BAB II

TINJAUAN PUSTAKA

1.1 Profil Perusahaan

1.1.1 Logo Perusahaan

Logo adalah sebuah identitas perusahaan sebagai tanda / alamat yang akan mengkomunikasikan arti dan memberi makna terhadap jati diri perusahaan. Logo juga sebagai jiwa yang menghidupkan dan memberikan semangat bagi perusahaan di dalam melaksanakan misi dan mencapai visi perusahaan. Berikut ini adalah logo dari perusahaan PT. Tri Polyta Indonesia Tbk. :



Gambar 2.1. Logo Perusahaan

1.1.2 Sejarah Singkat Perusahaan

PT. Tripolyta Indonesia Tbk. adalah perusahaan perseroan terbatas yang didirikan berdasarkan akte notaris No. 40 tanggal 2 November 1984. akte notaris tersebut kemudian di ubah menjadi akte notaris No. 117 tanggal 17 November 1987. kedua akte tersebut telah mendapat persetujuan dari Menteri Kehakiman Republik Indonesia sesuai dengan SK No. C2-1786.HT.01.01- tahun 1988. Surat Keputusan tersebut ditetapkan pada tanggal 29 februari 1988 dan diumumkan dalam verita negara Republik Indonesia pada tanggal 5 Agustus 1988.

PT. Tripolyta Indonesia Tbk memperoleh fasilitas berdasarkan Undang – Undang Penanaman Modal Dalam Negeri dengan Surat Persetujuan Negeri PMDN tanggal 6 September 1985. Pabrik ini mulai

dikonstruksi pada tanggal 15 januari 1990 dan diresmikan pada tanggal 25 Mei 1992 oleh Presiden RI yang menjabat saat itu, Bapak Soeharto. Pabrik ini di bangun di atas tanah seluas 16,013 ha yang berlokasi di kawasan industri berat Cilegon. Kantor pusat perusahaan ini berada di Wisma Barito Pacific Tower lantai 9, Jakarta.

PT. Tripolyta Indonesia Tbk mengawali kegiatan produksinya pada bulan Mei 1992 yang bergerak dalam bidang Industri Petrokimia *(Intermediate)* dengan memproduksi *polypropylene* jenis *homopolymer* dan *kopolymer* acak. PT. Tri Polyta Indonesia Tbk pada awal berdirinya hanya memiliki dua unit rangkaian proses produksi (train) yang masing-masing berkapasitas 80.000 ton per tahun dan jenis *polypropylene* yang dihasilkan adalah *homopolymer* dan *kopolymer* acak (*random copolymer*).

Dalam perkembangannya terjadi peningkatan kemampuan produksi melalui proses modifikasi terhadap kapasitas produk *(debottlenecking) tahap I pada tahun 1994*, yakni masing-masing menjadi 120.000 ton per tahun. Jad, total produksi setelah proses modifikasi pada tahun 1994 tersebut adalah sebesar 240.000 ton per tahun.

Pada tahun 1994 di bangun train III dengan kapasitas 120.000 ton per tahun yang memproduksi *polypropylene* dengan jenis *kopolymer impak (impact copolymer*) dengan kapasitas 120.000 ton per tahun. Train III tersebut mulai beroperasi sejak tahun 1995. Saat ini, PT. Tripolyta Indonesia Tbk telah memiliki tiga buah train (rangkaian proses produksi) dan empat buah reaktor dengan total produksi 360.000 ton pertahun. Sehingga mampu menguasai lebih dari 50% pangsa pasar *polypropylene* di Indonesia.

PT. Tripolyta Indonesia Tbk memperoleh bahan baku pembuatan *polypropylene* yatu *propilene* dari PT. Chandra Asri Petrochemical Centre sebanyak 80 % dan sisanya merupakan impor dari negara Korea, USA, Singapura dan negara-negara yang ada di Timur Tengah. Perusahaan ini menggunakan teknologi yang dikenal dengan nama “UNIPOL Polypropylene Process” yang merupakan pengembangan dari “Union

Carbide Corporation USA" yang bekerja sama dengan Shell Company Corporation dalam penyediaan katalisnya.

PT. Tripolyta Indonesia Tbk merupakan salah satu perusahaan yang memproduksi *polypropylene* sebagai bahan baku plastik. PT. Tripolyta Tbk telah membantu menghemat devisa negara hingga US$ 170 Juta dengan memenuhi kebutuhan bahan baku plastik dalam negeri sehingga dapat mengurangi jumlah impor. Penanaman modal perusahaan ini tergolong penanaman modal dalam negeri. Investasi yang diperlukan untuk menjalankan pabrik ini adalah sebesar Rp 531 Milyar dengan modal tetap *(fixed capital)* sebesar Rp 476 Milyar dan modal kerja (*working capital*) sebesar Rp 55 Milyar. Untuk memperkuat struktur permodelan saham NASDAQ National Market Sistem dan pada tanggal 14 Maret 1995 melepaskan sahamnya di New York *Stock Exchange*. Penawaran saham di luar negeri diikuti dengan pencatatan di Bursa Efek Jakarta (BEJ) sejak 24 Juni 1996.

PT. Tri Polyta Indonesia telah mendapatkan sertifikat ISO 9002 danISO 14001. Perusahaan yang telah berhasil mendapatkan sertifikat ISO (International Organization of Standardization) adalah perusahaan yang berkualitas dan sistem organisasinya memperoleh pelayanan jasa atau produk yang terjamin dan dapat dipertanggungjawabkan mutunya. ISO 9002 sendiri merupakan modal jaminan kualitas (Quality Assurance) untuk produksi, instalasi, dan pelayanan. Sedangkan ISO 14001 membahas tentang Environmental Management Sistem , antara lain meliputi kondisi kebersihan lingkungan, keselamatan, dan sistem manajemen di lingkungan tersebut.

1.1.3 Visi dan Misi Perusahaan

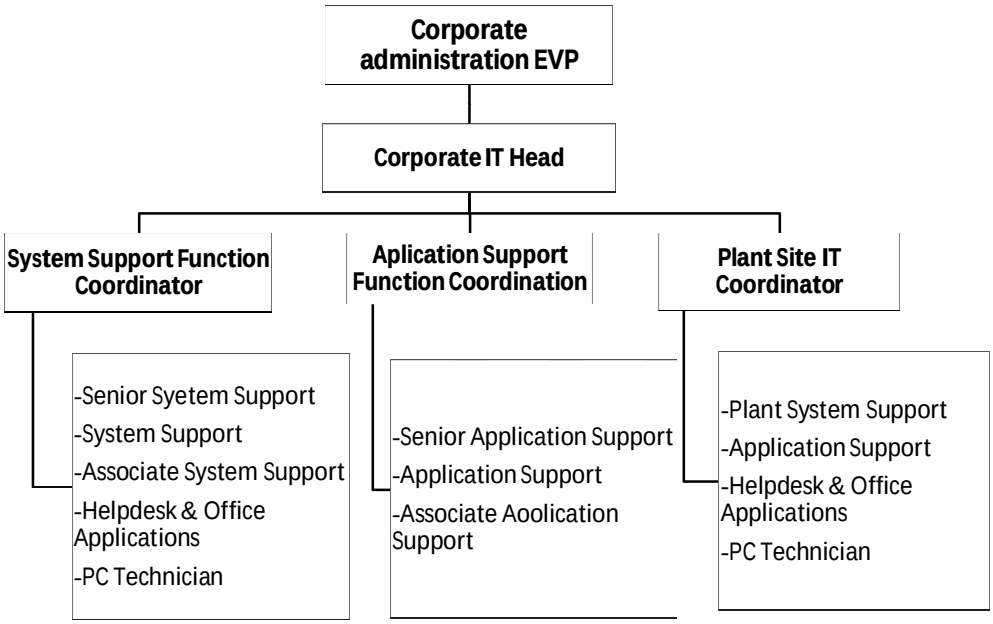
1.1.3.1 Visi

Menjadi pilihan utama pelanggan dan penyedia terbesar biji plastik polipropilena yang menjanjikan kesejahteraan masa depan pemegang saham, karyawan, dan pemangku kepentingan lainnya.

1.1.3.2 Misi

Kami menyediakan biji plastik polipropilena bermutu tinggi yang melampaui harapan pelanggan sehingga dapat memberikan manfaat maksimum kepada pemegang saham, karyawan, dan pemangku kepentingan lainnya.

1.1.4 Struktur Organisasi dan Job Desk



Gambar 2.2. Struktur Organisai *IT Department*

1.2 Landasan Teori

1.2.1 Basis Data (*Database*)

1.2.1.1 Pengertian Basis Data

Basis data dibayangkan sebagai sebuah lemari arsip. Jika kita memiliki sebuah lemari arsip dan berwenang untuk mengelolanya, maka kemungkinan besar kita akan melakukan hal – hal seperti : memberi map pada kumpulan – kumpulan arsip yang akan disimpan, menentukan kelompok atau jenis arsip, memberi penomoran dengan pola tertentu

yang nilainya unik pada setiap map, lalu menempatkan arsip – arsip tersebut dengan urutan tertentu di dalam lemari.

Basis data terdiri dari dua kata yaitu Basis yang dapat diartikan sebagai markas atau gudang, sedangkan Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi atau kombinasinya. Basis data sendiri dapat didefinisikan dalam sejumlah sudut pandang seperti :

a. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
b. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
c. Kumpulan file atau table atau arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.

Yang sangat ditonjolkan dalam basis data adalah pengaturan atau pemilihan atau pengelompokkan atau pengorganisasian data yang akan kita simpan sesuai fungsi ataupun jenisnya. Hal ini dapat berbentuk sejumlah file atau table terpisah atau dalam bentuk pendefinisian kolom-kolom data dalam setiap file atau table.

1.2.1.2 Sistem Basis Data

Sistem adalah sebuah tatanan (keterpaduan) yang terdiri atas sejumlah komponen fungsional (dengan satuan fungsi / tugas khusus) yang saling berhubungan dan secara bersama – sama bertujuan untuk memenuhi suatu proses atau pekerjaan tertentu. Basis data hanyalah sebuah objek yang pasif atau mati. Ia ada karena ada pembuatnya. Ia tidak akan pernah berguna jika tidak ada pengelola atau penggeraknya. Yang menjadi penggeraknya secara langsung adalah program atau

aplikasi (*software*). Gabungan keduanya menghasilakn sebuah sistem. Karena itu, secara umum sebuah sistem basis data merupakan sistem yang terdiri atas kumpulan file (table) yang saling berhubungan (dalam sebuah basis data di sebuah sistem komputer) dan sekumpulan program (DBMS) yang memungkinkan beberapa pemakai dan atau program lain untuk mengakses dan memanipulasi file – file (table - tabel) tersebut.

Lebih jauh lagi, dalam sebuah sistem basis data, secara lengkap akan terdapat komponen – komponen utama sebagai berikut :

a. Perangkat keras (*Hardware*)

Perangkat keras yang biasanya terdapat dalam sebuah sistem basis data adalah :

1. Komputer (satu untuk sistem yang *stand – alone* atau lebih dari satu untuk sistem jaringan).
2. Memori sekunder yang *on – line* (*harddisk*).
3. Memori sekunder yang *off – line* (*Tape* atau *Removable Disk*) untuk keperluan *backup* data.
4. Media atau perangkat komunikasi (untuk sistem jaringan).

b. Sistem Operasi (*Operating System*)

Secara sederhana, sistem operasi merupakan program yang mengaktifkan atau memfungsikan sistem komputer, mengendalikan seluruh sumber daya (*resource*) dalam komputer dan melakukan operasi – operasi dasar dalam computer (operasi I/O, pengelolaan file, dll). Program pengelola basis data hanya dapat aktif (*running*) jika sistem operasi yang dikehendakinya atau sesuai telah aktif.

c. Basis Data (*Database*)

Sebuah sistem basis data dapat memiliki beberapa basis data. Setiap basis data dapat berisi sejumlah objek basis data seperti *file / table*, indeks, dll. Di samping berisi atau menyimpan data, setiap basis data juga mengandung definisi struktur (baik untuk basis data maupun objek – objeknya secara detail).

d. Sistem (Aplikasi / perangkat lunak) Pengelola Basis Data (DBMS)

Pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah perangkat lunak yang khusus. Perangkat lunak ini disebut DBMS (*Database Management Sistem*) yang yang akan menentukan bagaimana data diorganisasi, disimpan, diubah dan diambil kembali. Ia juga menerapkan mekanisme pengamanan data, pemakaian data secara bersama, pemaksaan keakuratan atau konsistensi data, dan sebagainya.

e. Pemakai (*User*)

Ada beberapa jenis pemakai terhadap suatu sistem basis data yang dibedakan berdasarkan cara mereka berinteraksi terhadap sistem :

1. Programmer Aplikasi

   Pemakai yang berinteraksi dengan basis data melalui *Data Manipulation Language* (DML), yang disertakan (*embedded*) dalam program yang ditulis dalam bahasa pemrograman induk (seperti C, Pascal, Cobol, dan lain lain).

2. User Mahir (*Casual User*)

   Pemakai yang berinteraksi dengan sistem tanpa menulis modul program. Mereka menyatakan *query* (untuk akses data) dengan bahasa *query* yang disediakan oleh suatu DBMS.

3. User Umum (*End User / Naïve User*)

   Pemakai yang berinteraksi dengan sistem basis data melalui pemanggilan satu program aplikasi permanen (*executable program*) yang telah ditulis atau disediakan sebelumnya.

4. User Khusus (*Specialized User*)

   Pemakai yang menulis aplikasi basis data non-konvensional, tetapi untuk keperluan – keperluan khusus,

seperti untuk aplikasi AI, Sistem Pakar, Pengolahan Citra, dll, yang bisa saja mengakses basis data dengan / tanpa DBMS yang bersangkutan.

f. Aplikasi (*perangkat lunak*) lain (bedrsifat opsional)

Aplikasi (perangkat lunak) lain ini bersifat opsional. Artinya, ada atau tidaknya tergantung pada kebutuhan kita. DBMS yang kita gunakan lebih berperan salam pengorganisasian data dalam basis data, sementara bagi pemakai basis data (khususnya yang menjadi *end – user* / *naive - user*) dapat dibutuhkan / disediakan program khusus / lain untuk melakukan pengisian, pengubahan dan pengambilan data. Pemrogram ini ada yang sudah disediakan bersama dengan DBMS – nya, ada juga yang harus dibuat sendiri dengan menggunakan aplikasi lain yang khusus untuk itu (*delevopment tools*).

1.2.1.3 Model Data

Model data sendiri dapat didefinisikan sebagai kumpulan perangkat konseptual untuk menggambarkan data, hubungan data, semantic (makna) data dan batasan data. Oleh karena yang ingin ditunjukkan adalah makna dari data dan keterhubungannya dengan data lain, maka Model Data ini lebih tepat jika disebut Model Data Logik. Penyingkatannya menjadi Model Data dilakukan demi penyederhanaan penulisan.

Ada sejumlah secara dalam meprestasikan Model Data dalam perancangan basis data, yang secara umum dapat dibagi dalam 2 (dua) kelompok yaitu, yaitu :

a. Model Logik Data Berdasarkan Objek (*Objek – Based Logical Models*) yang terdiri dari :
    1. *Entity – Relationship Model*
    2. *Object – Oriented Model*
    3. *Semantic Data model*

4. *Functional Data model*

b. Model Lojik Data Berdasarkan *Record* (*Record – Based Locical* Models) yang terdiri dari :
   1. *Relational Model*
   2. *Hierarchical Model*
   3. *Network Model*

1.2.1.3.1 Model *Entity – Relationship* (Model Keterhubungan - Entitas)

Pada model Entity – Relationship, semesta data yang ada di 'Dunia Nyata' diterjemahkan dengan memanfaatkan sejumlah perangkat konseptual menjadi sebuah diagram data, yang umum disebut sebagai Diagram *Entity – Relationship* (Diagram E – R). sebelum kita mebahas lebih jauh tentang bagaimana Diagram E- R tersebut dapat kita gambarkan, maka yang harus lebih dulu diketahui adalah komponen – komponen pembentukan Model Entity – relationship. Sesuai namanya, ada dua komponen utama pembentuk Model Entity –Relationship, yaitu Entitas (*Entity*) dan Relasi (*Relation*). Kedua komponen ini di deskripsikan lebih jauh melalui sejumlah *Atribut* atau *Properti*.

1.2.1.3.1.1 Entitas (*Entity*) dan Himpunan Entitas (*Entitas Sets*)

Entitas merupakan invidu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain. Contoh – contoh himpunan entitas adalah :

a. Semua pelanggan atau pelanggan saja. Dengan entitas Budiman, Aminah dan seterusnya.
b. Semua mobil atau mobil saja. Dengan emtitas mobil Suzuki, mobil Toyota, mobil Honda dll.
c. Semua mahasiswa atau mahasiswa saja. Dengan entitas Ali, Budi, Iman dan seterusnya.

1.2.1.3.1.2 Atribut (*Attributes* / *Properties*)

Setiap Entitas pasti memiliki Atribut yang mendeskripsikan karakteristik (*properti*) dari Entitas tersebut. Sebagaimana telah disebutkan sebelumnya, penentuan / pemilihan atribut – atribut yang relevan bagi sebuah entitas meru[akan hal penting lainnya dalam pembentukan model data. Penetapan atribut bagi sebuah entitas umunya memang didasarkan pada fakta yang ada.

Yang relevan untuk lebih diperhatikan dalam pembuatan model E-R adalah kedudukan atribut dalam entitas. Harus dapat kita bedakan / ketahui, mana atribut yang berfungsi sebagai *Primary Key* dan mana yang bukan (atribut deskriptif).

1.2.1.3.1.3 Relasi (*Relationship*)

Relasi menunjukkan adanya hubungan di antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Misalnya, entitas seorang mahasiswa dengan nim = ‘980001’ dan *nama_mhs*=’Ali Akbar’ (yang ada di himpunan entitas Mahasiswa) mempunyai relasi dengan entitas sebuah mata kuliah dengan *kode_kul*=’IF-110’ dan *nama_kul*=’Struktur Data’. Relasi diantara kedua entitas tadi mengandung arti bahwa mahasiswa tersebut sedang mengambil atau mempelajari mata kuliah tersebut di sebuah perguruan tinggi yang kita tinjau. Kumpulan semua relasi di antara entitas – entitas yang terdapat pada himpunan entitas – himpunan entitas tersebut memebentuk Himpunan Relasi. Istilah Himpunan Relasi jarang sekali digunakan dan lebih sering disingkat dengan istilah Relasi.

1.2.1.3.1.4 Kardinalitas / Derajat Relasi

Kardinalitas relasi menunjukkan jumlah maksimum entitas yang dapat berlasi dengan entitas pada himpunan entitas yang lain. Sebagai contoh, dapat kita lihat bahwa entitas – entitas pada himpunan

entitas Mahasiswa dapat berelasi dengan satu entitas, banyak entitas atau bahkan tidak satupun entitas dari himpunan entitas Kuliah. Begitu juga sebaliknya, entitas – entitas pada himpunan entitas Kuliah ada yang berelasi dengan beberapa entitas pada himpunan entitas Mahasiswa dan ada pula yang berelasi dengan satu entitas pada himpunan entitas Mahasiswa. Dengan demikian Kardinalitas Relasi yang terjadi diantara dua himpunan entitas :

a. Satu ke Satu

   Setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, dan begitu juga sebaliknya.

b. Satu ke Banyak

   Setiap entitas pada himpunan entitas A dapat berhubungan banyak dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya, di mana setiap entitas pada himpunan entitas berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.

c. Banyak ke Satu

   Entitas pada himpunan A berhubungan dengan paling banyak dengan satu jenis entitas pada himpunan entitas B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas pada himpunan entitas B.

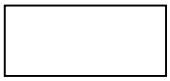
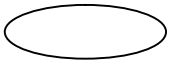
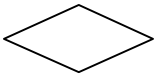
d. Banyak ke Banyak

   Pada himpunan A dapat berhubungan dengan banyak entitas pada himpunan entitas B, dan demikian juga sebaliknya, dimana setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A.

1.2.1.3.2 Diagram *Entity – Relationship* (ERD).

Model *Entity – Relational* yang berisi komponen-komponen Himpunan Entitas dan Himpunan Relasi yang masing – masing dilengkapi dengan atribut – atribut yang mempresentasikan seluruh fakta dari '*dunia nyata'* yang di tinjau, dapat digambarkan dengan lebih sistematis dengan menggunakan ERD (*Entity Relationship Diagram*). Notasi – notasi simbolik di dalam ERD yang dapat digunakan adalah :

Tabel 2.1. Notasi – Notasi Simbolik ERD (Entity Relationship Diagram)

	Persegi panjang, ,menyatakan Himpunan Entitas.
	Elips menyatakan Atribut (Atribut yang berfungsi sebagai *key* digarisbawahi).
	Belah ketupat menyatakan Himpun Relasi.
	Garis sebagai penghubung antara Himpunan Relasi dengan Himpunana Entitas dan Himpunan Entitas dengan atributnya.
a. 1 dan 1 untuk relasi satu-ke-satu. b. 1 ke N untuk relasi satu-ke-banyak. c. N ke N untuk relasi banyak-ke-banyak.	Kardinalitas Relasi dapat dinyatakan dengan banyaknya garis cabang atau dengan pemakaian angka.

1.2.1.3.3 Varian Entitas dan Relasi

Idealnya himpunan entitas yang terlibat dalam sebuah ERD adalah himpunan entitas kuat / bebas. Himpunan entitas demikian tidak memiliki ketergantungan dengan himpunan entitas lainnya. Namun demikian, dalam pembentukan ERD kita tidak selalu dapat melibatkan himpunan entitas seperti itu. Ada kalanya kita juga melibatkan himpunan entitas yang lemah atau merupakan bagian dari himpunan entitas lainnya.

1.2.1.3.3.1 Himpunan Entitas Lemah (*Weak Entity Sets*)

Berisi entitas – entitas yang kemunculannya tergantung pada eksistensinya dalam sebuah relasi terhadap entitas lain. Himpunan entitas demikian biasanya tidak memiliki atribut yang dapat berfungsi sebagai *key*, yang dapat menjamin keunikan entitas di dalamnya.

1.2.1.3.3.2 Sub Entitas (*Subtype Entities*)

Merupakan himpunan entitas yang beranggotakan entitas – entitas yang merupakan bagian dari himpunan entitas yang lebih *superior* (utama). Sub Entitas ini merupakan hasil dekomposisi (spesialisasi) himpunan entitas berdasarkan pengelompokan tertentu.

1.2.1.4 Arsitektur Sistem

Arsitektur sistem merujuk pada konfigurasi sistem secara keseluruhan yang akan menjadi “tempat hidup” dari DBMS, basis data dan aplikasi yang memanfaatkannya. Sudah seharusnya ditetapkan sejak awal sebelum memulai perancangan basis data, atau paling tidak sebelum melaksanakan tahap implementasi basis data.

Beberapa jenis arsitektur sistem yang dapat digunakan dan akan diuraikan pada bab ini adalah :

a. Sistem Tunggal (*Stand - Alone*).
b. Sistem Tersentralisasi (*Centralized Sistem*).
c. Sistem *Client - Server*

1.2.1.4.1 Sistem Tunggal / Mandiri (*Stand-Alone*)

Pada arsitektur ini, DBMS, basis data dan aplikasi basis data ditempatkan pada computer yang sama dengan demikian, pemakai yang dapat menggunakannya disetiap saat juga hanya satu orang (*single user*). Arsitektur ini merupakan arsitektur sistem yang paling sederhana dan paling murah dan dapat menjadi sebuah pilihan jika basis data yang dikelola tidak terlalu besar dan lebih bersifat membantu mempercepat pekerjaan administrative.

1.2.1.4.2 Sistem Tersentralisasi (*Centralized Sistem*)

Arsitektur ini terdiri atas sebuah mesin server dan sejumlah terminal (yang menjadi tempat *user* berinterkasi dengan sistem). Yang tersentralisasi dalam arsitektur ini dapat mencakup basis data, DBMS dan aplikasi basis data atau basis data saja. Untuk lingkup sentralisasi yang pertama, maka jenis *server*-nya sering disebut sebagai DBMS atau server aplikasi dan terminalnya lebih tepat disebut *dumb-terminal* (terminal pasif). Sedangkan jika yang disentralisasi hanya basis data, server yang digunakan bias disebut *file-server* dan terminalnya disebut *work station*.

1.2.1.4.3 Sistem *Client Server*

Sebagai sistem tersentralisasi, arsitektur ketiga ini juga diterapkan pada sebuah sistem jaringan. Sistem *Client – Server* ini ditujukan untuk mengatasi kelemahan – kemelamahan yang terdapat pada sistem tersentralisasi sebelumnya. Kelemahan pada bentuk sistem tersentralisasi yang pertama, yaitu beratnya beban server yang harus menangani semua proses, diatasi dengan membagi beban itu menjadi dua bagian, *client* (yang menjalankan aplikasi basis data) dan *server* (yang menjalankan DBMS dan berisi basis data) pada mesin yang berbeda. Sedang kelemahan pada bentuk sistem tersetralisasi yang

kedua yaitu, padatnya lalu lintas data antara server dan *work-station* diatasi dengan mekanisme transfer data yang lebih efisien.

Sistem ini terdiri atas dua komponen utama, yaitu *client* dan *server*. *Client* berisi aplikasi basis data dan *server* berisi DBMS dan basis data. Setiap aktivitas yang dikehendaki para pemakai akan lebih dulu didatangi oleh *client*. *Client* selanjutnya mengupayakan agar semua proses sedapat mungkin ditangani sendiri. Jika ada proses yang harus melibatkan data yang tersimpan pada basis data, barulah *client* melakukan "kontak" dengan *server*.

1.2.1.5 DBMS (*Database Management Sistem*)

DBMS umunya menyediakan program khusus (*utilitas / utility*) yang dapat digunakan secara interaktif untuk melakukan berbagai operasi terhadap basis data, seperti pembuatan *table*, penghapusan *table*, penambahan data, pengubahan data, pencarian data, penghapusan data, dll. Namun disamping adanya program khusus itu, DBMS juga umumnya menyediakan sekumpulan perintah dalam bentuk *command line*, yakni perintah yang dituliskan pemakai untuk maksud yang sama. Perintah – perintah ini dapat diberikan dan dikerjakan oleh DBMS melalui utilitas lain yang juga disediakan DBMS atau melalui program / aplikasi yang dibuat sendiri oleh pemakai. Kumpulan perintah ini dapat disebut sebagai Bahasa Basis Data (*Database Language*).

Karena DBMS yang digunakan untuk mengelola basis data dan *development tools* yang digunakan untuk menulis aplikasi basis data belum tentu dibuat berasal dari perusahaan pembuat perangkat lunak (*vendor*) yang sama, maka diperlukan sebuah bahasa basis data yang sifatnya standar. Dengan bahasa yang standar, kita dapat menggunakan bahasa tersebut untuk mengakses data dimanapun tanpa memperdulikan DBMS yang digunakan. Atau bahkan sebaliknya data yang dikelola sebuah DBMS dapat diakses dari manapun, tidak peduli *development*

tools yang digunakan. Dengan begitu terdapat independensi antara DBMS dan *development tools*.

1.2.2 Data Flow Diagram

DFD (*Data Flow Diagram*) adalah suatu model logika data atau proses yang dibuat untuk menggambarkan dari mana asal data dan dan kemana tujuan data yang akan keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut dan interaksi antara data yang tersimapan dan proses yang dikenakan pada data tersebut.

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir atau dimana data tersebut akan disimpan.

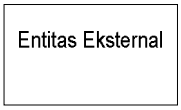
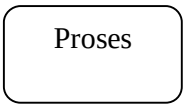
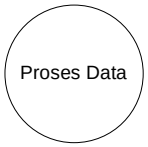
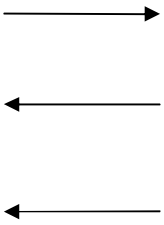
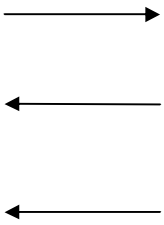
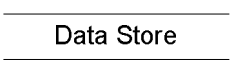
DFD merupakan alat yang digunakan pada metodologi pengembangan sistem yang terstruktur. Diagram yang menunjukkan bagaimana data berpindah / berjalan didalam sistem Informasi yang akan dikembangkan.

Data flow diagram dapat menunjukkan ringkasan dari sistem yang luas / besar dari sistem input, proses dan output. Adapun keuntungan menggunakan DFD, yaitu :

1. Bebas menjalankan implementasi teknis sistem.
2. Komunikasi pengetahuan sistem yang ada dengan *user*.
3. Menganalisis sistem yang diajukan untuk menentukan ketepatan data – data dan proses yang dibutuhkan.

Dalam pembuatan DFD ini memiliki beberapa peraturan yang nantinya dapat mempermudah proses pembuatannya. Selain itu ada simbolik dalam DFD yang dapat digunakan, sebagai berikut :

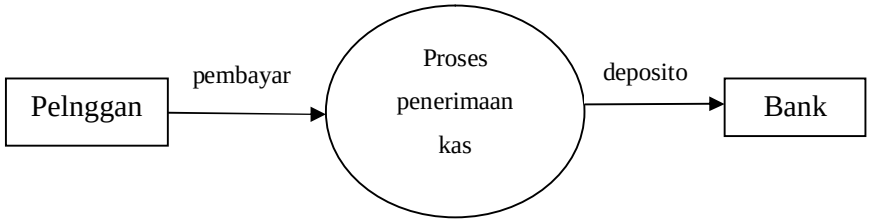
Tabel 2.2. Simbolik dalam DFD (*Data Flow Diagram*)

Gane/Sarson	Yourdon/De Marco	Keterangan
		a. Sumber / tujuan aliran data dari / ke sistem. b. Entitas eksternal, dapat berupa orang / unit terkait yang berinteraksi dengan sistem tetapi diluar sistem
		a. Fungsi dijelaskan oleh nama yang dicantumkan dalam lingkaran. b. Nama proses : Kata kerja + Kata benda. c. Orang, unit yang mempergunakan atau melakukan transformasi data. Komponen fisik tidak didefinisikan.
Aliran data 	Aliran data 	a. Mengidentifikasikan satu aliran data pada waktu yang tidak dispesifikasikan. b. Aliran data dengan arah khusus dari sumber ke tujuan. c. Nama data dicantumkan pada garis.
		a. Menyatakan arsip meja, lemari, *tape*, disk atau media lainnya. b. Nama data / arsip diletakkan di dalam symbol.

		c. Penyimpanan data atau tempat data direfer oleh proses.

1.2.2.1 Type DFD (*Data Flow Diagram*)

DFD terdiri dari tiga tipe, *Context diagram, DFD Physical* dan *DFD Logical. Context diagram* merupakan diagram tingkat atas, yaitu diagram yang paling tidak detail, dari sebuah sistem informasi yang menggambarkan aliran – aliran data ke dalam dank keluar sistem.



Gambar 2.3. Diagram Konteks

DFD *Physical* merupakan representasi grafik dari sebuah sistem yang menunjukkan entitas – entitas internal dan eksternal dari sistem tersebut, dan aliran – aliran ke dalam dan keluar dari entitas – entitas tersebut. Entitas – entitas personal adalah personel, tempat (misal sebuah bagian organisasi), atau mesin (misalnya, komputer) dalam sistem tersebut yang mentransformasi data. Jadi dapat dikatakan tidak menunjukkan apa yang dilakukan, tetapi menunjukkan dimana, bagaimana dan oleh siapa proses – proses dalam sebuah sistem dilakukan.

DFD *Logical* merupakan grafik dari sebuah sistem yang menunjukkan proses – proses dalam sistem tersebut dan aliran – aliran data ke dalam dan keluar dari proses – proses tersebut. Kita menggunakan dokumentasi DFD *Logical* untuk membuat dokumentasi sebuah sistem informasi. Karena DFD *Logical* dapat mewakili logika sistem tersebut, yaitu apa yang dilakukan oleh sistem tersebut, tanpa perlu menspesifikasi dimana, bagaimana dan oleh siapa proses – proses dalam sistem tersebut

dilakukan. Dengan demikian, DFD *Logical* menggambarkan kegiatan – kegiatan yang dilakukan oleh sebuah sistem, sedangkan DFD *Physical* menggambarkan infrastruktur yang dimiliki oleh sebuah sistem.

DFD memiliki dekomposisi atau *Levelling* DFD yang jumlah proses yang ter-*identifikasi* dan harus digambarkan dalam DFD, mungkin saja berjumlah ratusan. Apapun cara terbaik untuk mengorganisasikan gambar DFD adalah dengan mem-*partisi* DFD secara *top-down*. Dalam penggambaran symbol DFD, terdapat beberapa peraturan yang harus diperhatikan sehingga dalam penggambarannya tidak terjadi kesalahan. Peraturan – peraturan yang ada adalah sebagai berikut :

1. Antar entity luar tidak diizinkan terjadi hubungan atau relasi.
2. Tidak boleh ada aliran data antara entitas luar dengan data *store*.
3. Tidak boleh ada aliran data antara data *store.*

1.2.3 Spesifikasi Kebutuhan Perangkat Lunak

Model analisis adalah teknik pertama untuk mempresentasikan sistem. Pemodelan analisis menggunakan kombinasi dari text dan diagram untuk mepresentasikan kebutuhan perangkat lunak (data, fungsi, dan tingkah) dengan cara yang dimengerti. Ada dua tipe pemodelan analisis yang digunakan, yaitu analisis terstruktur dan analisis berorientasi objek.

Model data dimodelkan menggunakan diagram E – R untuk menggambarkan objek data, atribut dan relasi. Model fungsional dimodelkan menggunakan DFD untuk menggambarkan bagaimana data ditransformasikan kedalam sistem, model tingkah laku (*Behavioral modeling*) dimodelkan menggunakan *state transition diagrams* (STD) untuk menggambarkan tingkah laku kejadian diluar sistem. Analisis kerja suatu produk harus dikaji dalam hal *completeness*, *correctness* dan *consistency*.

Analisis juga memiliki tiga model analisis, yaitu :

a. Analisis Terstruktur (DeMarco)

Analisis produk harus sangat dipertahankan, terutama persyaratan spesifikasi perangkat lunak. Masalah ukuran harus ditangani dengan menggunakan metode efektif partisi. Sedangkan grafis harus digunakan bila memungkinkan. Membedakan antara pertimbangan logis (penting) dan fisik (implementasi), selain itu membantu menemukan sesuatu dengan persyaratan partisi dan partisi dokumen sebelum spesifikasi. Menemukan cara untuk melacak dan mengevaluasi *user interface*. Merancang alat yang menggambarkan logika dan kebijakan yang lebih baik dari teks naratif.

b. Analisis Model Objektif

Menggambarkan apa yang *customer* butuhkan sehingga dapat dijadikan bahan dasar untuk proses *design*, selain itu dapat merencanakan sekumpulan kebutuhan yang dapat dijadikan bahan validasi pada saat perangkat lunak dibangun.

c. Analisis Model Elemen

Ada beberapa elemen – elemen yang mendukung, yaitu :

1. Kamus data : berisi deskripsi dari semua objek data yang dibutuhkan atau diproduksi oleh perangkat lunak.
2. *Entity Relationship Diagram* (ERD) : menggambarkan hubungan antara objek data.
3. *Data Flow Diagram* (DFD) : memberikan indikasi tentang bagaimana data berubah karena mereka bergerak melalui sistem. Juga menggambarkan fungsi yang mentrtansformasikan aliran data (fungsi dipresentasikan dalam DFD menggunakan spesifikasi proses atau PSPEC).

1.2.4 Desain

Desain perangkat lunak adalah suatu proses interaktif yang melalui persyaratan diterjemahkan ke dalam "*Blue Print*" untuk membangun suatu perangkat lunak. Sepanjang proses desain, kualitas desain yang

melengkapi dinilai dengan serangkaian kajian teknis formal atau desain awal. Pedoman bagi evaluasi suatu desain yang baik adalah :

a. Desain harus mengimplementasi keseluruhan persyaratan *eksplisist* yang dibebankan dalam model analisis, dan harus mengakomodasi semua persyaratan implisist yang diinginkan pelanggan.
b. Desain harus menjadi panduan yang dapat dibaca.
c. Desain harus memberikan suatu gambaran lengkap mengenai perangkat lunak.

1.2.5 Pengujian Black Box

Testing merupakan sebuah proses mengeksekusi program dengan maksud untuk menemukan kesalahan. Sebuah kasus testingyang baik adalah kasus yang mempunyai kemungkinan besar menemukan kesalahan yang belum ditemukan. Sebuah test yang sukses adalah test yang menemukan kesalahan yang belum ditemukan.

Mengetahui fungsi yang dirancang dapat dilakukan oleh produk, test dapat dilakukan untuk mendemostrasikan bahwa setiap fungsi dapat berjalan dengan baik, dan pada yang bersamaan mencari kesalahan dalam setiap fungsi *àblack-boxtesting*.

Untuk mengetahui isi dari produk, test dapat dilakukan untuk memastikan bahwa semua operasi dapat berjalan sesuai spesifikasi dan semua komponen telah dicoba *àwhite-boxtesting*.

Menghasilkan himpunan kondisi input yang akan mencoba semua kebutuhan fungsional sebuah program. Mencoba menemukan kesalahan dalam kategori :

1. Fungsi yang tidak benar atau hilang
2. Kesalahan antarmuka
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan kinerja
5. Kesalahan inisialisasi dan terminasi

Test case ini bertujuan untuk menunjukkan fungsi perangkat lunak tentang cara beroperasinya, apakah pemasukan data keluaran telah berjalan sebagaimana yang diharapkan dan apakah informasi yang disimpan secara eksternal selalu dijaga kemutakhirannya.

Tehnik pengujian *black-box* berfokus pada domain informasi dari perangkat lunak, dengan melakukan test case dengan menpartisi domain input dari suatu program dengan cara yang memberikan cakupan pengujian yang mendalam.

Metode pengujian graph-based mengeksplorasi hubungan antara dan tingkah laku objek-objek program. Partisi ekivalensi membagi domain input ke dalam kelas data yang mungkin untuk melakukan fungsi perangkat lunak tertentu. Analisis nilai batas memeriksaa kemampuan program untuk menangani data pada batas yang dapat diterima.

Metode pengujian yang terspesialisasi meliputi sejumlah luas kemampuan perangkat lunak dan area aplikasi. GUI, arsitektur client/ server, dokumentasi dan fasilitas help dan sistem real time masing-masing membutuhkan pedoman dan tehnik khusus untuk pengujian perangkat lunak

1.2.6 Delphi

Delphi merupakan salah satu bahasa pemrograman tingkat tinggi berbasis *Windows.* Delphi digolongkan ke dalam bahasa pemrograman visual yang menitik beratkan pada pemrograman berorientasi objek (*object oriented programming*). Membuat program menggunakan Delphi tidaklah terlalu sulit karena bahasa pemrograman ini dikembangkan menggunakan bahasa Pascal.

Struktur program Delphi, dibangun berdasarkan struktur program Pascal yang menitik beratkan pada struktur pemrograman dengan berorientasi pada objek. Objek dalam Pascal ditulis ke dalam sebuah *file* yang disebut dengan *unit.* Unit inilah yang nantinya akan digunakan sebagai acuan dalam pembuatan program menggunakan Pascal. Unit

dalam pascal berupa kode – kode program, sedangkan unit dalam Delphi digunakan untuk menyimpan pendeklarasian form dalam bentuk *class*. *Class* yang di deklarasikan dalam unit, oleh Delphi akan diterjemahkan sebagai sebuah form dan objek lain yang digunakan dari *component palette*. Setiap form yang digunakan dalam Delphi pasti akan dideklarasikan dalam unit secara otomatis oleh software ini.

Dalam sebuah pembuatan sebuah program aplikasi, Delphi biasa terdiri dari beberapa file. Apapun bentuk file komponen yang digunakan dalam pembuatan program dengan Delphi, pasti akan disimpan dalam sebuah file dalam bentuk unit. File – file unit tersebut disatukan dengan kode – kode program yang dibangun ke dalam satu file yang disebut dengan *project*.

1.2.6.1 Database Desktop Paradox

Database desktop merupakan suatu program "*Add-Ins*", yaitu program terpisah yang langsung terdapat pada Borland Delphi. Pada database desktop terdapat beberapa DBMS yang terintegrasi di dalamnya, antara lain; Paradox 7, Paradox 4, Visual dBase, Foxpro, Ms. SQL, db2, dan Interbase. Dari beberapa DBMS tersebut kita akan memilih salah satu yaitu Paradox 7. Dalam Paradox 7 ini, pada satu *file* database hanya mengizinkan satu tabel, berbeda dengan DBMS lain yang mengizinkan beberapa tabel pada satu *file* database seperti Ms. Acces. Struktur *field* pada Paradox 7 adalah :

1. Field Name

   Field Name merupakan nama pengenalan kolom pada suatu table. Terdapat beberapa aturan dalam penulisan field name antara lain :

   a. Panjang maksimum 25 karakter.
   b. Tidak boleh diawali dengan spasi tapi boleh mengandung spasi.
   c. Unik, artinya tidak ada nama kolom yang sama.
   d. Tidak boleh menggunakan tanda koma (,), tanpa pipe (|), dan tanda seru (!).

e. Hindari kata – kata yang merupakan perintah SQL.

2. Type

Digunakan untuk menentukan tipe data yang dapat ditampung dalam *field*. Macam – macam tipe data yang sering di pakai dalam Paradox adalah sebagai berikut :

Tabel 2.3. Macam – Macam Tipe Data

Symbol *Type Field*	Jenis *Field*	Keterangan
A	Alpha Numeric	String, Alphabet, Numeric
I	Integer	Integer (Bilangan Bulat)
N	Number	Pecahan
D	Date	Tanggal
T	Time	Jam
M	Memo	Memo / Catatan panjang
G	Grapich	Gambar
+	Autoincrement	Otomatis naik setiap ada penambahan data
$	Money	Mata uang
S	Shortint	Integer kecil (-32768 s.d 32767)
@	Time and Date	Tanggal dan waktu

3. Size

Merupakan ukuran dari panjang data yang diizinkan untuk field.

4. Key

Dapat berupa primary key sebagai key yang menbedakan untuk setiap baris kolom. Syarat untuk primary key adalah unik, artinyaa tidak boleh ada data yang memiliki primary key yang sama.