadap basis data

Langkah 5.2 - Memilih organisasi file

Untuk menentukan organisasi file yang efisien untuk setiap *base relation*.

Langkah 5.3 - Memilih indeks

Untuk menentukan apakah penambahan indeks akan meningkatkan unjuk kerja sistem. Ada tiga jenis indeks yaitu:

- ***Primary index***

  Pengindeksan dilakukan pada kolom kunci (*key field*), yang diurutkan terlebih dahulu secara sekuensial.

- ***Clustering index***

  Pengindeksan dilakukan pada kolom bukan kunci (*non-key field*), yang sudah diurutkan terlebih dahulu secara sekuensial. Menurut **Connolly (2002, p1155)**, kolom bukan kunci itu disebut juga dengan *clustering attribute*.

- ***Secondary index***

  Menurut **Connolly (2002, p1155)**, merupakan pengindeksan yang dilakukan pada kolom yang tidak terurut di dalam file data.

Langkah 5.4 - Memperkirakan kebutuhan *disk space*

Memperkirakan besarnya ruang penyimpanan yang dibutuhkan untuk mendukung implementasi basis data pada tempat penyimpanan kedua.

Hal ini sangat tergantung pada target DBMS dan perangkat keras yang digunakan. Perkiraan ukuran dapat dilakukan dengan mengukur besar data tiap baris dan jumlah baris pada setiap relasi.

Langkah 6 - Merancang *user view*

Bertujuan untuk merancang pandangan pengguna yang diidentifikasikan selama tahap pengumpulan kebutuhan dan analisa pada daur hidup aplikasi basis data relasional (*database application lifecycle*).

Langkah 7 - Mekanisme Keamanan

Bertujuan untuk menentukan bagaimana kebutuhan keamanan akan direalisasikan. Keamanan bagi basis data sangat diperlukan karena basis data merupakan sumber daya perusahaan yang penting. Dua tipe keamanan basis data menurut **Connolly (2002, p502)**, yaitu:

- **Keamanan sistem**

  Memberikan perlindungan terhadap akses dan penggunaan basis data pada tingkat sistem, seperti *user name* dan *password.*

- **Keamanan data**

  Memberikan perlindungan akses dan penggunaan objek basis data, sepeti relasi dan *view* dan aksi terhadap objek yang dapat dimiliki oleh pemakai.

Langkah 8 - Mempertimbangkan pengenalan pengontrolan *redundancy*

Untuk menentukan apakah pengenalan pengontrolan *redundancy* dengan mengendurkan aturan normalisasi akan meningkatkan unjuk kerja sistem.

Langkah 9 - Memantau operasional sistem

Bertujuan untuk memantau operasional sistem dan meningkatkan unjuk kerja sistem untuk memperbaiki keputusan desain yang tidak sesuai atau menggambarkan kebutuhan-kebutuhan perubahan.

- **DBMS *Selection (optional)***

  DBMS (*Database Management System*) menurut **Whitten (2004, p760)** adalah perangkat lunak khusus yang digunakan untuk membuat, mengakses, mengontrol, dan mengatur sebuah basis data.

  Karena suatu organisasi memerlukan perluasan atau perubahan pada sistem yang sedang berjalan, maka akan menjadi hal yang perlu untuk mengevaluasi produk-produk DBMS yang baru. Tujuannya untuk memilih sebuah sistem yang sesuai dengan kebutuhan perusahaan saat ini maupun di masa yang akan datang, yang seimbang dengan biaya-biaya yang dikeluarkan termasuk dalam pembelian produk DBMS, perangkat lunak maupun perangkat keras tambahan yang dibutuhkan untuk mendukung sistem basis data, dan biaya-biaya lain yang berhubungan dengan perubahan dan pelatihan pegawai. Tahapan utama dalam memilih DBMS antara lain:

  1) Mendefinisikan syarat-syarat sebagai referensi

  Dibuat dengan menyatakan tujuan dan ruang lingkup pembelajaran, tugas-tugas yang akan dikerjakan, penjelasan kriteria (berdasarkan spesifikasi kebutuhan pengguna) yang akan digunakan dalam mengevaluasi produk-produk DBMS, daftar produk-produk

yang dimungkinkan, semua batasan-batasan dan skala waktu yang dibutuhkan untuk pembelajaran.

2) Daftar singkat dua atau tiga produk

Kriteria yang dianggap penting dalam keberhasilan implementasi dapat digunakan untuk membuat daftar produk-produk DBMS dalam evaluasi, seperti dana yang tersedia, tingkat dukungan *vendor*, kecocokan dengan perangkat lunak lainnya, dan apakah produk hanya berjalan pada perangkat keras tertentu.

3) Evaluasi produk

Fitur-fitur yang digunakan dalam evaluasi produk-produk DBMS dikelompokkan menjadi definisi data, definisi fisik, kemampuan akses, penanganan keperluan-keperluan, pengembangan, dan fitur-fitur lainnya.

4) Merekomendasikan pilihan dan memproduksi laporan

Langkah terakhir dari pemilihan DBMS adalah mendokumentasikan prosesnya dan membuat pernyataan dalam penemuan dan rekomendasi atas produk DBMS tertentu.

- ***Application Design***

Menurut **Connolly (2002, p287-288)**, perancangan aplikasi (*aplication design*) adalah merancang antarmuka pemakai (*user interface*) dan program aplikasi yang akan memproses basisdata. Perancangan basisdata dan perancangan aplikasi adalah aktivitas yang dilakukan secara bersamaan pada *database application lifecycle*. Dalam kasus

sebenarnya, tidak mungkin dapat menyelesaikan perancangan aplikasi sebelum perancangan basisdata selesai.

Dalam perancangan aplikasi harus memastikan semua pernyataan fungsional dari spesifikasi kebutuhan pemakai (*user requirement spesification*) yang menyangkut perancangan aplikasi program yang mengakses basisdata dan merancang transaksi yaitu cara akses ke basisdata dan perubahan terhadap isi basisdata (*retrieve*, *update* dan keduanya). Artinya bagaimana fungsi yang dibutuhkan bisa terpenuhi dan merancang antarmuka pemakai yang tepat. Antarmuka yang dirancang harus memberikan informasi yang dibutuhkan dengan mewujudkan sifat "*user-friendly*".

- ***Prototyping***

  Menurut **Connolly (2002, p291-292)**, *prototyping* adalah membuat model kerja dan aplikasi basisdata, yang memungkinkan perancang atau pemakai untuk mengevaluasi hasil akhir sistem, baik dari segi tampilan maupun fungsi yang dimiliki sistem. Tujuan dari pengembangan *prototype* aplikasi basisdata adalah untuk memungkinkan pemakai menggunakan *prototype* untuk mengidentifikasi kelebihan atau kekurangan sistem, dan memungkinkan perancang untuk memperbaiki atau melengkapi kelebihan dari aplikasi basisdata baru.

  Ada dua strategi *prototyping* yang umum digunakan, yaitu *requirement prototyping* dan *evolutionary prototyping*. *Requirement prototyping* yaitu menggunakan *prototype* untuk menetapkan kebutuhan dari tujuan aplikasi basisdata dan ketika kebutuhan sudah terpenuhi,

prototye tidak digunakan lagi atau dibuang. Sedangkan *evolutionary prototyping* menggunakan tujuan yang sama tapi *prototype* tetap digunakan.

- ***Implementation***

  Menurut **Connolly (2002, p292)**, implementasi adalah mendefinisikan basisdata secara eksternal, konseptual, dan internal. Implementasi merupakan realisasi dari basisdata dan perancangan aplikasi. Implementasi basisdata dicapai dengan menggunakan *Data Definition Language* (DDL) dari DBMS yang dipilih atau *Graphical User Interface* (GUI). Pernyataan DDL digunakan untuk membuat struktur basisdata dan *file* basisdata kosong. Pandangan pemakai (*user view*) lainnya juga diimplementasikan dalam tahapan ini. Bagian dari aplikasi program adalah transaksi basisdata yang diimplementasikan dengan menggunakan *Data Manipulation Language* (DML) dari sasaran DBMS, mungkin termasuk *host programming language* seperti: Visuall Basic, Delphi, C, C++, Java, Cobol, atau Pascal.

- ***Data Conversion and Loading***

  Menurut **Connolly (2002, p292-293)**, *data conversion and loading* adalah mengambil data dari sistem yang lama untuk dipindahkan ke dalam sistem yang baru. Tahapan ini dibutuhkan ketika system basisdata baru menggantikan system yang lama. Pada masa sekarang, umumnnya DBMS memiliki kegunaan untuk memasukkan *file* ke dalam basisdata baru. Biasanya membutuhkan spesifikasi dari sumber *file* dan sasaran basisdatanya. Kegunaan ini memungkinkan pengembang untuk

mengkonversi dan menggunakan aplikasi program lama untuk digunakan oleh system baru. Ketika *conversion and loading* dibutuhkan, prosesnya harus direncanakan untuk memastikan kelancaran transaksi dari keseluruhan operasi.

- ***Testing***

  Menurut **Connolly (2002, p293)**, *testing* adalah proses menjalankan program aplikasi untuk menemukan kesalahan-kesalahan. Sebelum digunakan, aplikasi basisdata yang baru dikembangkan harus diuji secara menyeluruh. Untuk mencapainya harus hati-hati dalam menggunakan perencanaan strategi uji dan menggunakan data asli untuk semua proses pengujian. Di dalam definisi testingini tidak menggunakan pandangan yang biasa, testing adalah proses demonstrasi tanpa kesalahan. Dalam kenyataannya testing tidak luput dari kesalahan. Jika *testing* menunjukkan keberhasilan, maka pengujian akan menemukan kesalahan pada program aplikasi dan mungkin struktur basisdatanya.

  Di dalam merancang basisdata, pemakai yang menggunakan system baru seharusnya terlibat di dalam proses *testing*. Situasi yang ideal untuk melakukan uji system adalah menguji basisdata pada perangkat keras yang berbeda, tetapi hal ini sering tidak dilakukan. Jika data yang asli digunakan, perlu *backup* untuk mengantisipasi kesalahan. Setelah *testing* selesai, system aplikasi siap digunakan dan diserahkan kepada pemakai.

- ***Operational Maintenance***

  Menurut **Connolly (2002, p293-294)**, *operational maintenance* adalah proses memantau dan memelihara system setelah diinstal. Pada

tahapan sebelumnya, basisdata benar-benar diuji dan diimplementasikan. Sekarang system beralih pada tahapan pemeliharaan. Yang temasuk aktivitas dari tahapan pemeliharaan adalah sebagai berikut:

1. Memantau kinerja dari system. Jika kinerjanya menurun di bawah level yang dapat diterima, mungkin basisdata perlu diperbaiki.
2. Memelihara dan *upgrade* aplikasi basisdatanya (jika diperlukan).

Ketika basisdata sepenuhnya bekerja, harus dipasikan kinerjanya dapat berada dalam tingkat yang dapat diterima. Sebuah DBMS biasanya menyediakan berbagai kegunaan untuk membantu administrasi basisdata termasuk kegunaan untuk mengisi data ke dalam basisdata dan untuk memantau system. Kegunaan ini memungkinkan system pemantauan memberikan informasi tentang pemakaian basisdata dan strategi eksekusi *query*. Basisdata administrator dapat menggunakan informasi ini untuk memperbaiki system agar dapar memberikan kinerja yang lebih baik.

2.2 Teori-Teori Lain

2.2.1 Analisis SWOT

Analisis SWOT menurut **Rangkuti, F (2004, p18)** adalah identifikasi berbagai faktor secara sistematis untuk merumuskan strategi perusahaan. Analisis ini didasarkan pada logika yang dapat memaksimalkan kekuatan (*strength*), dan peluang (*opportunity*), namun secara bersamaan dapat meminimalkan kelemahan (*weakness*), dan ancaman (*threats*). Penelitian menunjukkan bahwa kinerja perusahaan dapat ditentukan oleh kombinasi faktor internal dan eksternal. Kedua faktor itu harus dipertimbangkan dalam analisis SWOT.

Secara lebih mendalam, **John, A. P. dan Robinson, R. B. (1997, p230)** menguraikan analisis SWOT sebagai berikut:

- *Strength* adalah sumber daya, keterampilan, atau keunggulan-keunggulan lain relatif terhadap pesaing dan kebutuhan pasar atau ingin dilayani oleh perusahaan. Kekuatan adalah kompetensi khusus yang memberikan keunggulan komparatif bagi perusahaan di pasar.
- *Weakness* adalah keterbatasan atau kekurangan dalam sumber daya, keterampilan, dan kapabilitas yang secara serius menghambat kinerja efektif perusahaan.
- *Opportunity* adalah situasi penting yang menguntungkan lingkungan perusahaan.
- *Threat* adalah situasi penting yang tidak menguntungkan.

2.2.2 Matriks SWOT

Matriks ini dapat menggambarkan secara jelas bagaimana peluang dan ancaman eksternal yang dihadapi perusahaan dapat disesuaikan dengan kekuatan dan kelemahan yang dimilikinya. Matriks ini dapat menghasilkan 4 set kemungkinan alternatif strategi.

Keterangan dari matriks SWOT :

a. Strategi SO

Strategi ini dibuat berdasarkan jalan pikiran perusahaan, yaitu dengan memanfaatkan seluruh kekuatan untuk merebut dan memanfaatkan peluang sebesar-besarnya.

b. Strategi ST

Ini adalah strategi dalam menggunakan kekuatan yang dimiliki perusahaan untuk mengatasi ancaman.

c. Strategi WO

Strategi ini diterapkan berdasarkan pemanfaatan peluang yang ada dengan cara meminimalkan kelemahan yang ada.

d. Strategi WT

Strategi ini didasarkan pada kegiatan yang bersifat defensive dan berusaha meminimalkan kelemahan yg ada serta menghindari ancaman.

2.3 Teori Mengenai Sistem Pakar

2.3.1 Sistem Pakar

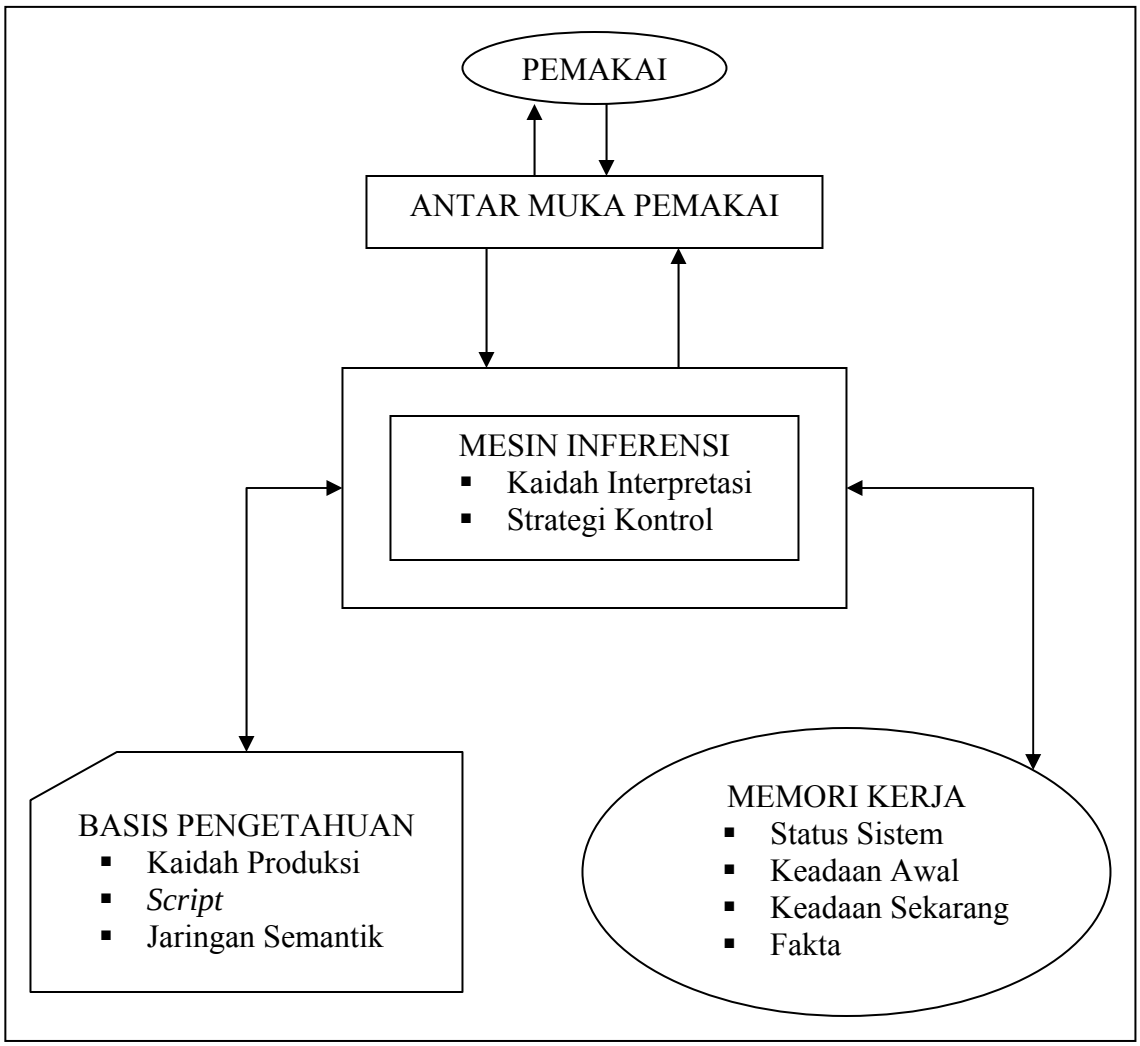
2.3.1.1 Pengertian Sistem Pakar

Menurut **Sri Kusumadewi (2003, p109),** sistem pakar (*expert system*) adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Pengertian berusaha mengadopsi pengetahuan' disini adalah komputer dapat memperoleh pengetahuan yang dibutuhkan melalui upaya yang diberikan oleh seorang *programmer* yang dikenal sebagai seorang perekayasa pengetahuan (*knowledge engineer*). Dimana pengetahuan dipandang sebagai suatu informasi yang terorganisasi dan teranalisa sehingga mudah dimengerti dan dapat diterapkan dalam memecahkan suatu persoalan atau mengambil suatu keputusan.

Sistem pakar terdiri dari tiga komponen utama, yaitu:

- Antar muka pemakai (*user interface)*
- Mesin inferensi
- Pangkalan pengetahuan

Basis pengetahuan berisi semua fakta, ide, hubungan, dan interaksi suatu domain kecil. Mesin inferensi bertugas melacak pangkalan pengetahuan untuk mencari fakta dan hubungannya yang relevan, sehingga dapat menganalisa pengetahuan dan menarik kesimpulan berdasarkan pangkalan pengetahuan, seperti yang terlihat pada **Gambar 2-15**. Dengan kedua komponen ini komputer dapat ditingkatkan daya gunanya hingga dapat menyerupai unjuk kerja seorang pakar dibidangnya.perangkat antar muka pemakai berfungsi sebagai media untuk memasukkan pengetahuan ke dalam pangkalan pengetahuan dan melakukan komunikasi dengan pemakai.



Gambar 2-15 Diagram Sistem Pakar

2.3.1.2 Proses Pemecahan Masalah

Proses pemecahan masalah dalam sistem pakar meliputi:

A. Representasi Pengetahuan

Bagian utama dari sistem pakar adalah pangkalan pengetahuan, sehingga dalam memprogram sistem pakar, langkah awalnya adalah membangun pangkalan pengetahuan. Upaya pengumpulan dan pengorganisasian pengetahuan dari para ahli maupun buku/jurnal ke dalam pangkalan pengetahuan merupakan suatu pekerjaan yang paling rumit dan memerlukan waktu terlama dalam proses pembuatan sistem pakar. Pangkalan pengetahuan mempunyai dua karakteristik, yaitu:

a. Diprogram dengan suatu bahasa komputer kemudian disimpan dalam memori.
b. Dirancang agar fakta-fakta dan pengetahuan yang terkandung di dalamnya bisa digunakan untuk penalaran.

Untuk memudahkan dalam membangun pangkalan pengetahuan maka mula-mula pengetahuan yang telah terkumpul dipresentasikan lebih dahulu, biasanya dalam merepresentasikan pengetahuan digunakan:

a. Jaringan Semantik

Jaringan semantik menyerupai jaringan pohon (tree), bedanya adalah bentuk strukturnya tidak beraturan. Pada dasarnya jaringan semantik merupakan gambaran pengetahuan grafis yang menunjukkan hubungan antar berbagai objek. Juga, pada jaringan semantik, node-node yang ada menggambarkan objek dan informasi deskriptif tentang objek-objek yang dapat berupa benda, pikiran , kejadian, atau tindakan.

b. Script

Script digunakan untuk menggambarkan urutan peristiwa. Dalam menggambarkan urutan peristiwa, *script* menggunakan serangkaian *slot* yang berisi informasi tentang orang, objek, dan tindakan yang terjadi pada suatu peristiwa.

c. Kaidah Produksi

Kaidah produksi merupakan kaidah atau produksi yang terdiri dari dua *statement* yang membentuk suatu pengetahuan. Kaidah bagian pertama disebut *anticedent* berfungsi mengekspresikan situasi atau premis, sedangkan bagian kedua disebut konsekuen yang menyatakan suatu tindakan tertentu atau konklusi yang diharapkan jika situasi atau premis benar. Bentuk produksi adalah sebagai berikut:

ANTICENDENT	-	KONSEKUEN
SITUASI	-	TINDAKAN
PREMIS	-	KONKLUSI

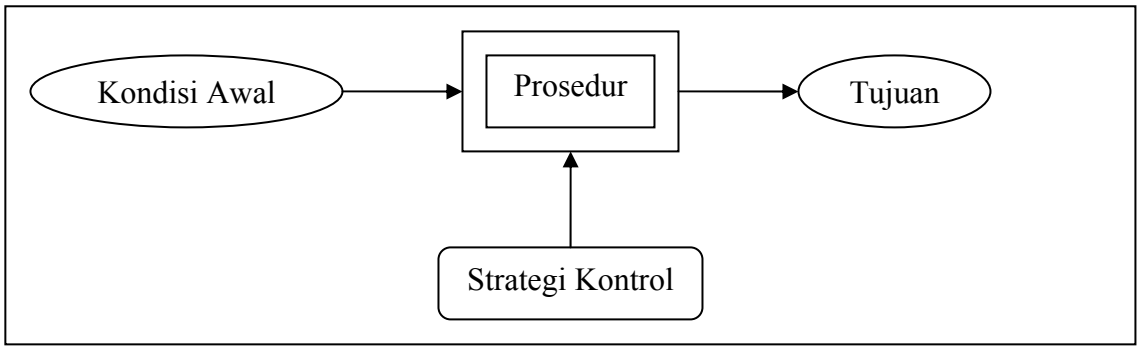
Biasanya yang pertama atau bagian kiri merupakan *statement* dengan awalan **IF**, sedanga yang kedua merupakan *statement* dengan awalan **THEN**.

Pada tugas akhir ini, dalam proses membangun pangkalan pengetahuan digunakan alat presentasi dengan memakai Kaidah Produksi.

B. Pendekatan Penyelesaian Masalah

Dalam setiap pemecahan masalah melibatkan tiga unsur utama yaitu keadaan awal masalah atau disebut *initial state*, tujuan, dan operator. Keadaan awal menyatakan dengan jelas situasi masalah dan keberadaan kondisi-kondisinya. Dari

kondisi awal ini kemudian dimanipulasi oleh prosedur dengan suatu strategi kontrol tertentu untuk mencapai tujuan.



Gambar 2-16 Hubungan antara Keadaan Awal, Prosedur, dan Tujuan

Dalam proses pelacakan pada umumnya ada dua metode yang dipakai, yaitu:

1) Pelacakan Buta

Pelacakan buta dapat dibagi menjadi empat metode, yaitu:

a. Pelacakan Pertama Melebar (Breadth First)

Dalam metode ini pelacakan dilakukan secara melebar dari setiap level dengan awal pengujian mulai dari node *root* sampai tuntas. Jadi pada pelacakan pertama melebar, node yang ada pada setiap tingkat seluruhnya diuji sebelum pindah ke tingkat berikutnya.

b. Pelacakan Pertama Mendalam (Depth First)

Pelacakan pertama mendalam bermula dari node *root* dan bergerak ke bawah ke tingkat dalam yang berurutan. Suatu operator diterapkan pada node yang menimbulkan node berikutnya dalam suatu urutan. Proses ini berlangsung terus sampai solusi ditemukan dan apabila menemui jalan buntu maka pelacakan akan dimulai lagi dari awal ke cabang lain.

c. Pelacakan Maju (Forward Chaining)

Permasalahan dirumuskan dengna jalan sedemikian rupa sehingga node-node yang mempunyai kemungkinan sebagai jawaban ditandai sebagai keadaan awal, kemudian pelacakan dapat dimulai dari satu atau beberapa keadaan awal menuju kepada keadaan tujuan. Pelacakan dengan pelacakan maju cara kerjanya seperti proses penalaran deduktif.

d. Pelacakan Mundur (Backward Chaining)

Seperti pada pelacakan maju, hanya saja prosesnya dibalik. Yaitu dari keadaan tujuan menuju pada keadaan awal, cara kerjanya mirip proses penalaran induktif.

2) Pelacakan Heuristik

Pelacakan heuristik sebenarnya merupakan pelacakan buta yang telah diberi pengarahan atau bimbingan sehingga proses pelacakannya menjadi lebih cepat.

2.3.2 Persediaan

2.3.2.1 Pengertian Dasar Persediaan

Persediaan (inventori) didefinisikan **Chase, Aquilano dan Jacobs (1998, p500)** sebagai persediaan dari setiap jenis barang atau bahan baku yang digunakan di dalam suatu organisasi. Sedangkan sistem inventory menurut **Chase, Aquilano dan Jacobs (1998, p501)** adalah seperangkat kebijakan dan pengendalian yang memonitor tingkat dari persediaan dan memperhitungkan tingkat persediaan yang harus dipelihara, kapan persediaan harus ditambah dan berapa banyak pesanan kembali harus dilaksanakan. Tujuan persediaan menurut **Chase, Aquilano dan Jacobs (1998, p513)** adalah:

1. Untuk dapat memelihara kemandirian dari operasi-operasi. Persediaan dari bahan baku pada pusat kerja menyebabkan adanya fleksibilitas dalam operasi.
2. Untuk dapat memenuhi variasi permintaan. Jika permintaan dapat diketahui dengan tepat maka mungkin bagi kita untuk dapat memproduksi tepat sesuai dengan permintaan. Bagaimanapun juga permintaan tidak dapat diketahui secara pasti dan persediaan dapat mengatasi keragaman dari jumlah permintaan.
3. Untuk dapat menghasilkan sebuah fleksibilitas dalam penjadwalan produksi. Dengan adanya persediaan memperpanjang waktu bagi perencanaan produksi untuk menghasilkan barang.
4. Untuk dapat melakukan pengamanan terhadap variasi dari waktu pengiriman bahan baku. Ketika bahan baku kita pesan penundaan pengiriman dapat terjadi karena berbagai alasan misalnya : pengiriman yang salah, persediaan dari *supplier* yang tidak mencukupi dan lain-lain.
5. Untuk mendapatkan keuntungan dari ukuran pemesanan ekonomis. Ada beberapa biaya yang harus dikeluarkan apabila kita memesan barang yaitu biaya buruh, biaya telepon, biaya pengetikan, biaya pengiriman pos dan lain-lain. Semakin besar pengepakkan semakin kecil harga satuan per unit.

2.3.2.2 Tipe Dari Inventory

Inventory menurut **Tersine (1994, p3)** terdiri dari *supplies*, bahan baku, produk setengah jadi dan barang jadi. *Supplies* adalah item dari inventory yang digunakan pada suatu fungsi organisasi yang bukan merupakan bagian dari produk jadi. Contoh *supplies* adalah pensil, kertas, lampu, disket, alat potong, dan fasilitas perawatan lainnya. Bahan baku adalah item yang dibeli dari *supplier* untuk digunakan sebagai input daripada proses produksi. Bahan baku akan diubah menjadi barang jadi. Contoh bahan baku

pabrik mebel adalah kayu, lem, skrup, paku, cat, pernis dan lain-lain. Produk setengah jadi adalah produk yang proses produksinya belum selesai dan masih berada dalam proses produksi. Produk setengah jadi mencerminkan akumulasi dari pekerjaan yang belum selesai dan antrian terhadap material dalam proses produksi. Barang jadi adalah produk final, yang sudah dapat dijual, didistribusikan atau disimpan.

Tugas inventory untuk semua kategori ini tergantung dari komponen yang diamati. Ini disebabkan karena produk jadi dari satu komponen mungkin menjadi bahan baku bagi komponen yang lainnya. Konsumen dari inventory barang jadi mungkin konsumen biasa, perusahaan retail , distributor atau pabrik lainnya.

2.3.2.3 Jenis-Jenis Persediaan

Penanganan modal dalam persediaan mempermudah perusahaan untuk melancarkan proses produksi dan untuk mengantisipasi permintaan yang mudah berubah. Pada bagian ini, persediaan dapat digolongkan dalam dua bentuk, yaitu berdasarkan fungsi dan berdasarkan proses yang dialami.

A. Persediaan Berdasarkan Fungsi

Jenis-jenis persediaan menurut **Tersine (1994, p7-8)** yang umum dimiliki pada suatu perusahaan diantaranya adalah sebagai berikut:

1. ***Working Stock* (*Cycle atau Lot Size Stock*).**

   Adalah persediaan yang diperlukan dan disimpan sebelum diperlukan agar ini bertujuan untuk meminimasikan biaya pemesanan dan penyimpanan, dan mendapatkan potongan harga. Secara umum, jumlah rata-rata persediaan di "tangan" yang dihasilkan dari ukuran lot membentuk stok aktif suatu organisasi.

2. ***Safety Stock* (*Buffer atau Fluctuation Stock*).**

   Adalah persediaan yang disimpan untuk mengantisipasi kemungkinan supply

dan demand yang tidak pasti. Dalam siklus pemenuhan kembali, stok ini berfungsi sebagai tameng terhadap kekurangan stok.

3. ***Anticipation Stock* (*Seasonal* atau *Stabilization Stock*).**

   Adalah persediaan yang digunakan untuk menghadapi permintaan musiman yang memuncak, keperluan sampingan (promosi, pemogokan buruh, penutupan karena libur). Stok ini disediakan atau diproduksi sebelum diperlukan dan berkurang selama permintaan puncak, dengan harapan agar tingkat produksi rata-rata tetap tercapai dan jumlah tenaga kerja tetap stabil.

B. Persediaan Berdasarkan Proses Produksi

Persediaan menurut **Sofjan Assauri (1993, p222-223)** dapat dikelompokkan menurut bentuknya, dimana hal ini berkaitan dengan jenis dan posisi barang tersebut dalam urutan pengerjaan produk, yaitu:

1. **Persediaan bahan baku mentah (*Raw Materials*)**

   Yaitu persediaan dari barang-barang berwujud yang digunakan dalam proses produksi, barang mana yang dapat diperoleh dari sumber alam atau dibeli dan pemasok ataupun perusahaan yang menghasilkan bahan baku bagi perusahaan yang menggunakannya.

2. **Persediaan komponen (*Componen-part*)**

   Yaitu persediaan yang terdiri dari komponen-komponen yang diterima dari perusahaan lain, yang dapat secara langsung dirakit dengan komponen lain, tanpa melalui proses produksi sebelumnya.

3. **Persediaan barang setengah jadi atau persediaan barang dalam proses (*Work in Process*)**

   Yaitu persediaan yang keluar dari tiap-tiap bagian dalam suatu pabrik atau

bahan-bahan yang telah diolah menjadi suatu bentuk, tetapi masih perlu diproses kembali untuk kemudian dijual sebagai barang jadi.

4. **Persediaan barang jadi (*Finished Good*)**

   Yaitu persediaan yang telah selesai diproses atau diolah dalam pabrik dan siap dijual kepada pelanggan atau perusahaan lain.

5. **Persediaan bahan-bahan pembantu (*Supplies Stock*)**

   Yaitu persediaan barang atau bahan yang diperlukan dalam proses produksi untuk membantu berhasilnya produksi atau yang dipergunakan dalam bekerjanya suatu perusahaan tetapi tidak merupakan bagian atau komponen dari barang jadi.

2.3.2.4 Pengujian Distribusi Normal

Sebaran peluang kontinu yang paling penting dalam statistika adalah sebaran/distribusi normal dengan kurvanya yang berbentuk genta. Untuk mengetahui apakah suatu populasi mengikuti sebaran normal atau tidak., dapat digunakan uji Goodness of Fit, yaitu uji yang digunakan untuk menentukan apakah populasi memiliki suatu distribusi teoritik tertentu. Uji ini didasarkan pada seberapa baik kesesuaian antara frekuensi yang teramati dalam data sampel dengan frekuensi harapan pada distribusi yang dihipotesiskan.

Sebuah alternatif penting untuk menguji kesesuaian distribusi adalah metode yang ditemukan oleh dua matematikawan Rusia : Kolmogorov dan smirnov pada akhir dekade 1930. Dengan uji Kolmogorov Smirnov, dapat diperiksa apakah distribusi nilai-nilai sampel yang teramati sesuai dengan distribusi teoritis tertentu. Uji kolmogorov Smirnov beranggapan bahwa distribusi variabel yang sedang diuji bersifat kontinu dan sampel dicuplik dari populasi secara acak sederhana.

Uji kesesuaian Kolmogorov Smirnov dapat diterapkan pada dua keadaan:

1. Menguji apakah suatu sampel mengikuti sebuah bentuk distribusi populasi teoritis.
2. Menguji apakah dua buah sampel berasal dari dua populasi identik

Prinsip daripada uji Kolmogorov smirnov adalah menghitung selisih *absolute* antara fungsi distribusi frekuensi kumulatif sampel (disebut F_s (x)) dan fungsi distribusi frekuensi teoritis (disebut $F_t(x)$) pada masing-masing interval kelas.

Hipotesis yang diuji dinyatakan sebagai berikut (dua sisi):

H_0 : F(x) = F(x) untuk semua x dari - ~ samapai +~

H_1 : F(x) / F(x) untuk paling sedikit sebuah x

Dengan F(x) adalah fungsi distribusi frekuensi kumulatif populasi pengamatan.

Statistik uji Kolmogorov smirnov merupakan selisih *absolute* terbesar antara $F_s(x)$ dan $F_t(x)$, yang kita sebut deviasi maksimum D. *Statistic* D ditulis sebagai berikut:

$$\mathbf{D = |F_s(x)+F_t(x)| \text{ maks}} \qquad \mathbf{1=1,2,3,\ldots}$$

Nilai D kemudian dibandingkan dengan nilai kritis pada tabel distribusi pencuplikan, pada ukuran sampel n dan tingkat kemaknaan. H_0 ditolak bila nilai teramati maksimum D lebih besar atau sama dengan nilai kritis D maksimum. Dengan penolakan H_0 berarti distribusi teramati dan distribusi teoritis berbeda secara bermakna. Sebaliknya dengan tidak menolak H_0 berarti tidak terdapat perbedaan bermakna antara distribusi teramati dan distribusi teoritis. Perbedaan-perbedaan yang tampak hanya disebabkan variasi pencuplikan (*sampling variation*).

Langkah-Iangkah Uji Goodness of fit distribusi normal :

1. H_0: populasi data mengikuti distribusi normal
2. H_1 : populasi data tidak mengikuti distribusi normal

3. Tentukan taraf nyata (α)
4. Daerah kritis : $P_{value} \leq \alpha$
5. Perhitungan : Bandingkan P_{value} dengan nilai α

   Untuk P_{value} lebih besar atau sama dengan α maka terima H_0 sedangkan untuk P_{value} lebih kecil dari α maka tolak H_0.
6. Kesimpulan : terima/tolak H_0 dan simpulkan bahwa populasi mengikuti/tidak mengikuti distribusi normal.

2.3.2.5 Model Persediaan Probabilistik

G. Hadley dan T.M. Whitin mengelompokkan dua jenis model pengendalian persediaan dengan permintaan stokastik, yaitu:

A. Continuous Review (Q,r)* atau *Reorder Point Model

Model ini secara umum dikenal dengan sistem pengendalian persediaan model Q, yaitu sistem pengendalian persediaan dengan ukuran tetap. Algoritma pengerjaan menurut **Hadley dan Within** untuk menghitung jumlah pemesanan optimal Q dan titik pemesanan kembali r adalah:

1. Tentukan nilai Q_w awal dari rumus EOQ.

$$Q_w = \sqrt{\frac{2A\lambda}{h}}$$

2. Dengan mempergunakan nilai Q yang diperoleh dari langkah 1, hitung nilai r_1 dengan persamaan:

$$H(r) = \frac{Q_w h}{\pi\lambda}$$

$$H(r) = \Phi\left(\frac{r_1 - \mu_L}{\sigma_L}\right)$$

dengan bantuan tabel normal z dan 1-F(z), diperoleh nilai r_1

Hitung nilai $\phi\left(\frac{r_1-\mu_L}{\sigma_L}\right)=\frac{1}{\sqrt{2\pi}}e^{-z^2/2}$

3. Dengan menggunakan nilai r_1 yang diperoleh, hitung $\hat{\eta}$ dengan persamaan :

$$\hat{\eta}=(\mu_L-r_1)\Phi\left(\frac{r_1-\mu_L}{\sigma_L}\right)+\sigma\phi\left(\frac{r_1-\mu_L}{\sigma_L}\right)$$

4. Hitung Q_1 dengan persamaan:

$$Q=\sqrt{\frac{2\lambda[A+\pi\hat{\eta}(r)]}{h}}$$

5. Selanjutnya dengan nilai Q_1 hitung kembali nilai r_2 seperti langkah ke-2.

$$H(r_2)=\frac{Q_1 h}{\pi\lambda}$$

$$H(r_2)=\Phi\left(\frac{r_2-\mu_L}{\sigma_L}\right)$$

dengan bantuan tabel normal z dan 1-F(z), diperoleh nilai r_2.

6. Langkah pengerjaan diteruskan sampai nilai r dan Q yang diperoleh mendekati nilai r dan Q pada iteraksi sebelumnya (stabil). Nilai Q dan r ini akan menjadi nilai yang optimal.

dimana :

π = biaya *backorder* (*backorder cost*)

x = permintaan pada periode sekarang

h = Biaya simpan

r = *reorder point*

K = Biaya Total

A = Biaya pesan

λ = Jumlah permintaan/tahun

μ_L = Jumlah permintaan selama *lead time*

σ_L = Standar Deviasi selama *leadtime*

B. *Periodic Review Model (R, T)*

Model ini secara umum dikenal dengan sistem pengendalian persediaan dengan waktu pemesanan tetap. Algoritma pengerjaan untuk menghitung jumlah pemesanan optimal T dan titik pemesanan maksimum R adalah **Hadley and Within (1995, p239-243)**:

1. Menentukan harga T dengan:

$$T = \sqrt{\frac{2A}{h\lambda}}$$

2. Hitung nilai R dengan persamaan:

$$\hat{H}(R;T) = \frac{ICT}{\pi}$$

$$\hat{H}(R;T) = \Phi\left(\frac{R-\mu_{L+T}}{\sigma_{L+T}}\right)$$

dengan bantuan tabel normal z dan 1-F(z), diperoleh nilai R.

Hitung nilai $\phi\left(\frac{R-\mu_{L+T}}{\sigma_{L+T}}\right) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}$

3. Dengan menggunakan nilai R yang diperoleh, hitung E(R;T) dengan persamaan:

$$E(R;T) = (\mu_{L+T} - R)\Phi\left(\frac{R-\mu_{L+T}}{\sigma_{L+T}}\right) + \sigma\phi\left(\frac{R-\mu_{L+T}}{\sigma_{L+T}}\right)$$

4. Hitung K_1 dengan persamaan:

$$K = \frac{L}{T} + IC\left[R - \mu_L - \frac{\lambda T}{2}\right] + \frac{\pi}{T}\left[(\mu_{L+T} - R)\Phi\left(\frac{R-\mu_{L+T}}{\sigma_{L+T}}\right) + \sigma_{L+T}\phi\left(\frac{R-\mu_{L+T}}{\sigma_{L+T}}\right)\right]$$

5. Ulangi langkah 2 sampai dengan 4 dengan mengubah T = T+ΔT, dimana ΔT = 1 hari, dengan prasyarat sebagai berikut:
   a. Jika K baru lebih besar daripada K sebelumnya, maka interaksi penambahan T dihentikan dan dicoba dengan menggunakan interaksi pengurangan T, yaitu dengan T = T-ΔT, sampai akhirnya diperoleh nilai T optimum dengan total biaya yang paling minimum.
   b. Jika K baru lebih kecil daripada K sebelumnya, maka interaksi penambahan T dilanjutkan dan proses dihentikan pada saat K baru memberikan hasil lebih besar daripada K sebelumnya. Harga T dan R dengan K yang terendah merupakan yang paling optimal.

      dimana :

      L = biaya pemesanan (*order cost*)

      π = biaya *backorder* (*backorder cost*)

      x = permintaan pada periode sekarang

      I = Biaya fraksi

      C = Harga bahan baku

      T = Interval pemesanan

      R = Jumlah pemesanan maksimum

      K = Biaya Total

      A = Biaya pesan

      λ = Jumlah permintaan/tahun

μ_L = Jumlah permintaan selama *lead time*

μ_{L+T} = Jumlah permintaan selama *lead time* dan periode pemesanan

σ_{L+T} = Standar Deviasi selama lead time dan periode pemesanan

2.3.2.6 Klasifikasi ABC

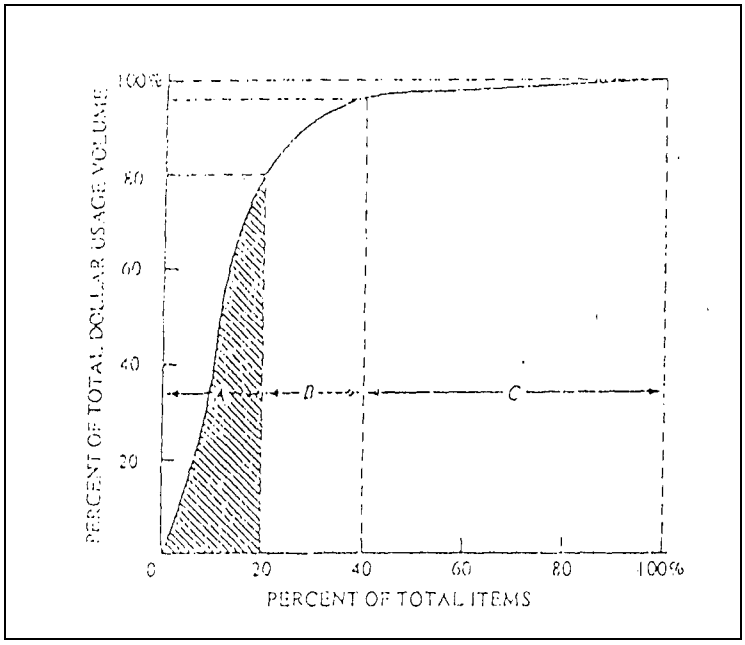
Bila dalam suatu perusahaan terdapat banyak jenis bahan (*items*) yang harus diteliti dan diawasi, maka diperlukan pengawasan persediaan yang membutuhkan banyak tenaga dan biaya. Sehingga perlu adanya kebijaksanaan pengawasan dengan mempertimbangkan keefisienan dan keefektifan, yaitu item mana saja yang memerlukan pengawasan yang agak ketat dan item mana yang pengawasannya dapat dilakukan agak longgar. Jenis bahan yang memerlukan pengawasan yang agak ketat adalah jenis bahan yang mempunyai nilai penggunaan yang cukup besar (mahal). Sebaliknya pengawasan yang agak longgar dapat dilakukan terhadap item yang mempunyai nilai penggunaan yang cukup rendah, dan biasanya terdiri dari jenis-jenis bahan yang agak banyak. Dalam penentuan kebijaksanaan pengawasan terhadap jenis-jenis bahan yang ada dalam persediaan, dapat digunakan metode analisis ABC (ABC *Analysis Method*). Metode Analisis ABC ini menggunakan "Pareto Analysis", yang menekankan bahwa sebagian kecil dari jenis-jenis bahan yang terdapat dalam persediaan mempunyai nilai penggunaan yang cukup besar yang mencakup kira-kira lebih dari 60% dari seluruh nilai penggunaan bahan yang terdapat dalam persediaan.

Metode analisis ABC ini digunakan untuk memberikan penekanan perhatian pada golongan jenis-jenis bahan yang terdapat dalam persediaan yang mempunyai nilai penggunaan yang relatif tinggi/mahal. Menurut **Sofjan Assauri (1993, p265)**, biasanya metode analisis ini dipergunakan dalam perusahaan-perusahaan yang

mempunyai berbagai jenis bahan dalam persediaan yang mempunyai nilai penggunaan yang berbeda-beda.

Dengan metode ini, persediaan dapat digolongkan atau dikelompokkan ke dalam tiga kelas sesuai dengan volume nilai penggunaan (jumlah produk dalam setahun dan biaya pembelian atau biaya produksi unit), yaitu kelompok barang A, B, dan C. Kelompok barang A terdiri dari jenis bahan yang mempunyai nilai penggunaan sekitar 75%-80% dari seluruh nilai penggunaan bahan, dan mewakili 20%-25% dari seluruh jumlah bahan yang terdapat dalam persediaan. Kelompok barang B terdiri dari jenis bahan yang mempunyai nilai penggunaan yang lebih rendah, yang mencapai 10%-15% dari seluruh nilai penggunaan bahan, dan mewakili 20%-25% dari seluruh jumlah bahan yang terdapat dalam persediaan. Sedangkan kelompok barang C terdiri dari jenis bahan yang mempunyai nilai penggunaan yang rendah, sekitar 5%-l0% dari keseluruhan nilai penggunaan bahan, tetapi jumlah bahan/barangnya mencapai 60%-65% dari seluruh jumlah bahan yang terdapat dalam persediaan.

Menurut **Tersine (1994, p547)**, nilai penggunaan persediaan untuk masing-masing jenis bahan diperoleh dengan mengalikan jumlah permintaan dalam setahun dengan biaya/harga per unit bahan. Jumlah permintaan dalam setahun digunakan untuk menghindari distorsi dan perubahanperubahan musiman. Keseluruhan persediaan diurutkan sesuai dengan nilai penggunaan yang paling besar ke yang paling kecil. Jenis-jenis bahan kemudian diklasifikasikan dengan cara seperti yang telah disebutkan diatas.



Gambar 2-17 Pareto Klasifikasi ABC

Berikut ini adalah tabel yang memuat perbandingan tingkat pengawasan, tipe pendataan, ukuran lot, frekuensi pengamatan, dan ukuran *safety stock* dari kelas A,B, dan C dari klasifikasi ABC.

Kelas	Tingkat Pengawasan	Tipe Pendataan	Ukuran Lot	Frekuensi Pengamatan	Ukuran Safety Stock
A	Ketat	Teliti dan lengkap	Rendah	Kontinu	Kecil
B	Sedang	Bagus	Menengah	Kadang-kadang	Sedang
C	Longgar	Sederhana	Besar	Jarang	Besar

Tabel 2-1 Perbandingan Kelas A, B, dan C