BAB II

LANDASAN TEORI

2.1 Sistem Pendukung Keputusan

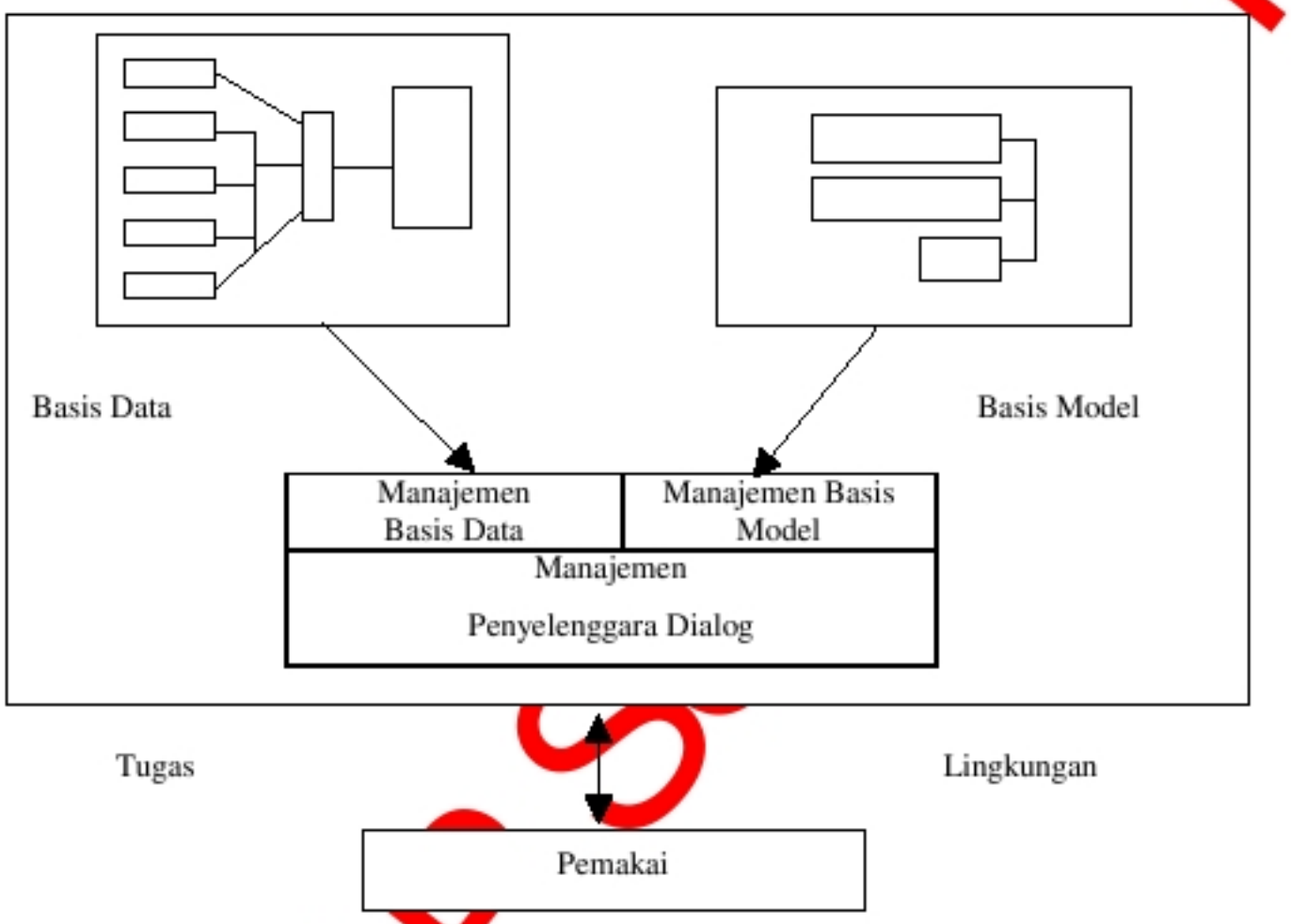
Sistem Pendukung Keputusan (SPK) merupakan suatu kumpulan prosedur pemrosesan data dan informasi yang berorientasi pada penggunaan model untuk menghasilkan berbagai jawaban yang dapat membantu manajemen dalam pengambilan keputusan dimana SPK harus sederhana, mudah dan adaptif. Adapun ciri utama dalam SPK ini yang sekaligus sebagai keunggulannya adalah kemampuan SPK untuk menyelesaikan masalah-masalah yang tidak terstruktur. Menurut Sudirman dan Widjajani (1996) mengemukakan bahwa ciri-ciri SPK yang dirumuskan oleh Alters Keen adalah :

1. SPK ditujukan untuk membantu keputusan-keputusan yang kurang terstruktur dan umumnya dihadapi oleh para manajer yang berada di tingkat puncak.
2. SPK merupakan gabungan antara kumpulan model kualitatif dan kumpulan data.
3. SPK memiliki fasilitas interaktif yang dapat mempermudah hubungan antara manusia dengan komputer.
4. SPK bersifat luwes dan dapat menyesuaikan dengan perubahan-perubahan yang terjadi.

SPK tidak dimaksudkan untuk menggantikan manajer dalam mengambil keputusan, namun manajer dan komputer bekerja sama sebagai tim pemecah masalah yang bersifat terstruktur maupun tidak terstruktur serta dilengkapi dengan sistem yang bersifat interaktif.

2.2 Komponen Sistem Pendukung Keputusan

Terdiri dari 3 subsistem yaitu subsistem manajemen basis data, subsistem manajemen basis model, dan subsistem perangkat lunak penyelenggara dialog. Hubungan dari subsistem-subsistem tersebut digambarkan pada gambar 2.1.



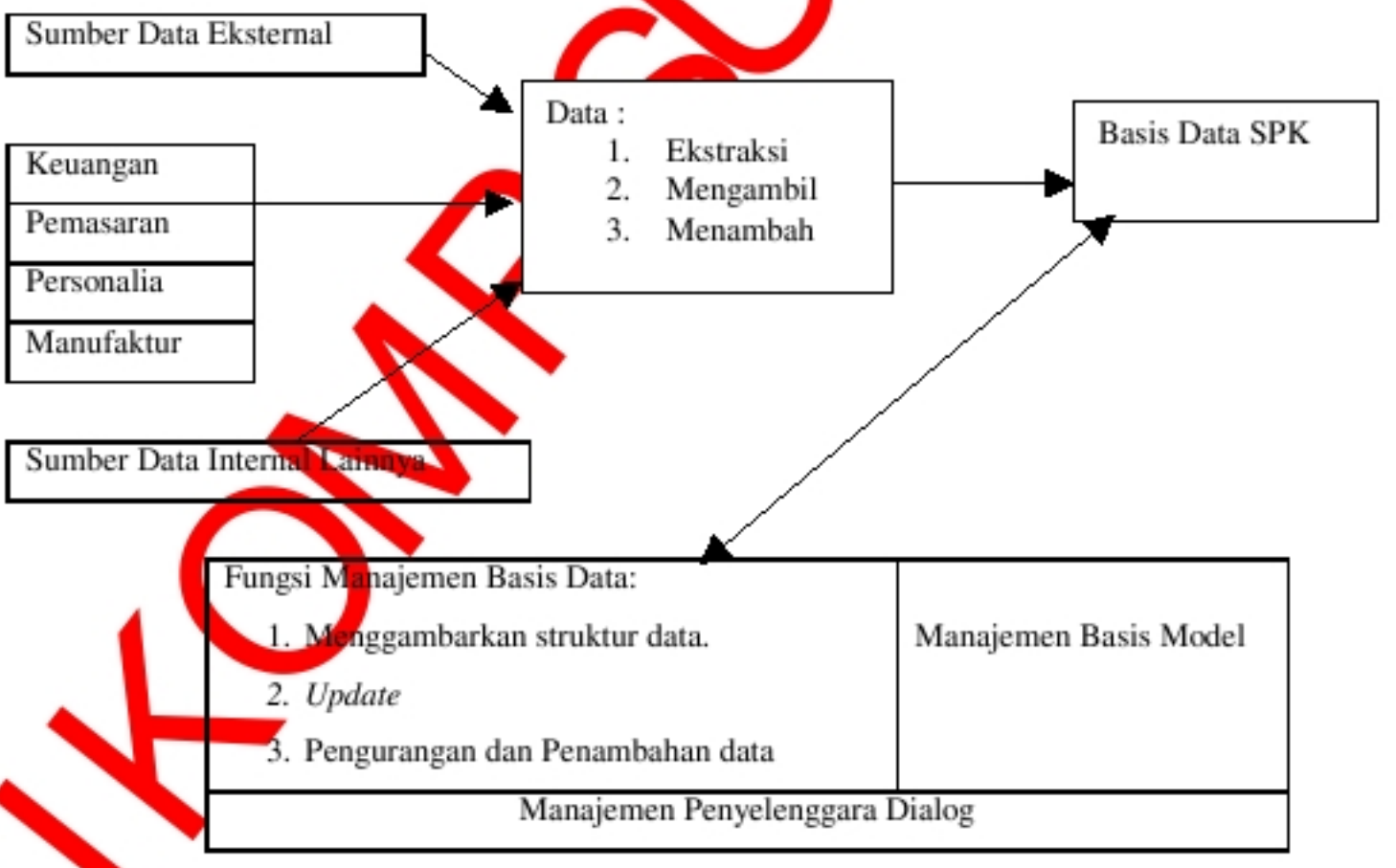
Gambar 2.1 Komponen SPK (Sprague, 1982)

a. Subsistem Manajemen Basis Data

Merupakan bagian-bagian yang menyediakan data yang dibutuhkan oleh sistem. Data ini di organisasikan dalam suatu basis data yang disebut Sistem Manajemen Basis Data (DBMS) seperti terlihat pada gambar 2.2. Ada beberapa perbedaan *database* untuk SPK dan non-SPK. Pertama sumber data untuk SPK lebih kaya daripada non-SPK. Perbedaan lain adalah proses pengambilan dan ekstraksi dari sumber data yang sangat besar. SPK membutuhkan proses ekstraksi dan DBMS yang dalam pengelolaannya harus cukup *flexible* untuk memungkinkan penambahan dan pengurangan secara

cepat. Kemampuan yang dibutuhkan dari manajemen basis data sebagai berikut:

1. Mengkombinasikan berbagai variasi data melalui pengambilan dan ekstraksi data.
2. Menambahkan sumber data dengan cepat dan mudah.
3. Menggambarkan struktur data logikal sesuai dengan pengertian pemakai sehingga pemakai mengetahui apa yang tersedia dan dapat menentukan kebutuhan penambahan dan pengurangan.
4. Menangani data secara personil sehingga pemakai dapat mencoba berbagai alternatif pertimbangan personil.
5. Kemampuan untuk mengelola berbagai variasi data.



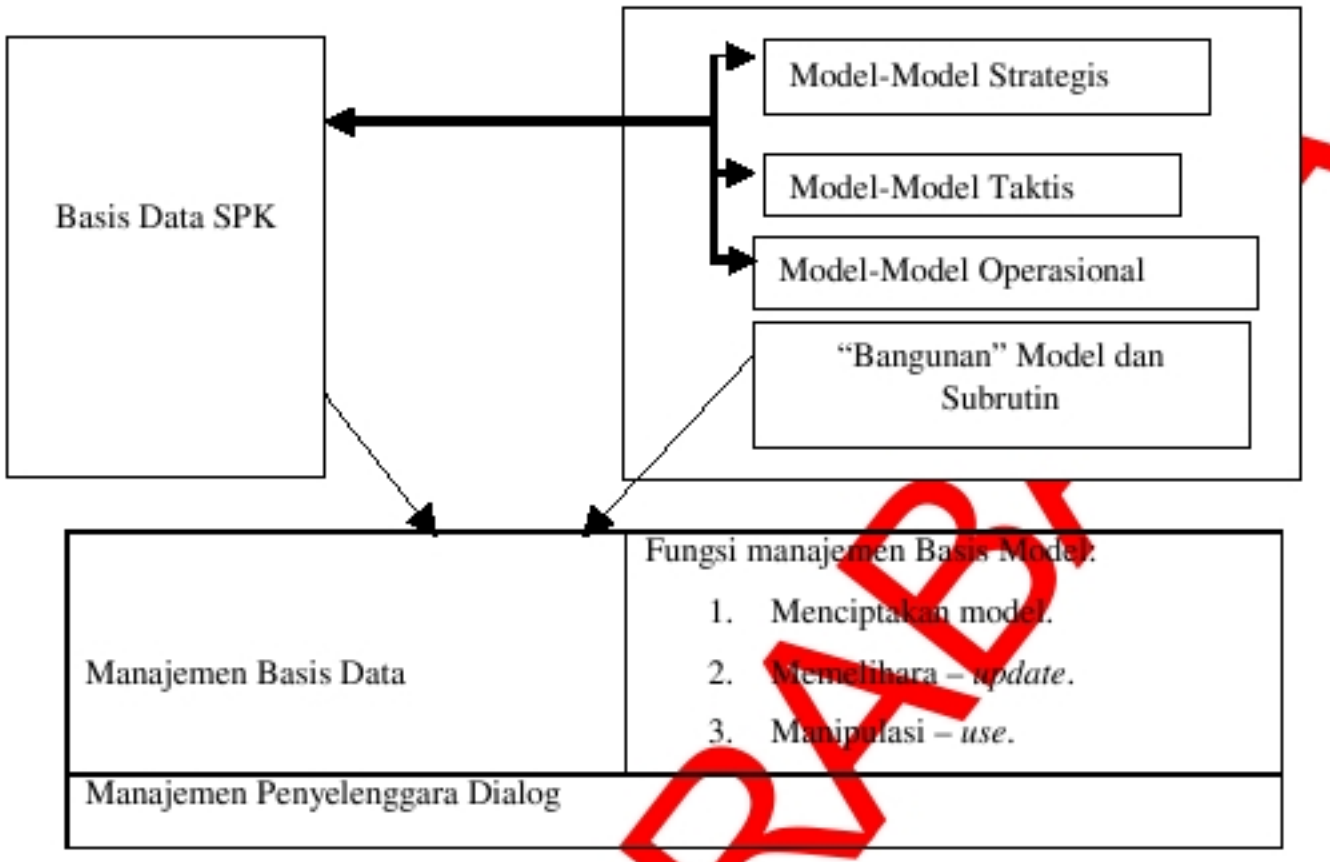
Gambar 2.2 Subsistem Manajemen Basis Data (Sprague, 1982)

b. Subsistem Manajemen Basis Model

Salah satu dari kelebihan SPK adalah kemampuannya untuk mengintegrasikan akses data dan model-model keputusan. Hal ini dapat dilakukan dengan menambah model-model keputusan ke dalam sistem informasi yang menggunakan *database* sebagai mekanisme integrasi dan komunikasi diantara model-model. Salah satu persoalan yang berkaitan dengan model adalah bahwa penyusunan model seringkali terikat pada struktur model yang mengasumsikan adanya masukan yang benar dan cara keluaran yang tepat. Sementara model cenderung tidak mencukupi adanya kesulitan dalam mengembangkan model yang terintegrasi untuk menangani sekumpulan keputusan yang saling bergantungan. Untuk menangani masalah ini dengan menggunakan koleksi berbagai model yang terpisah, dimana setiap model digunakan untuk menangani bagian yang berbeda dari masalah yang dihadapi.

Gambar 2.3 menggambarkan komponen-komponen dari subsistem model. Kemampuan yang dimiliki subsistem basis model seperti pada meliputi:

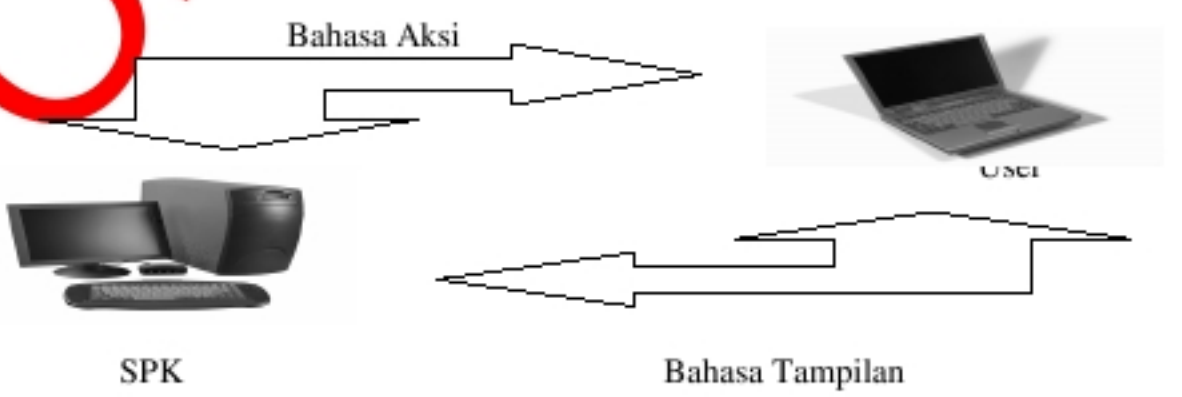
1. Menciptakan model baru secara cepat dan mudah.
2. Mengakses dan mengintegrasikan model-model keputusan.
3. Mengelola basis model dengan fungsi manajemen yang analog dan manajemen basis data.



Gambar 2.3 Subsistem Manajemen Basis Model (Sprague, 1982)

c. Subsistem Perangkat Lunak Penyelenggara Dialog

Subsistem dialog adalah fleksibilitas dan kekuatan karakteristik SPK yang timbul dari kemampuan interaksi antara sistem dan pemakai. Menurut Bennet, komponen sistem dialog adalah pemakai, terminal, dan sistem perangkat lunak dan ia membagi subsistem dialog menjadi 3 bagian yaitu bahasa aksi, bahasa tampilan, dan bahasa pengetahuan seperti pada gambar 2.4.



Gambar 2.4 Subsistem Penyelenggaraan Dialog (Sprague, 1982)

Selain itu kemampuan yang harus dimiliki oleh SPK untuk mendukung dialog pemakai/sistem meliputi:

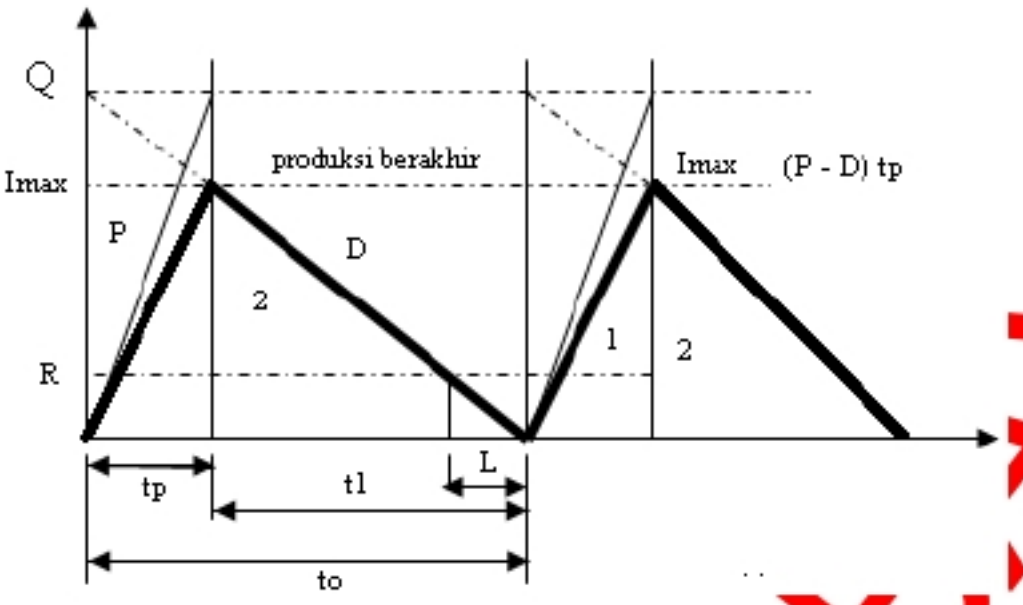
1. Menangani berbagai variasi gaya dialog.
2. Mengakomodasi tindakan pemakai dengan berbagai peralatan masukan.
3. Menampilkan data dengan berbagai variasi format dan peralatan keluaran.
4. Memberikan dukungan yang fleksibel untuk mengetahui basis pengetahuan pemakai.

2.3 Economic Production Quantity

Apabila suatu perusahaan memproduksi suatu barang dengan permintaan konstan dan dimasukkan ke dalam inventory, maka EOQ (Economic Ordering Quantity) dapat dicari dengan model EPQ, dimana *order cost* pada metode EOQ diganti dengan set-up cost.

Yang dimaksud dengan set-up cost adalah biaya yang diperlukan untuk mempersiapkan equipment atau stasiun kerja untuk melaksanakan pekerjaan tersebut. Model EPQ digunakan mencakup asumsi bahwa unit-unit ditambahkan dalam inventori saat produksi dalam proses. (Tersine, 1994).

Tujuan dari model EPQ ini adalah untuk menentukan waktu optimal antara waktu siklus produksi, sehingga diketahui waktu optimal yang tepat dari perhitungan volume produksi dan siklus produksi optimal.



Gambar 2.5 Grafik model EPQ.

Dimana :

Q = Volume Produksi

I_{max} = Persediaan maximum

R = Reproduction point (titik berproduksi kembali)

P = Rata-rata produksi

D = Rata-rata Permintaan

t_p = Waktu yang diperlukan dalam tahap produksi

t_0 = Waktu siklus produksi

L = Lead time

t_1 = Waktu rata-rata permintaan ke-t.

Metode EPQ multi item pada dasarnya sama dengan metode EPQ single item. Dengan catatan bahwa banyaknya jenis produk tersebut dibuat pada suatu siklus dan peralatan yang sama. Dengan demikian nilai optimum dari operasi produksi untuk tiap-tiap produk dapat dicari.

Maksimum persediaan atas dasar produk yaitu $(p_i - r_i).tp$ dan rata-rata persediaan adalah setengah dari keseluruhan tersebut atau

Tersine (1994:127) $\frac{(p_i - r_i)tp}{2}$ .....(2.1)

Jumlah produk yang dipakai untuk produksi adalah

Tersine (1994:127) $Q_i = pi \,.\, tp = \frac{Ri}{n}$ .....(2.2)

Dimana n adalah jumlah siklus produksi per tahun. Sehingga dengan demikian formulasi rata-rata persediaan dapat dinyatakan sebagai berikut :

Tersine (1994:127) Rata-rata persediaan $= \frac{(pi - ri)\,tp}{2} = \frac{(pi - ri)Ri}{2 \,.\, n \,.\, pi}$ .....(2.3)

Dimana :

i	=	Tipe atau jenis produk.
R	=	Kebutuhan per tahun dalam unit.
n	=	jumlah siklus produksi pertahun.
n^*	=	Jumlah siklus produksi optimal pertahun.
P	=	Biaya produksi per unit.
Q	=	Volume produksi.
Q^*	=	Volume produksi optimal.
p	=	Rata-rata produksi.
r	=	Rata-rata permintaan.
C	=	Biaya set-up per-run.
H	=	Biaya simpan per-unit dan per tahun
m	=	Banyaknya jenis produksi.

Jika *stock out* produksi dianggap tidak ada, maka total biaya produksi dapat di informasikan sebagai berikut :

Total Biaya Produksi = Biaya Produksi + Biaya Set-up + Biaya Simpan.

Tersine (1994:127) $$TC = \sum_{i=1}^{m} R_i \,.\, P_i + n \sum_{i=1}^{m} C_i + \frac{1}{2.n} \sum_{i=1}^{m} \frac{(p_i - r_i).R_i \,.\, H_i}{P_i} \quad ....(2.4)$$

Dimana :

m = Banyaknya jenis produk.

i = Produk ke-i.

Untuk memperoleh total cost yang minimum diambil turunan pertama dari TC terhadap n dan disamadengankan nol sehingga diperoleh :

Tersine (1994:127) $$\frac{dTC}{dn} = \sum_{i=1}^{m} C_i - \frac{1}{2.n^2} \sum_{i=1}^{m} \frac{(p_i - r_i).H_i \,.\, R_i}{p_i} = 0 \quad .....(2.5)$$

Jika persamaan (4) dan (5) diturunkan dan dicari hasil perhari, maka diperoleh :

Tersine (1994:127) $$n^* = \frac{\sqrt{\dfrac{\sum_{i=1}^{m} \dfrac{(p_i - r_i).R_i \,.\, H_i}{p_i}}{2 \sum_{i=1}^{m} C_i}}}{288} \quad .....(2.6)$$

Dengan demikian jumlah produk ke-i yang diproduksi dapat dihitung dengan formulasi :

Tersine (1994:128) $$Q_i^* = \frac{R_i}{n^*} \quad .....(2.7)$$

Sedangkan total cost minimumnya adalah :

Tersine (1994:128) $$TC^* = \sum_{i=1}^{m} R_i \,.\, P_i + n^* \sum_{i=1}^{m} C_i + \frac{1}{2.n^*} \sum_{i=1}^{m} \frac{(p_i - r_i).\,R_i\,.\,H_i}{P_i}$$

$$TC^* = \sum_{i=1}^{m} R_i \,.\, P_i + \frac{2\,.\,(n^*)^2 \sum_{i=1}^{m} \frac{(p_i - r_i).\,R_i\,.\,H_i}{P_i}}{2\,.\,n^*} \quad \ldots\ldots\ldots(2.8)$$

Sedangkan dari dTC = 0, diperoleh bahwa :

Tersine (1994:128) $$2\,.\,(n^*)^2 \sum_{i=1}^{m} C_i = \sum_{i=1}^{m} \frac{(p_i - r_i).\,R_i\,.\,H_i}{p_i}$$

$$TC^* = \sum_{i=1}^{m} R_i \,.\, P_i + 2\,.\,(n^*) \sum_{i=1}^{m} C_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2.9)$$

Untuk menghitung waktu optimal antara waktu siklus produksi akan digunakan formulasi :

Tersine (1994:128) $$T^* = \sum_{i=1}^{m} \frac{Q_i^*}{pi} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2.10)$$

Dan untuk menghitung waktu produksi yang tersedia tiap siklusnya digunakan formulasi :

Tersine (1994:128) $$\frac{N}{n^*} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2.11)$$

Dimana :

N = Total waktu kerja dalam satu tahun.

n* = Siklus produksi optimal.

2.4 Visual Basic 6.0

Visual Basic 6.0 Enterprise Edition merupakan *software* buatan *Microsoft* yang lebih dikhususkan untuk pembuatan desain database untuk aplikasi bisnis. *Visual Basic* menggabungkan kemudahan pemakaian suatu aplikasi pemrograman berbasis visual, kecepatan, dan kemampuan manajemen database yang disediakan terintegrasi penuh, dan mampu berkomunikasi dengan bermacam-macam tipe database. Dirancang dengan kelebihan tertentu yang juga dapat mendukung database *Microsoft Access*.

Secara umum ada beberapa manfaat yang diperoleh dari pemakaian program *Visual Basic*, diantaranya:

1. Dipakai dalam membuat program aplikasi berbasis *windows*.
2. Dipakai dalam membuat objek-objek pembantu program, seperti fasilitas *Help*, *Control Active X*, aplikasi Internet dan sebagainya.
3. Digunakan untuk menguji program (*Debugging*) dan menghasilkan program akhir *EXE* yang bersifat *Executetable*, atau dapat langsung dijalankan.

2.5 Microsoft Access 2000

Access merupakan perangkat lunak dari *Microsoft* yang dapat digunakan sebagai tool untuk membuat database yang berisi tabel yang diperlukan oleh aplikasi yang nantinya menentukan rancangan antar muka untuk aplikasi tersebut.

Dengan menggunakan *Access*, pembuatan database ini menjadi sangat mudah karena *Access* mudah dimengerti dan tidak terlalu rumit dalam proses pembuatannya.

Program database memungkinkan untuk bekerja dengan beberapa tabel. Dalam proses kerjanya, pengoperasian data pada tabel dalam database didukung oleh enam objek database, yaitu :

1. *Query* adalah sebuah objek database yang digunakan untuk menampilkan, menyunting dan menganalisis suatu data dengan cara lain.
2. *Form* adalah sebuah objek database yang digunakan untuk membuat kontrol-kontrol proses memasukkan, memeriksa dan memperbaiki data.
3. *Report* adalah sebuah objek yang digunakan untuk menampilkan data dengan format yang rendah.
4. *Pages* adalah sebuah objek khusus yang digunakan untuk menampilkan dan bekerja dengan data yang diambil dari internet atau intranet.
5. *Macro* adalah rangkaian dari beberapa perintah yang dapat disimpan dan dijalankan ulang secara otomatis.
6. *Module* adalah program-program yang ditulis *Access Basic*.

2.6 Perancangan Sistem

Perancangan adalah langkah pertama dalam fase pengembangan untuk semua sistem atau produk yang terencana. Didefinisikan sebagai proses mengaplikasikan bermacam-macam teknik dan prinsip untuk tujuan pendefinisian suatu alat, proses, atau sistem dalam detail yang cukup untuk mencapai realisasi fisik. Tujuan dari perancangan adalah untuk menghasilkan suatu model atau representasi dari *entity* yang akan dibuat. Pendekatan perancangan *software*

komputer melibatkan disiplin ilmu yang lain, berubah secara berkelanjutan, analisa yang lebih baik, dan pengetahuan yang terus berkembang.

Pentingnya perancangan *software* dapat dinyatakan dengan satu kata yaitu kualitas. Perancangan adalah tempat dimana kualitas dibantu dalam pengembangan software. Perancangan *software* adalah sebuah proses dimana permintaan diterjemahkan dalam representasi *software*. Perancangan adalah satu-satunya cara yang secara akurat menerjemahkan pemintaan customer ke dalam sistem atau produk *software* yang sudah jadi. Perancangan sistem adalah dasar dari semua *software engineering*. Tanpa perancangan, akan beresiko membangun sistem yang tidak stabil.

Beberapa karakteristik dari perancangan yang baik adalah sebagai berikut :

1. Perancangan haruslah *modular*, yaitu *software* haruslah secara logika dibagi dalam komponen-komponen yang menjalankan fungsi tertentu.
2. Perancangan haruslah berist representasi data dan prosedur yang berbeda dan terpisah.

2.6.1 Sistem Flowchart

Sistem Flowchart merupakan alat bantu yang banyak digunakan untuk menggambarkan sistem secara fisikal. Di dalam menggambarkan arus data dokumen serta proses yang dilakukan terhadap data dan dokumen dalam sistem, serta dengan cara menggunakan simbol-simbol standard yang biasa dipakai. Sistem flow dalam sistem ini ada dua yaitu Sistem Flow Manual dan Sistem Flow Komputerisasi.

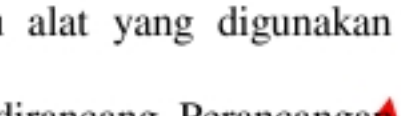
2.6.2 Data Flow Diagram (DFD)

Data Flow Diagram (DFD) adalah suatu alat yang digunakan untuk pemodelan atau menggambarkan sistem yang akan dirancang. Perancangan sistem dengan menggunakan DFD ini diawali dengan masuknya arus data kedalam proses dan dihasilkan arus data yang keluar dari proses. Dan setiap proses dilengkapi dengan penjelasan yang lengkap mengenai identifikasi proses dan nama proses. Meskipun suatu analisa yang disebut dengan DFD mempunyai struktur tersendiri, namun sistem analisa dapat meletakkan secara bersamaan sebuah gambar yang merepresentasikan seluruh proses-proses data dalam sebuah organisasi. Pendekatan data *flow* menitik beratkan pada logika yang tersirat dari suatu sistem. Dengan menggunakan kombinasi simbol, sistem analisa dapat membuat sebuah gambaran dari suatu proses yang sebenarnya dengan menggunakan dokumen sistem.

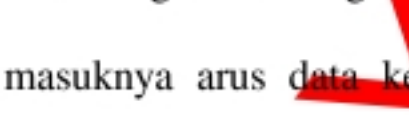
Simbol yang digunakan pada DFD dalam Tugas Akhir ini menggunakan Gane & Sarsan yaitu :

1. Arus data

2. Simpanan data



3. Kesatuan luar



4. Proses

Arus data di DFD diberi simbol suatu panah. Arus data ini digunakan untuk menunjukkan arah aliran data dari proses, kesatuan luar dan file yang dibuat.

S*impanan data* merupakan nama file untuk menyimpan data atau untuk mengambil data sesuai proses apa yang sedang di kerjakan.

Kesatuan luar merupakan kesatuan (entity) di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan input atau menerima output dari sistem.

Proses adalah kegiatan atau kerja yang dilakukan oleh orang, mesin atau komputer dari hasil suatu arus data yang akan keluar dari proses. Identifikasi proses dapat berupa angka yang ditulis pada bagian atas proses yang digunakan sebagai nomor acuan dari proses. Nama proses menunjukkan kegiatan yang dilakukan oleh proses tersebut dan diletakkan dibawah identifikasi proses.

A. Keuntungan Pembuatan Data Flow

Data *flow* mempunyai lima keuntungan utama dari penjelasan-penjelasan jalannya data dalam sistem, yaitu:

1. Kebebasan yang berasal dari kepercayaan untuk mengimplementasikan secara benar teknik sistem dari suatu sistem yang baru.
2. Memberikan pengertian dari hubungan sistem-sistem dan subsistem yang ada.
3. Komunikasi mengenai pengetahuan sistem bagi *user* melalui DFD.
4. Analisa dari sebuah usulan sistem untuk menentukan jika data dan proses-proses yang ada dapat didefinisikan secara mudah.
5. Penggunaan data *flow* merupakan keuntungan tambahan yang dapat digunakan sebagai latihan bagi sistem analis, kesempatan sistem analis menjadi lebih baik untuk mengerti tentang hubungan sistem dan subsistem yang ada didalamnya.

Keuntungan dari kelima penggunaan data *flow* tersebut dapat digunakan sebagai *tools* yang interaktif dengan *user*. Hal yang menarik dalam penggunaan DFD adalah ditunjukkannya kepada *user* gambaran-gambaran secara lengkap dari sistem. *User* dapat menanyakan guna memberikan komentar pada konsep, sistem analis dapat merubah sistem berdasarkan keinginan *user*. Keuntungan terakhir dari penggunaan DFD adalah dapat mengikuti sistem analis untuk mendeskripsikan komponen-komponen yang digunakan dalam suatu diagram. Analisa dapat ditampilkan untuk menjamin bahwa semua *output* mempunyai isi atau memperoleh data inputan dari prosesnya.

B. Pembuatan DFD (Data Flow Diagram)

DFD dapat dan harus digambarkan secara sistematis. Pertama, dibutuhkan sistem analis untuk mengkonsep data flow, dari atas ke bawah seperti ditunjukkan pada gambar 2.2 berikut:

Gambar 2.6 Pembuatan data flow diagram

Untuk memulai sebuah DFD dari suatu sistem biasanya dituangkan dalam sebuah daftar dengan empat kategori yaitu entity luar, arus data, proses, dan penyimpanan data. Daftar ini akan membantu menentukan batasan-batasan dari suatu sistem yang akan digambarkan. Pada dasarnya daftar itu berisi elemen-elemen data yang dikarang. Elemen-elemen tersebut terdiri dari:

a. Pembuatan konteks diagram

Konteks diagram adalah level yang tertinggi dalam sebuah DFD dan hanya berisi satu proses serta merupakan representasi dari sebuah sistem. Proses dimulai dengan penomeran ke-0 dan untuk seluruh entity luar akan ditunjukkan dalam konteks diagram yang sama seperti data awal yang dikirim dari entity luar. Konteks diagram tidak berisi penyimpanan data.

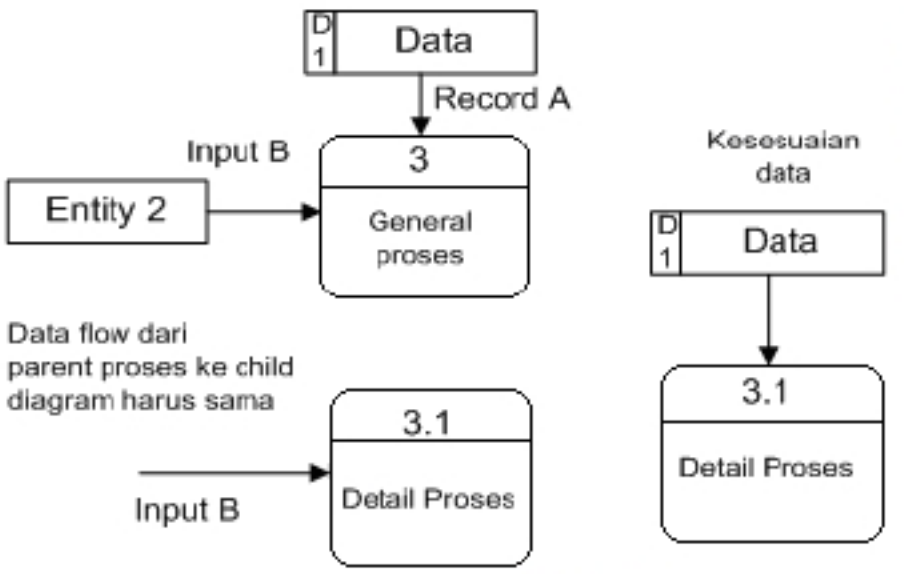
b. Pembuatan diagram level 0 serta level berikutnya

Diagram level 0 dihasilkan oleh konteks diagram dan berisi proses-proses. Pengisian proses-proses yang berlebihan pada level ini akan menghasilkan sebuah diagram yang salah, sehingga sulit untuk dimengerti. Masing-masing proses diberikan penomeran dengan sebuah integer. Umumnya dimulai dari kiri atas dan penyelesaiannya di kanan bawah dalam sebuah bentuk diagram.

c. Pembuatan *child* diagram

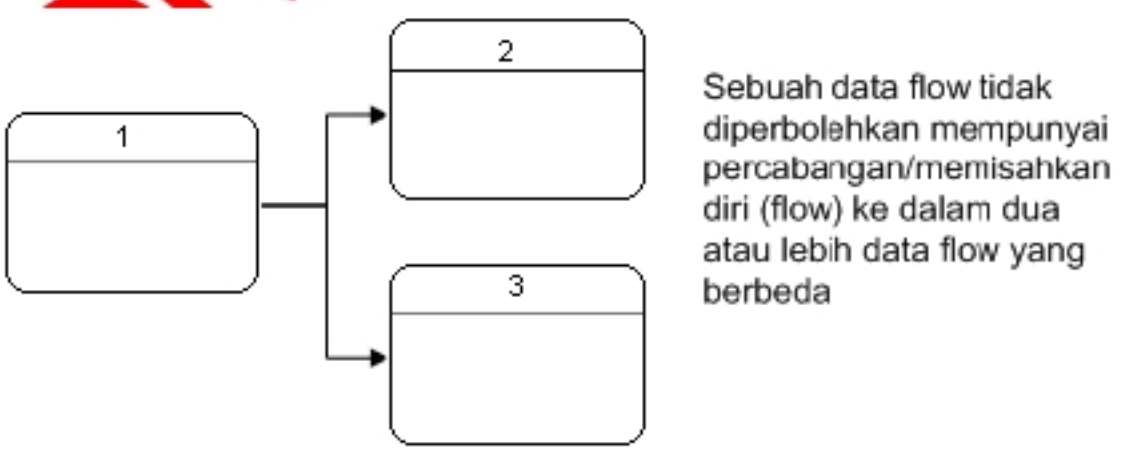
Child diagram diberikan nomor yang sama seperti proses diatasnya (parent proses) dalam diagram level 0. Contohnya, proses 3 harus diturunkan ke diagram 3, proses pada child diagram menggunakan penomeran unik untuk masing-masing proses dengan mengikuti penomeran proses diatasnya. Contohnya, dalam diagram 3 proses-proses diberikan nomor 3.1, 3.2, 3.3 dan seterusnya. Konversi ini diikuti oleh analis sistem untuk menelusuri seri-seri

dari proses-proses yang dikeluarkan oleh beberapa level, jika ada proses diagram level 0 digambarkan sebagai 1, 2, dan 3 maka *child* diagramnya 1, 2, dan 3 pada level yang sama. Ilustrasi level detil dengan sebuah *child* DFD dapat ditunjukkan pada gambar 2.3 yaitu:

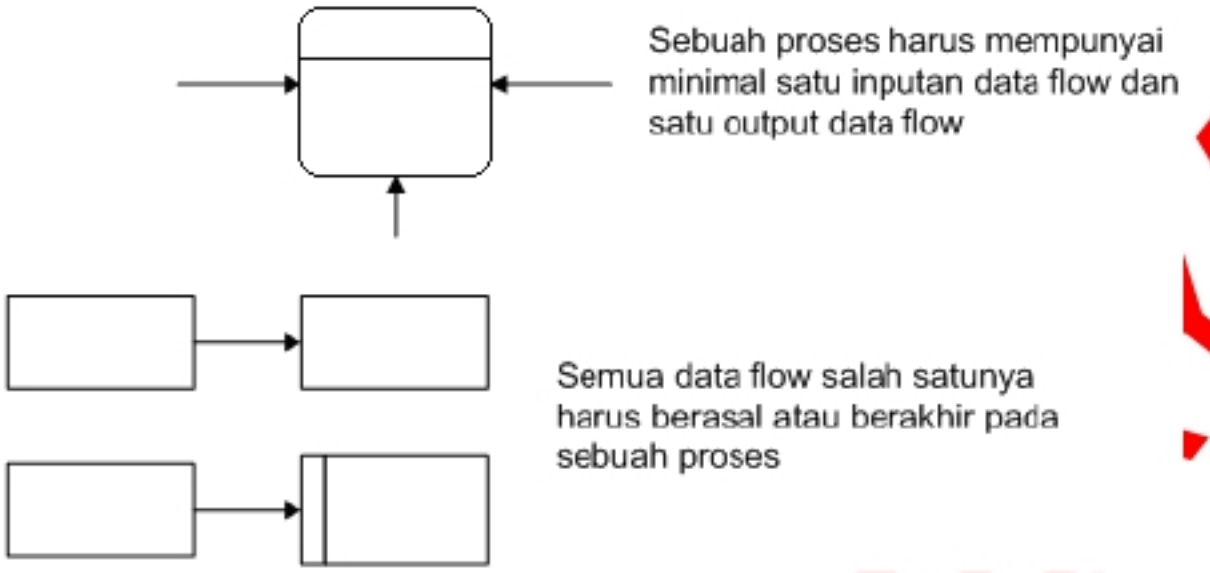


Gambar 2.7 Pembuatan *Child* Diagram

d. Pengecekan kesalahan-kesalahan pada diagram digunakan untuk melihat kesalahan-kesalahan yang terdapat pada sebuah DFD. Beberapa kesalahan-kesalahan yang umum terjadi ketika penggambaran/pembuatan DFD, ditunjukkan pada gambar 2.4 dan 2.5 berikut yaitu:



Gambar 2.8 Contoh 1 kesalahan penggambaran DFD

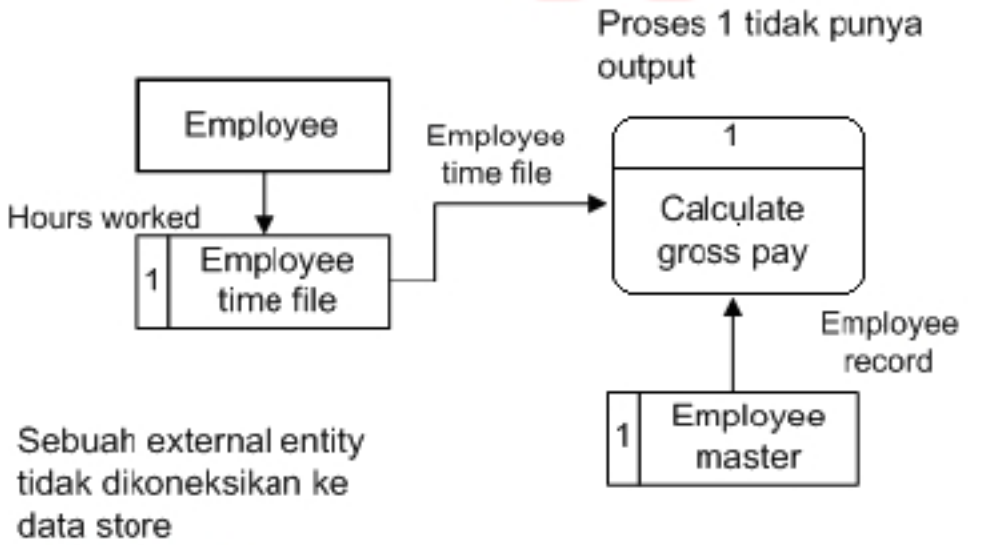


Gambar 2.9 Contoh 2 kesalahan penggambaran DFD

1. Lupa untuk meng*input*kan sebuah arus data atau arah panah langsung. Sebagai contoh adalah penggambaran proses yang menunjukkan sebuah data *flow* seperti *input* atau seperti *output*. Tiap-tiap proses pengubahan data harus menerima *input* dan *output*. Tipe kesalahan ini terjadi ketika sistem analis lupa memasukkan sebuah data *flow* atau meletakkan sebuah arah panah ditempat yang salah.
2. Hubungan penyimpanan data dan entity luar secara langsung satu sama lain. Data *store* dan entity tidak mungkin dikoneksikan satu sama lain, data *store* dan entity luar harus dikoneksikan melalui sebuah proses.
3. Kesalahan penamaan (label) pada proses-proses atau data *flow*. Pengecekan DFD untuk memastikan bahwa tiap-tiap objek atau data *flow* telah diberikan label. Sebuah proses haruslah di indikasikan seperti nama dari sistem atau menggunakan format kata kerja-kata benda. Tiap data *flow* haruslah dideskripsikan dengan sebuah kata benda.
4. Memasukkan lebih dari sembilan proses dalam sebuah DFD. Memiliki banyak proses akan mengakibatkan kekacauan pada diagram sehingga dapat

menyebabkan kebingungan dalam pembacaan sebuah proses dan akan menghalangi tingkat komunikasi. Jika lebih dari sembilan proses dalam sebuah sistem, maka beberapa grup dalam proses dilakukan bersama-sama ke dalam sebuah sub sistem dan meletakkannya dalam sebuah *child* diagram.

5. Menghilangkan suatu arus data. Pengujian dari suatu diagram yang menunjukkan garis / arah (*flow*), dimana untuk setiap proses data *flow* hanya mempunyai *input* data, *output* kecuali dalam kasus dari detil (*child*). Setiap *child* data dari DFD, arah arus data seringkali digambarkan untuk mengidentifikasikan bahwa diagram tersebut kehilangan data *flow*. Seperti ditunjukkan pada gambar 2.6 berikut:



Gambar 2.10 Contoh mengidentifikasikan diagram kehilangan data *flow*

6. Buat ketidaksesuaian komposisi dalam *child* diagram, dimana tiap *child* diagram harus mempunyai *input* dan *output* arus data yang sama seperti proses dilevel atasnya (parent proses). Pengecualian untuk rule ini adalah kurangnya *output*, seperti kesalahan garis yang ada didalam *child* diagram.

C. Perbedaan DFD (Logika dan Fisik)

Tabel 2.1. Perbedaan DFD (Logika dan Fisik)

Disain	Logika	Fisik
Gambaran model	Operasi-operasi bisnis	Bagaimana sistem akan diimplementasikan (atau bagaimana sistem dijalankan)
Apa yang ditampilkan oleh proses	Aktivitas bisnis	Program-program, modul program, dan prosedur-prosedur manual
Apa yang ditampilkan oleh data *store*	Koleksi-koleksi dari data yang dikesampingkan dari bagaimana data tersebut di simpan	File-file fisik dan database-database dari file manual
Kontrol sistem	Menunjukkan kontrol-kontrol bisnis	Menunjukkan kontrol-kontrol untuk validasi *input* data, untuk memperoleh sebuah *record*, untuk memastikan kesuksesan proses dan untuk keamanan sistem

2.6.3 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) digunakan untuk menginterpretasikan, menentukan dan mendokumentasikan kebutuhan untuk sistem pemrosesan database. ERD menyediakan bentuk untuk menunjukkan struktur keseluruhan kebutuhan data dari aplikasi. Adapun elemen-elemen dari ERD ini adalah :

1. *Entitas,* adalah sesuatu yang dapat diidentifikasikan di dalam lingkup pemakai, sesuatu yang penting bagi pemakai dari sistem yang akan dikembangkan.
2. *Atribut,* entitas memiliki atribut yang berfungsi untuk menjelaskan karakteristik dari entitas.

3. *Pengidentifikasi,* data-data entitas memiliki nama yang berfungsi untuk mengidentifikasikan mereka. Sebuah identifikasi dapat bersifat unik atau tidak unik.

Hubungan atau relasi berfungsi untuk menunjukkan hubungan satu entitas dengan entitas yang lain. Hubungan ini boleh memiliki atribut. Banyaknya entitas dalam suatu relasi menunjukkan tingkat dari relasi yang bersangkutan, namun yang banyak digunakan dalam aplikasi-aplikasi adalah model yang menggunakan relasi tingkat dua atau yang disebut dengan hubungan biner. Hubungan biner ini memiliki tiga tipe yaitu hubungan biner satu ke satu, hubungan biner satu ke banyak dan hubungan biner banyak ke banyak.