BAB II

TINJAUAN PUSTAKA

2.1. Profil Perusahaan

2.1.1. Sejarah Perusahaan

PT Shafira Lintas Semesta (*Shafira Tour & Travel*) adalah satu dari beberapa perusahaan yang bergerak dibidang pelayanan usaha Perjalanan Wisata di Jawa Timur, yang didirikan pada tahun 2001 oleh Bapak Drs. H. Mohammad Ansor Alamsyah, MM. dengan akte pendirian notaris Untung Darnoesoewirjo, S.H. Akta No. 45. Shafira juga memperoleh ijin pariwisata No. 503/44.1/436.5.12/2007.

Usaha jasa yang dikembangkan Shafira *Tour & Travel* antara lain :

1. Umroh dan Haji.
2. *Domestic dan International ticket.*
3. *Domestic inbound tour.*
4. *International outbound tour.*
5. *Hotel reservation.*
6. *Travel document.*
7. *Rent car.*
8. *Money changer.*

Shafira Tour & Travel makin berkembang pesat dengan layanan Umrah & Haji *Plus*. Shafira *Tour & Travel* memperoleh ijin Umroh : D/563 tahun 2005.

Kantor utama Shafira berada di Gedung Graha Pena Jawa Pos lantai 2, Jalan Ahmad Yani 88 Surabaya, telepon : +6231-8273300, sedangkan kantor operasional berada di Graha Shafira Juanda *Business Center* (JBC) B-4, Jalan Juanda No. 1 Aloha Surabaya, telepon : +6231-8555558. Shafira *Tour & Travel* memiliki sebuah anak perusahaan yang dinamakan PT Kopindo yang berada di Jakarta. PT kopindo telah mendapat ijin haji dari Kementrian Agama Republik Indonesia, sehingga para jamaah haji yang menggunakan jasa Shafira *Tour & Travel* berada di bawah bendera PT Kopindo. Shafira *Tour & Travel* ini juga tercatat sebagai Asosiasi Muslim Penyelenggara Umrah & Haji (AMPUH). Sudah banyak jamaah umrah & haji plus yang sudah dibantu oleh Shafira *Tour & Travel* ini pergi ke tanah suci.

2.1.2. Logo Perusahaan

Sebuah logo akan menjadi suatu *brand images* dimana dari suatu perusahaan. Logo juga bersifat persepsi kuat terhadap perusahaan.



Gambar 2.1 Logo Shafira

Adapun arti dari simbol-simbol logo tersebut yaitu :

1. Modern : Shafira selalu mengikuti perkembangan jaman dan terdepan dalam inovasi.
2. Simple : Mudah berurusan dengan Shafira, tanggap dan tidak birokratis.
3. Luwes : Berjiwa melayani.
4. Hitam : Berwibawa.
5. Merah : Semangat berkobar, berjiwa muda dan ceria.
6. Melesat ke atas mendahului yang lain.

2.1.3. Struktur Organisasi dan *Job Description*

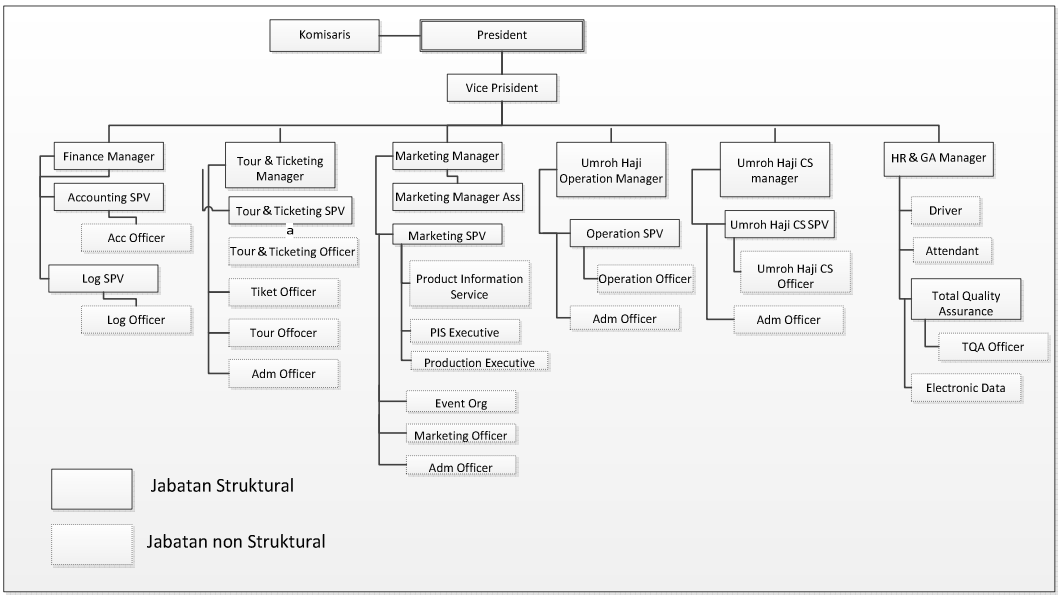
2.1.3.1. Struktur Organisasi

Organisasi Shafira Lintas Semesta terdiri atas :

1. Pimpinan Shafira, yaitu :
   a. Komisaris Shafira.
   b. *President* Shafira.
   c. *Vice President* Shafira.
2. Pengelola fungsi-fungsi pengembangan Shafira, yaitu *unit marketing.*
3. Pengelola fungsi dukungan *manajemen*, yaitu :
   a. Unit finance.
   b. Unit HR & general affair.
4. Pengelola operasi & layanan, yaitu :
   a. Unit tour & ticketing.

b. Unit umrah haji *operation.*

c. Unit umrah haji *customer service* (CS).



Gambar 2.2 Struktur Organisasi Shafira

2.1.3.1. *Job Description*

2.1.3.2.1. *Frontliners*

1. Customer Service

Customer Service adalah bagian terdepan dalam suatu perusahaan yang berperan menghubungkan antara perusahaan secara langsung dengan *customer*. Bidang pekerjaan yang dilakukan oleh *Customer Service* di Shafira *Tour & Travel* adalah :

a. Memberikan informasi seputar produk-produk Shafira *Tour & Travel*.

b. Menerima pendaftaran jamaah.

c. *Handle complain customer*.

d. Operator telepon.

e. Penghubung antara divisi keuangan dan operasional.

2. *Marketing*

Marketing atau pemasaran tidak pernah lepas dari sebuah usaha produk atau jasa. Demikian pun dengan Shafira yang selalu mengembangkan produk jasanya dengan melakukan berbagai strategi *marketing* untuk menarik minat konsumen.

3. *Ticketing*

Divisi *ticketing* berhubungan dengan *reservasi* tiket penerbangan baik domestik maupun internasional, selain penerbangan umroh dan haji.

4. *Tour*

Divisi *tour* dalam deskripsi pekerjaannya adalah membuat promo *tour*, ketika promo tersebut sudah matang kemudian *booked* hotel dan transfortasi. Setelah promo, akomodasi, transfortasi *fix*, baru dilakukan pemasaran produk, bisa juga melalui presentasi jika dalam suatu grup. Ketika ada *customer* yang tertarik untuk menggunakan jasa *tour* tersebut, kemudian dilakukan *reservasi* dan terakhir adalah melakukan pembayaran.

5. *Finance*

Bagian *finance* atau keuangan di Shafira terbagi menjadi beberapa subbagian, antara lain :

a. Kasir *Ticketing*

Pekerjaan yang dilakukan kasir *ticketing* adalah menerima pembayaran tiket dari *customer* dan menerima *invoice* dari bagian *ticketing* yang kemudian mengarsipkannya, setelah itu membuat laporan keuangan.

b. Administrasi *Ticketing*

Pekerjaan yang dilakukan oleh administrasi *ticketing* adalah *cross check* laporan keuangan kasir, mengerjakan *report* selisih atau keuntungan.

c. Money Changer

Money changer dilakukan oleh kasir. Shafira menerima berbagai penukaran mata uang.

d. Kasir Umroh

Pekerjaan yang dilakukan oleh kasir umroh adalah menerima pembayaran tiket dari konsumen dan menerima *invoice* dari *customer service* yang kemudian mengarsipkannya, setelah itu membuat laporan keuangan.

e. *Checker* Keuangan Umroh

Tugas *Checker* Keuangan Umroh adalah mengevaluasi pembukuan yang telah dikerjakan oleh kasir dan mencocokannya dengan *invoice*, membuat jurnal keuangan dan pembukuan umroh.

f. Manager Finance

Manager finance merupakan fungsi *control* dari seluruh laporan keuangan Shafira.

2.1.3.2.2. Operasional

Bagian operasional terbagi menjadi beberapa bagian, yaitu :

1. Pengurusan Dokumen

   Operasional pengurusan dokumen menerima kelengkapan dokumen jamaah dari CS, kemudian memasukkan (*input*) data ke dalam sistem.

2. *Handle* Jamaah

   Handle jamaah ini dilakukan menjelang keberangkatan jamaah. *Team* Shafira menyiapkan segala sesuatu yang dibutuhkan oleh jamaah.

3. *Ticketing*

   Bagian ini menangani jadwal keberangkatan beserta penerbangan yang akan digunakan.

2.2. Landasan Teori

2.2.1 *Data warehouse*

Data warehouse adalah sebuah *database* yang mewakili sejarah bisnis suatu perusahaan atau organisasi. Data historis dari *data warehouse* digunakan dalam aktivitas analisis yang mendukung keputusan bisnis dalam beberapa tingkat. Data di dalam *data warehouse* diorganisir untuk mendukung analisis, bukan transaksi pemrosesan dalam waktu nyata, seperti pada OLTP. Sedangkan OLAP adalah teknologi yang memproses *data warehouse* dalam struktur multidimensi, menyediakan jawaban yang cepat untuk *query* analisis yang kompleks. Tujuan dari OLAP adalah mengorganisir sejumlah data yang besar, agar bisa dianalisis

dan dievaluasi dengan cepat (Chuck Ballard, Dirk Herreman, Don Schau, Rhonda Bell, Eunsaeng Kim, Ann Valencic; IBM Corp, 1998.)

Data warehouse merupakan kumpulan informasi logikal, yang dikumpulkan dari berbagai operasional *database* yang mendukung aktivitas para bisnis analis dalam pekerjaan mengambil keputusan. Kedengaran cukup sederhana pada awalnya, tetapi *data warehouse* menggambarkan hal yang mendasar perbedaan cara pandang tentang mengorganisasi dan me-manajemen informasi dalam suatu organisasi.

Data warehouse merupakan kombinasi dari *database* yang berbeda-beda, *data warehouse* mengkombinasikan informasi dengan meringkas (*summarizing*) dan mengelompokkan (*aggregation).* Pada saat mengekstrak informasi dari berbagai operasional *database* untuk membuat suatu *data warehouse,* pengumpulan informasi yang diperlukan untuk pembuat keputusan. Informasi yang diperlukan ini didefinisikan oleh persetujuan para pengguna sesuai dengan informasi yang mereka butuhkan dalam pengambilan keputusan. Jadi suatu *data warehouse* hanya berisi informasi yang relevan dengan kebutuhan *user* untuk mendukung pengambilan keputusan.

Data warehouse adalah multidimensional dalam relasional model *database,* informasi ditentukan dalam suatu rangkaian dari tabel yang berdimensi dua. Tidak begitu halnya dengan *data warehouse,* banyak *data warehouse* adalah multidimensional, artinya bahwa *data warehouse* terdiri dari *layer-layer*, kolom dan baris. *Layer-layer* dalam suatu *data warehouse* menunjukkan informasi yang

disampaikan pada dimensi yang berbeda. Multidimensional ini menggambarkan informasi dengan ditunjukkan sebagai suatu kubus atau *hypercubes.*

Data warehouse adalah bentuk khusus dari *database*. Mengingat hal bahwa suatu *database* adalah koleksi dari informasi yang dikumpulkan dan diakses melalui suatu logikal dari informasi, begitu juga hal yang sama untuk suatu *data warehouse*. Para pengguna dari *data warehouse* menyatakan informasi yang mereka butuhkan secara logikal dan tidak memperhatikan atau merisaukan tentang baris, kolom, atau *layer*. *Data warehouse* juga mempunyai suatu kamus data. Kamus data dalam *data warehouse* mengolah struktur logika dari informasi dan memiliki dua karakteristik tambahan yang sangat penting, *origin* dan metode. Jadi suatu kamus data *data warehouse* selalu mengikuti informasi yang bersumber dari operasional *database* dan dengan suatu metode (total, jumlah, rata-rata, standar deviasi dan lain-lain).

Data warehouse mendukung proses pembuatan keputusan, bukan proses transaksi. Dalam suatu organisasi, banyak *database* yang ada merupakan *database* yang berorientasikan pada transaksi. Banyak *database* yang mendukung *Online Transaction Processing* (OLTP) oleh karena itu merupakan operasional *database*. *Data warehouse* tidak berorientasikan pada transaksi, *data warehouse* ada untuk mendukung berbagai macam pekerjaan pengambilan keputusan dalam suatu organisasi. Oleh karena itu *data warehouse* mendukung *Online Analytical Processing* (OLAP).

Definisi lain dari konsep *data warehouse* adalah *database* yang secara khusus dibuat strukturnya untuk proses *query* dan analisa. Dalam dunia bisnis sebuah *data warehouse* biasanya berisi keadaan atau gambaran data tentang sejarah bisnis (misalnya, penjualan produk satu tahun) dari sebuah organisasi bisnis. Definisi klasik oleh Bill Inmon tentang *data warehouse* dapat disimpulkan menjadi empat kriteria.

Pertama, sebuah *data warehouse* adalah *subject-oriented* (berorientasi pada subjek). Data suatu organisasi berubah dari yang berorientasi pada aplikasi (*application-orriented*) ke berorientasi pada subjek (*subject-oriented*). Menurut hal ini, daripada mengorganisasikan data dengan penyesuaian pada aplikasi atau fungsi-fungsi yang ada, data lebih baik dikumpulkan berdasarkan subjek area asalnya. Misalkan data tentang konsumen yang diintegrasikan pada sebuah entitas *database* akan lebih baik daripada mengklasifikasikannya menjadi beberapa entitas untuk proses peminjaman, begitu juga yang lainnya.

Kedua, *data warehouse* adalah terintegrasi. *Data warehouse* menggabungkan berbagai format data dan menyandikannya kedalam suatu bentuk yang konsisten sehingga dalam membandingkan dan mengumpulkan data dengan melalui berbagai dimensi yang sesuai.

Ketiga, *data warehouse* adalah *time-variant*. Dalam kata lain, setiap *rows* dari data dibedakan berdasarkan waktu. Dalam hal ini setiap baris biasanya memiliki satu atau lebih yang menggunakan tipe *data time-stamp*.

Keempat, *data warehouse* tidak mudah berubah. Data yang ada dalam suatu *data warehouse* tidak mengalami proses penghapusan atau peng-update-an, terkecuali untuk perawatan dan koreksi terhadap kesalahan yang terjadi. Data hanya disimpan ke dalam *data warehouse* atau ditampilkan dari *data warehouse.*

Keempat karakteristik di atas saling terkait dan harus diimplementasikan agar terbentuk suatu *data warehouse* yang dapat mendukung pengambilan keputusan secara efektif. Implementasi dari keempat karakteristik di atas membutuhkan struktur data dari *data warehousing,* data dari berbagai sumber operasional akan diekstrak dan diintegrasikan ke dalam *data warehouse* sehingga data yang dihasilkan tidak lagi bersifat operasional melainkan informatif.

Dalam membangun sebuah *data warehouse,* informasi yang diambil dari berbagai *processing system* harus relevan, dalam periode waktu yang konsisten dan tidak mengalami perubahan secara cepat. Keuntungan yang diberikan sehubungan dengan adanya *data warehouse* antara lain (Han, 2001) :

a. Memberikan *competitive advantages* dengan menampilkan informasi yang relevan dalam mengukur *perfomance* dan membuat keputusan kritis untuk menghadapi persaingan dengan *competitor*.
b. Meningkatkan produktivitas bisnis dikarenakan *data warehouse* secara cepat dan mudah mengumpulkan informasi secara tepat.
c. Memberikan fasilitas *Customer Relationship Management* (CRM) karena *data warehouse* mampu memberikan pandangan yang konsisten tentang *customer* dan barang yang dimiliki perusahaan.

Pembuatan sebuah *data warehouse* selalu diawali dengan membuat bisnis dimensional model yang menggambarkan dimensi dan ukuran dari subjek yang dipilih didasarkan pada kebutuhan pengguna. Tidak seperti dalam sistem *Online Transaction Processing* (OLTP) yang mengorganisasi data dalam bentuk normalisasi secara ketat, data yang terdapat dalam *data warehouse* diatur dalam bentuk yang tidak ternomalisasi (denormalisasi). Dengan demikian waktu yang dibutuhkan untuk menjalankan prosesnya menjadi lebih cepat. Berikut akan dijelaskan tentang pengertian denormalisasi :

a. Denormalisasi

   Denormalisasi merupakan kebalikan dari proses normalisasi. Relasi pada suatu *database* yang mengalami denormalisasi memungkinkan terjadinya *data redundant* di dalamnya. Artinya adalah memungkinkan adanya data yang sama atau berulang dalam sebuah tabel. Hal inilah yang dapat mengakibatkan kesalahan dalam proses pengolahan data. Namun di sisi lain, denormalisasi memberikan keuntungan dalam segi *perfomance*. Karena itu aplikasi yang membutuhkan waktu cepat terhadap proses *query* cenderung memilih denormalisasi untuk mengatur relasi antar tabel dalam suatu *database.*

   Denormalisasi menyebabkan data *redundant* namun meningkatkan *perfomance.* Melihat keuntungan tersebut, OLAP *tools* dan *data warehouse* menggunakan denormalisasi dalam *database*. Dengan demikian proses *query* dapat berlangsung dengan cepat.

2.2.1.1 Tugas *Data warehouse*

Ada empat tugas yang bisa dilakukan dengan adanya *data warehouse*, keempat tugas tersebut yaitu :

a. Pembuatan laporan

   Pembuatan laporan merupakan salah satu kegunaan *data warehouse* yang paling umum dilakukan. Dengan menggunakan *query* sederhana didapatkan laporan per hari, per bulan, per tahun atau jangka waktu kapanpun yang diinginkan.

b. *On-Line Analytical Processing* (OLAP)

   Dengan adanya *data warehouse,* semua informasi baik *detail* maupun hasil *summary* yang dibutuhkan dalam proses analisa mudah didapat. OLAP mendayagunakan konsep data multidimensi dan memungkinkan para pemakai menganalisa data sampai mendetail, tanpa mengetikkan satupun perintah SQL. Hal ini dimungkinkan karena pada konsep multidimensi, maka data yang berupa fakta yang sama bisa dilihat dengan menggunakan fungsi yang berbeda. Fasilitas lain yang ada pada *software* OLAP adalah fasilitas *rool-up* dan *drill-down*. *Drill-down* adalah kemampuan untuk melihat *detail* dari suatu inforamsi dan *roll-up* adalah kebalikannya.

c. Data Mining

   Data mining merupakan proses untuk menggali (*mining)* pengetahuan dan informasi baru dari data yang berjumlah banyak pada *data warehouse*, dengan menggunakan kecerdasan buatan (*Artificial Intelegence*), statistik

dan matematika. *Data mining* merupakan teknologi yang diharapkan dapat menjembatani komunikasi antara data dan pelakunya.

d. Proses informasi *executive*

Data warehouse dapat membuat ringkasan informasi yang penting dengan tujuan membuat keputusan bisnis, tanpa harus menjelajahi keseluruhan data. Dengan menggunakan *data warehouse* segala laporan telah diringkas dan dapat pula mengetahui segala rinciannya secara lengkap, sehingga mempermudah proses pengambilan keputusan. Informasi dan data pada laporan *data warehouse* menjadi target informatif bagi pengguna.

2.2.1.2 Proses dan Arsitektur *Data warehouse*

Data warehouse dibangun dengan mengintegrasikan data-data yang berasal dari berbagai sumber data, yaitu *database* operasional. Dalam suatu perusahaan, data operasional biasanya berada pada daerah kekuasaan departemen masing-masing dalam bentuk *database* OLTP. Untuk melakukan proses integrasi ini *data warehouse* menggunakan suatu aplikasi yang disebut ETL (*Extract, Transform, Load*).

Sesuai dengan namanya, aplikasi ETL melakukan fungsi-fungsi *Extract, Transform,* dan *Load.* Proses *extract* adalah proses pengambilan data dari sumber data. Disebut *extract*, karena proses pengambilan data ini tidak mengambil data

matang saja. Proses *extract* ini harus mengakomodir berbagai macam teknologi yang digunakan oleh sumber data dan diintegrasikan ke dalam *database* tunggal.

Kemudian data hasil *extract* ini menjalani proses transformasi yang pada prinsipnya adalah mengubah kode-kode yang ada menjadi kode-kode standar, misalnya kode propinsi. Hal ini perlu dilakukan mengingat data-data yang diambil berasal dari sumber yang berbeda yang kemungkinan memiliki standardisasi yang berbeda pula. Standardisasi diperlukan untuk nantinya memudahkan pembuatan laporan.

Arsitektur *data warehouse* yang umum biasanya menempatkan satu *server database* terpisah yang disebut *Staging* yang berfungsi untuk menangani proses *extract* dan *transform* ini sebelum dilakukan proses *load* ke tujuan akhir *data warehouse.*

Proses *load* dalam ETL adalah suatu proses mengirimkan data yang telah menjalani proses transformasi ke gudang data akhir, yaitu *data warehouse* itu sendiri dimana aplikasi *reporting* dan *business intelligence* siap mengakses. Di bawah ini adalah penjelasan dari tiap-tiap proses *Extract, Transform,* dan *Load* itu sendiri :

1. Extract

Bagian pertama dari suatu proses ETL adalah men-ekstrak data dari sumber data. Disebut ekstrak, karena proses pengambilan data ini tidak mengambil keseluruhan data yang ada di *database* operasional, melainkan hanya mengambil data matang saja. Proses ini meliputi penyaringan data yang akan

digunakan dalam pembuatan *data warehouse*. Dapat langsung dimasukkan langsung dalam *data warehouse* atau dimasukkan dalam tempat penampungan sementara terlebih dahulu.

Pada hakikatnya bagian dari ekstraksi melibatkan penguraian dari data yang telah diekstrak, menghasilkan suatu pengecekan jika data bertemu dengan suatu struktur atau pola yang diharapkan. Jika bukan, data tersebut mungkin ditolak secara keseluruhan.

2. Transform

Proses yang ke dua adalah transformasi data yang telah diekstrak ke dalam format yang diperlukan. Hal ini perlu dilakukan mengingat data yang diambil berasal dari sumber yang berbeda yang kemungkinan memiliki standardisasi yang berbeda pula. Data dari beberapa sistem perlu ditransformasi ke dalam format umum yang disepakati dan digunakan dalam *data warehouse*. Diperlukan pengetahuan tentang arti penyimpanan data dalam sistem operasional, seperti :

a. Bisa jadi untuk suatu item yang sama memiliki dua nama yang berbeda, misalnya nama produk dengan kode “1234” disebut “roti” sedangkan kode yang sama “1234” di tempat lain disebut “kue”.
b. Setiap sistem mungkin menggunakan skema pengkodean yang berbeda. Misal kode produk di suatu tempat direpresentasikan dengan karakter yang dipisahkan dengan tanda ‘-‘(xxx-xx-xxx) sedangkan di tempat lain untuk kode produk merupakan direpresentasikan dengan karakter yang

dipisahkan dengan tanda spasi (xxx xx xxx). Jika dimasukkan ke dalam *warehouse*, format yang digunakan harus sama.

c. Atribut tabel memiliki nama berbeda. Misalnya untuk atribut *gender* pada tabel *customer* suatu sistem menggunakan nilai "0" atau "1". Sedangkan di sistem lain, digunakan "M" atau "F".

d. Setiap sistem menggunakan unit pengukuran *(measure*) yang tidak sama. Misal *dollar* digunakan di US, dan *euro* digunakan di Erofa. Data ini harus dikonversi ke dalam unit pengukuran yang sama jika dimasukkan ke dalam *warehouse.*

Dalam proses transformasi, nama harus diseragamkan, dan skema pengkodean yang dipakai harus seragam pula. Jika data sekali ditransformasi, maka siap untuk di-load ke dalam *warehouse.* Seringkali area transformasi merupakan tempat yang terpisah.

3. Load

Tahap *load* adalah men-load data ke dalam target akhir (*end target*), yang pada umumnya adalah *data warehouse* (DW). Bergantung pada kebutuhan organisasi, proses ini bervariasi secara luas. Beberapa *data warehouse* memperbolehkan melakukan penulisan informasi yang ada secara kumulatif, dengan data yang diperbaharui tiap minggu, ketika DW lain (atau bahkan bagian lain dari DW yang sama) boleh menambahkan data baru dalam suatu format historis, sebagai contoh, tiap jam. Pemilihan waktu dan lingkup untuk menggantikan atau menambahkan aneka pilihan desain strategi

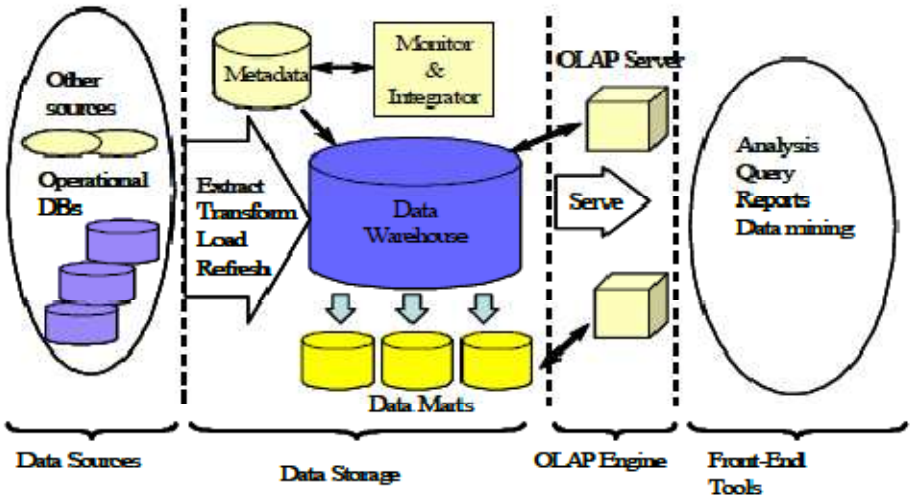
bergantung pada waktu yang tersedia dan kebutuhan bisnis tersebut. Kebanyakan sistem yang komplek dapat memelihara suatu histori dan jejak audit dari semua perubahan yang ada ke data yang di-load ke dalam *data warehouse*.

Arsitektur adalah sekumpulan atau struktur yang memberikan kerangka untuk keseluruhan rancangan suatu sistem atau produk, ada arsitektur *client-server*, arsitektur *networking* dan masih banyak arsitektur lainnya. Arsitektur data menyediakan kerangka dengan mengindentifikasikan dan memahami bagaimana data akan pindah melalui sistem dan digunakan dalam perusahaan.

Arsitektur data untuk *data warehouse* mempunyai komponen utama yaitu *read-only database*. Karakteristik arsitektur *data warehouse* :

1. Data diambil dari sistem asal (sistem informasi yang ada), *database* dan *file*.
2. Data dari sistem asal diintegrasikan dan ditransformasi sebelum disimpan ke dalam *Database Management System* (DBMS) seperti Oracle, MS SQL Server, Sybase dan lain-lain.
3. *Data warehouse* merupakan sebuah *database* terpisah bersifat hanya dapat dibaca yang dibuat khusus untuk mendukung pengambilan keputusan.
4. Pemakai mengakses *data warehouse* melalui aplikasi *front end tool*.

Arsitektur dan komponen utama dari *data warehouse* dapat dilihat pada gambar berikut ini :



Gambar 2.3 Arsitektur *data warehouse*

Gambar 2.3 Menunjukan proses *data warehouse*, dimana *data source* diolah (*extract*, *transform*, *load*) menjadi *data warehouse* untuk selanjutnya di *query* untuk menghasilkan *report* yang diinginkan. *Data mart* berisikan kumpulan data yang digunakan untuk pengambilan keputusan. *Data mart* hampir sama dengan *data warehouse*, tetapi perbedaannya adalah *data mart* lebih spesifik dan ditujukan untuk bagian-bagian tertentu.

2.2.1.3 *Multi-Dimensional Modeling*

Multi dimensional modeling adalah teknik untuk memvisualisasi model data sebagai suatu kumpulan dari ukuran yang dideskripsikan dengan aspek-aspek bisnis (Chuck Ballard, Dirk Herreman, Don Schau, Ronda Bell, Eunseang Kim,

Ann Valencic; IBM Corp., 1998). Hal ini khususnya sangat berguna untuk meringkas dan menyusun data dan memperlihatkan data untuk mendukung para analis data. *Dimensional modeling* memfokuskan pada data numerik, seperti harga, jumlah, berat, keseimbangan dan kejadian-kejadian.

Dimensional modeling mempunyai beberapa konsep :

1. Fact

Fact adalah suatu koleksi dari relasi data-data *items*, terdiri dari ukuran-ukuran dan konteks data. Setiap *fact* biasanya merepresentasikan sebuah bisnis *item*, suatu transaksi bisnis, atau sebuah kejadian yang dapat digunakan dalam analisis bisnis atau proses bisnis. Dalam *data warehouse*, *fact* di implementasikan dalam tabel dasar dimana semudah data *numeric* dan disimpan.

2. Dimensions

Dimensions adalah suatu koleksi dari anggota atau unit-unit data dengan tipe yang sama. Dalam sebuah diagram, suatu dimensi biasanya direpresentasikan dengan suatu *axis.* Dalam dimensional model, semua data menunjukan *fact table* yang diasosiasikan dengan satu dan hanya satu member sari setiap *multiple dimensions.* Jadi dimensi menunjukan latar belakang kontekstual dari *fact.* Banyak proses analisis yang digunakan untuk menghitung (*quatify*) dampak dari dimensi pada *fact.*

Dimensi adalah parameter dari apa yang ingin lakukan dalam *On-Line Analytical Processing (OLAP).* Sebagai contoh, dalam suatu *database* untuk menganalisa semua penjualan dari produk, dimensi berikut ini harus ada :

Waktu

Lokasi

Pembeli

Penjual

Skenario seperti kejadian, pembiayaan atau angka estimasi

Dimensi biasanya juga dapat dipetakan bukan angka numerik, *entity* yang bersifat informatif seperti merek atau karyawan.

Dimension member : suatu dimensi berisi banyak anggota-anggota. Suatu anggota dimensi adalah nama pembeda atau *indetifier* yang digunakan untuk membedakan posisi suatu data item. Sebagai contoh, semua bulan, empat bulanan dan tahunan membuat dimensi waktu. Dan semua kota, wilayah dan negara merupakan dimensi goegrafi.

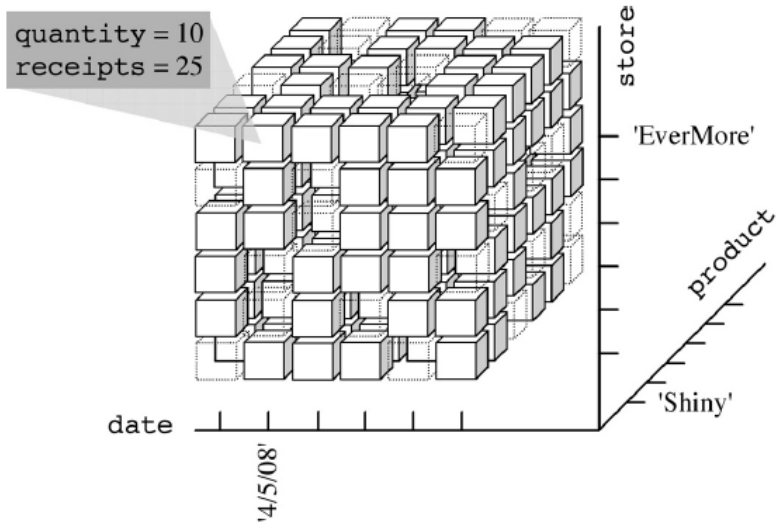
Dimension Hierarchies : dapat mengatur anggota dari suatu dimensi kedalam satu atau banyak hirarki. Setiap hirarki dapat juga mempunyai banyak level hirarki. Setiap member dari dimensi tidak boleh dilokasikan kedalam satu struktur hirarki.

3. Measures

Suatu *measures* (ukuran) adalah suatu besaran (angka numerik) atribut dari sebuah *fact*, yang menunjukan *performance* atau *behavior* (tingkah laku) dari bisnis secara relatif pada suatu dimensi. Angka atau nomor yang ditunjukan disebut dengan *variable*. Sebagai contoh ukuran dari penjualan dalam bentuk uang, besarnya penjualan, jumlah pengadaan, biaya pengadaan, banyaknya transaksi dan lainnya. Suatu ukuran dijelaskan dengan kombinasi dari member dari suatu dimensi dan diletakkan dalam *fact.*

2.2.1.3.1 Visualisasi Dari Suatu Dimensional Model

Cara yang paling populer dalam memvisualisasikan suatu model dimensional adalah dengan menggambarkan sebuah *cube* (Gambar 2.4) dapat menggambarkan tiga model dimensional menggunakan sebuah kubus. Biasanya suatu dimensional model terdiri dari lebih tiga dimensi dan digambarkan sebagai suatu *hypercube*, akan tetapi *hypercube* sulit untuk divisualisasikan, jadi sebuah kubus lebih biasa digunakan. Gambar berikut merupakan *measurement* adalah volume dari produksi, yang mana dijelaskan dengan kombinasi dari tiga dimensi yaitu lokasi produk dan waktu. Dimensi dari lokasi dan dimensi produk mempunyai dua level hiraki. Misalnya dimensi mempunyai *level region* dan *plant*.

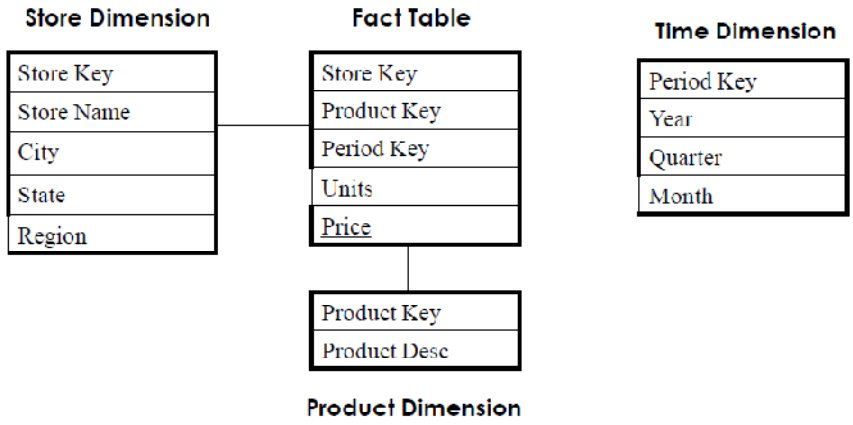


Gambar 2.4 Kubus suatu visualisasi dari dimensional model.

Terdapat tiga konsep yang dapat memodelkan lebih jelas dalam penentuan *fact table* dimensi dan *measure*, yaitu :

1. Model *star*

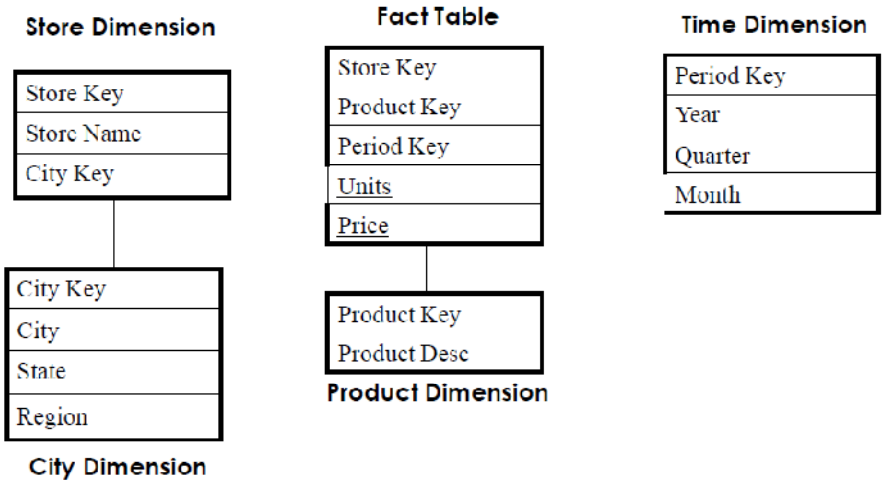
   Model *star* adalah model dasar dari pemodelan multidimensi yang memiliki satu tabel induk yang dinamakan *table fact* dan kumpulan dari tabel-tabel kecil yang disebut tabel dimensi disusun dalam pola-pola melingkar mengelilingi *table fact.* Untuk lebih jelasnya terlihat pada gambar 2.5



Gambar 2.5 Model *Star*

2. Model *SnowFlake*

Selain model *star* dalam menampilkan desain sebuah *data warehouse* juga dapat menggunakan model *snowflake.* Dalam sebuah model *snowflake* terdapat sebuah *fact table* yang dikelilingi oleh beberapa *dimension table.* Namun *dimension table* itu sendiri dapat seolah-olah berupa sebuah *fact table* lain yang juga memiliki dimensinya sendiri. Untuk lebih jelasnya terlihat pada gambar 2.6

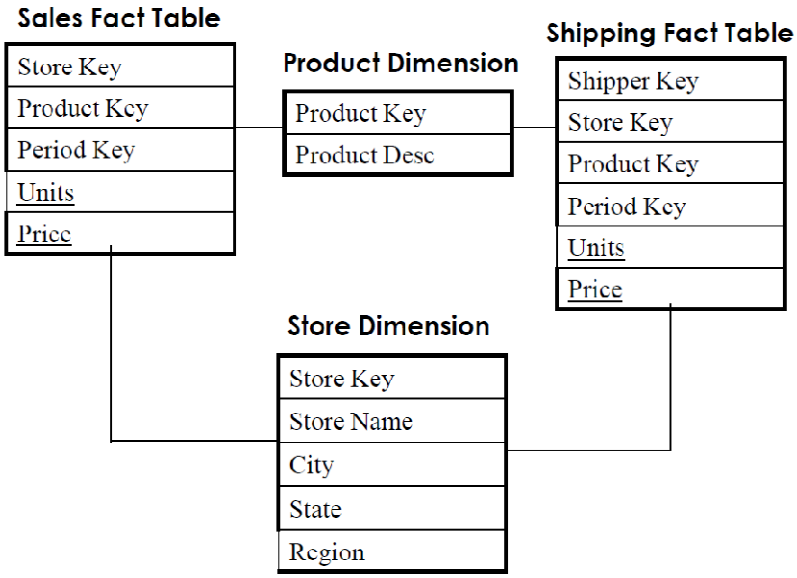


Gambar 2.6 Model *Snowflake*

Pada model *snowflake* diatas dapat dilihat bahwa tiap-tiap tabel dimensi berhubungan dengan dimensi yang lain seolah-olah tabel dimensi tersebut merupakan tabel fakta. Hal ini dapat terjadi karena pada model *snowflake* ini telah dilakukan normalisasi. Normalisasi ini tidak terdapat pada model *star*.

3. Model *Constellation.*

   Beberapa *fact tables* berbagi tabel dimensi. Ditampilkan sebagai koleksi dari kumpulan skema bintang yang sering disebut sebagai skema *galaxy*. Untuk lebih jelasnya terlihat pada gambar 2.7

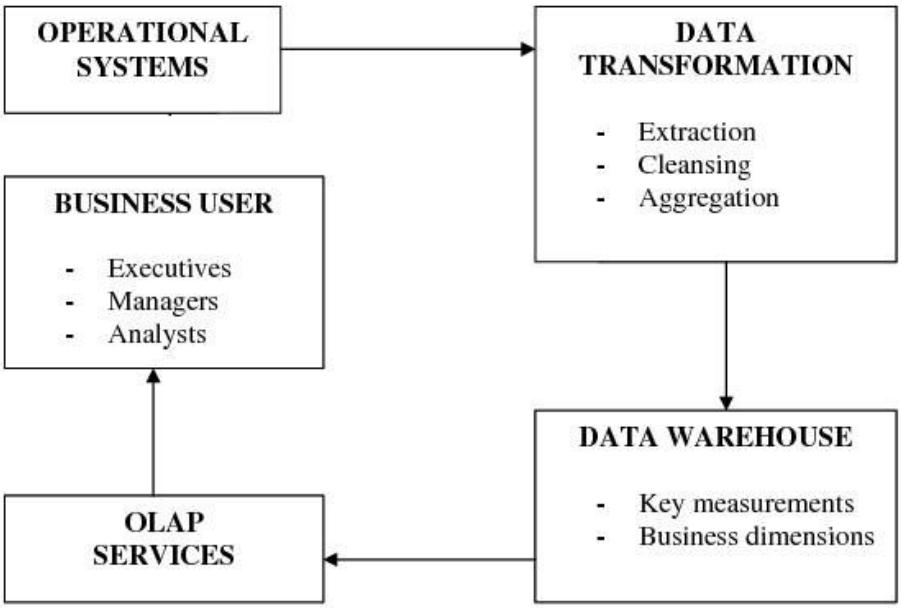


Gambar 2.7 Model *Constellation*

2.2.1.4 *Online Analytical Processing* (OLAP)

Online Analytical Processing (OLAP) merupakan salah satu *tools* yang digunakan untuk mengakses informasi dalam *data warehouse*. Teknologi OLAP memungkinkan *data warehouse* digunakan secara efektif untuk proses *online* analisis, memberikan respon yang cepat terhadap *analytical queries* yang kompleks.

Data model multidimensional dan teknik agregasi data yang dimilki oleh OLAP dapat mengatur dan membuat kesimpulan dari data dalam jumlah besar sehingga dapat di evaluasi secara cepat dengan menggunakan *online analysis* dan *grafical tool*. Sistem OLAP menyediakan kecepatan dan fleksibilitas untuk men-support proses analisis secara *real time*.



Gambar 2.8 *Data warehouse* : perpaduan beberapa teknologi

2.2.1.4.1 OLAP *Sevices*

OLAP services ialah suatu istilah yang menggambarkan beberapa perangkat yang bekerja bersama-sama untuk membantu dalam menganalisasis data yang ada.

Informasi dari *data warehouse* di ekstrak secara periodik dan digunakan untuk meng-update obyek yang ada dalam *OLAP service*. *OLAP server* mengambil data dari *data warehouse* dan *Relational OLAP* serta meng-update informasi yang disimpan dalam *multidimensional OLAP* (MOLAP). Setelah informasi dikumpulkan maka *pivot table service* yang bekerja sama dengan *Excel pivot table* atau perangkat lain yang mendukung OLE DB dengan ekstensi OLAP dapat melakukan akses dan memanipulasi data yang ada.

2.2.1.4.2 Teknik Penyimpanan Data Dalam OLAP

OLAP digunakan untuk memproses informasi dan menampilkanya dalam bentuk multidimensi. Walaupun data yang ditampilkan dalam satu bentuk namun tidak berarti data-data yang ada disimpan dalam satu bentuk pula. Ada tiga teknik dasar untuk menyimpan data OLAP :

1. MULTIDIMENSIONAL OLAP (MOLAP)

   Salah satu cara umum yang digunakan untuk menyimpan data ialah dalam vasis data multidimensional. Tidak seperti basis data relasional yang menyimpan informasi dalam sejumlah baris dalam tabel, sebuah basis data multidimensional menyimpan informasi dalam sejumlah *array*

multidimensional. Karena dimensi dapat diakses secara mudah, maka *user* dapat melakukan *query* pada basis data MOLAP dengan sangat cepat. Selain mengandung data mentah, basis data MOLAP juga mengandung agregasi data sehingga dapat memberikan respon yang cepat terhadap *query*.

2. *RELATIONAL* OLAP (ROLAP)

   Partisi menggunakan tabel *relational* dalam *data warehouse* untuk menyimpan agregasi, sedangkan *detail* dari *fact table* tetap tersimpan dalam *data warehouse fact table*. Banyak orang berpendapat bahwa basis data yang dirancang secara khusus untuk sebuah keperluan analisis tidak dibutuhan karena sebuah basis data relasional sudah cukup mampu untuk menampilkan informasi OLAP. Hal ini hanya berlaku pada tingkat tertentu saja, pada sebuah basis data yang terdiri dari ribuan atau ratusan ribu *records* maka menampilkan informasi OLAP akan menjadi sebuah masalah karena banyak data yang harus dihapus di *query.* Dan hal inilah yang menjadi keterbatasan partisi ROLAP.

3. *HYBRYD* OLAP (HOLAP)

   Partisi menggunakan struktur MOLAP untuk menyimpan agregasi dan meninggalkan *detail* dari *fact table* dalam partisi ROLAP. OLAP *service* dapat menggunakan data MOLAP dan ROLAP secara simultan untuk memecahkan suatu *query*.

2.2.1.4.3 OLAP *Cube* (Kubus OLAP)

Objek utama yang disimpan dalam sebuah basis data OLAP ialah *cube* (kubus). Sebuah kubus merupakan representasi multidimensi dari sekumpulan data, yang mengandung data secara *detail* maupun rangkumannya. Sebuah basis data OLAP dapat memiliki beberapa buah kubus sesuai dengan yang dibutuhkan, yang menggambarkan data yang ada dalam *data warehouse*.

Sebuah kubus dibangun menggunakan dua komponen utama yaitu ukuran (*measure*) dan dimensi. Ukuran merupakan nilai numerik dari *fact table* dalam *data warehouse* seperti harga dari unit maupun kualitas dari *item*. Sedangkan dimensi menggambarkan kategori dari ukuran yang ada, seperti bagaimana ukuran berubah setiap waktu. Beberapa operasi yang dapat dilakukan pada kubus atau *report* multidimensi ini adalah :

1. ***Processing***

   Operasi *Processing* pada kubus antara lain melakukan *loading* dan *refreshing* data pada kubus yang dilakukan oleh OLAP *service engine*. Tabel dimensi dibaca pertama kali untuk mengumpulkan *level* dengan anggota dari data aktual, setelah itu dilanjutkan dengan pembacaan *fact table* dan ikuti dengan menghitung spesifik agregasi, dan hasil yang dapat disimpan dalam kubus untuk diproses oleh *user*. *Processing* akan dilakukan terhadap kubus hanya jika terjadi perubahan terhadap struktur kubus tersebut atau terjadi perubahan terhadap data yang ada dalam *data warehouse*. Ketika akan melakukan *processing* terhadap kubus, terdapat

dua pilihan. Pertama dengan melakukan *refresh* terhadap kubus secara total, yang berarti menghapus seluruh data yang ada sebelumnya dan mengambil data yang baru dari *data warehouse*, atau dengan melakukan *incremental update* yang berarti mengambil data terbaru yang ada dalam *data warehouse* dan menambahkannya pada data dalam kubus. Pilihan kedua akan memakan waktu yang jauh lebih singkat, namun *incremental update* tidak dapat dilakukan jika struktur kubus telah mengalami perubahan.

2. Slice And Dice

Slice and dice memungkinkan untuk melihat kubus dari sudut pandang yang berbeda. Dengan *slice and dice* dapat ditentukan dimensi apa yang hendak ditampilkan dan bagaimana mereka ditampilkan. Hal inilah yang menjadi keunggulan OLAP. Dengan melihat kubus dari sudut pandang yang bebeda maka akan dapat dipelajari banyak hal dari data yang dimiliki.

3. Drill Down

Sebagian besar informasi yang ditampilkan dalam OLAP merepresentasikan kesimpulan yang lebih *detail*. *Drilling down* merupakan teknik untuk memecahkan sebuah informasi menjadi beberapa informasi yang lebih *detail*. Sebagai contoh, jika dilakukan *drill down* terhadap data tahunan maka akan dapat diperoleh data dalam catur wulan atau tri wulan.

4. ***Consolidation (Roll Up)***

   Consolidation atau yang lebih dikenal *roll up* merupakan kebalikan dari *Drill Down*. Data-data sebelumnya dilihat dalam format triwulan akan dapat dilihat dalam format tahunan. Dengan *roll up* data dapat dilihat secara lebih global.

5. ***Pivotting***

   Pivotting merupakan suatu teknik untuk saling menukarkan dimensi data. Dengan melakukan *pivotting* maka dapat diamati suatu informasi atau data dari sudat pandang yang berbeda, sehingga diharapkan akan dapat memperjelas analisis yang dilakukan.

6. ***Filtering***

   Filltering merupakan suatu teknik untuk menyaring informasi yang dibutuhkan, sehingga para analisis tidak dibingungkan oleh banyaknya informasi yang tersedia.

2.2.2 *Software* Pendukung Aplikasi

Dalam menyelesaikan proyek ini, digunakan bahasa pemrograman Borland Delphi 7 sebagai program *coding* dan desain *interface* dan MySql sebagai pembuat *data wearehouse.*

2.2.2.1 Borland Delphi

Borland Delphi atau biasa yang disebut Delphi saja, merupakan sarana pemrograman aplikasi *visual*. Bahasa pemrograman yang digunakan adalah

bahasa pemrograman Pascal atau kemudian juga yang disebut bahasa pemrograman Delphi. Delphi merupakan generasi penerus dari Turbo Pascal. Turbo Pascal yang diluncurkan pada tahun 1983 dirancang untuk dijalankan pada sistem operasi DOS (yang merupakan sistem operasi yang paling banyak digunakan pada saat itu). Sedangkan Delphi yang diluncurkan pertama kali tahun 1995 dirancang untuk beroperasi dibawah sistem operasi *windows*.

Delphi adalah *compiler* (penterjemah) bahasa Delphi (awalnya dari pascal) yang merupakan bahasa tingkat tinggi sekelas dengan basic, C. Bahasa pemrograman di Delphi disebut bahasa *procedural* yaitu bahasa atau sintaknya mengikuti urutan tertentu. Delphi disebut juga *Visual Programming* artinya komponen–komponen yang ada tidak hanya berupa teks tetapi muncul berupa gambar–gambar.

Delphi memiliki sarana untuk pembuatan aplikasi, mulai dari sarana untuk pembuatan *form, menu, toolbar,* hingga kemampuan untuk menangani pengelolaan basis data yang besar. Kelebihan–kelebihan yang dimiliki Delphi antara lain karena pada Delphi, *form* dan komponen-komponennya dapat dipakai ulang dan dikembangkan, tersedia *template* aplikasi dan *template form,* memiliki lingkungan pengembangan visual yang dapat diatur sesuai kebutuhan, menghasilkan *file* terkompilasi yang berjalan lebih cepat, serta kemampuan mengakses data dari bermacam–macam format.

Delphi menggunakan bahasa objek pascal didalam lingkungan pemrograman visual. Kombinasi ini menghasilkan sebuah lingkungan pengembangan aplikasi yang berorientasi objek (*Object Oriented Programming*). Dengan konsep seperti

ini, maka pembuatan aplikasi menggunakan Delphi dapat dilakukan dengan cepat dan menghasilkan aplikasi yang tangguh. *Form* dan komponen yang ada didalamnya, misalnya, dapat disimpan dalam suatu paket komponen yang dapat digunakan kembali, atau dimodifikasi seperlunya saja.

Khususnya untuk pemrograman *database*, Delphi menyediakan *object* yang sangat kuat, canggih dan lengkap, sehingga memudahkan pemrograman dalam merancang, membuat dan menyelesaikan aplikasi *database* yang diinginkan. Selain itu, Delphi juga dapat menangani data dalam berbagai format *database*, misalnya format MS.Access, Oracle, Foxro, Informix dan lain–lain. Format *database* yang dianggap asli dari Delphi adalah Paradox dan dBase.

Keunggulan yang dimiliki oleh Borland Delphi yaitu :

a. Memiliki banyak fitur dan komponen.

b. Mudah dalam penulisan *coding.*

c. *Compatible* dengan berbagai macam jenis *database.*

2.2.2.2 Pengenalan MySQL

MySQL merupakan sebuah *software* yang berguna sebagai suatu *Database Server* yang cukup terkenal. *Database server* itu sendiri merupakan suatu *software* yang bertugas untuk melayani permintaan (*request*) *query* dari *client.*

MySQL sebagai suatu *database server* mempunyai beberapa kemampuan, salah satunya harus menyediakan suatu sistem manajemen *database* yang dapat mengatur bagaimana menyimpan, menambah, mengakses data dan transaksi-transaksi *database* lainnya. MySQL cepat sekali berkembang, karena MySQL merupakan suatu *software* yang bersifat *Open Source*.

2.2.2.3 Pengenalan *Zeos* 6.6.6 *Stable*

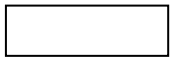
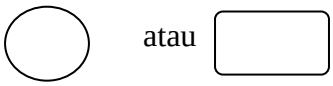
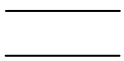
ZeosLib adalah sebuah paket yang terdiri dari beberapa komponen yang dapat menghubungkan secara langsung sebuah aplikasi dengan berbagai mesin basis data. Komponen-komponen *Zeos* tidak memiliki *internal driver* sehingga memerlukan pustaka eksternal (DLL) yang sesuai dengan mesin basis data yang dituju. Sebagai contoh, untuk dapat mengakses mesin basis data MySQL komponen ini memerlukan pustaka eksternal LibmySQL.dll atau variannya sesuai dengan versi MySQL yang dituju.

2.2.3 *Data Flow Diagram* (DFD)

Data Flow Diagram (DFD) merupakan alat yang biasa dipakai untuk mendokumentasikan proses dalam sistem. DFD adalah suatu gambaran grafis dari suatu sistem yang menggunakan sejumlah bentuk-bentuk simbol untuk menggambarkan bagaimana data mengalir melalui suatu proses yang saling berkaitan.

DFD menggunakan empat macam simbol dasar, yaitu *entity, data flow,* proses, dan *data store* (Kendall & Kendall, 2002). Untuk lebih jelasnya gambar simbol dan keterangannya dapat dilihat pada tabel 2.

Tabel 2.1 Simbol DFD, arti dan keterangannya

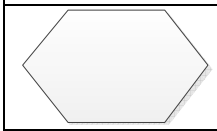
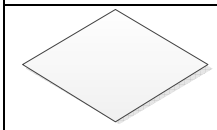
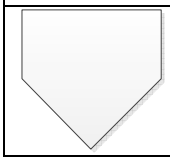
Simbol	Arti	Keterangan
	*Entity*	Merupakan sumber atau tujuan data.
	*Data flow*	Menunjukan aliran data diantara proses, *data storage*, dan *data source*. Memiliki panah yang menunjukkan kemana arah data mengalir.
	Proses	Menunjukkan transformasi dari data. Pada proses minimal ada satu data yang masuk ke proses dan data yang keluar dari proses, dimana data yang masuk tidak boleh sama dengan data yang keluar.
	*Data storage*	Tempat untuk menyimpan data.

Sumber : Kenneth E. Kendall & Julie E. Kendall, *System Analysis And Design, Person Education Internasional*, 2002, p. 192

2.2.4 *Flowchart*

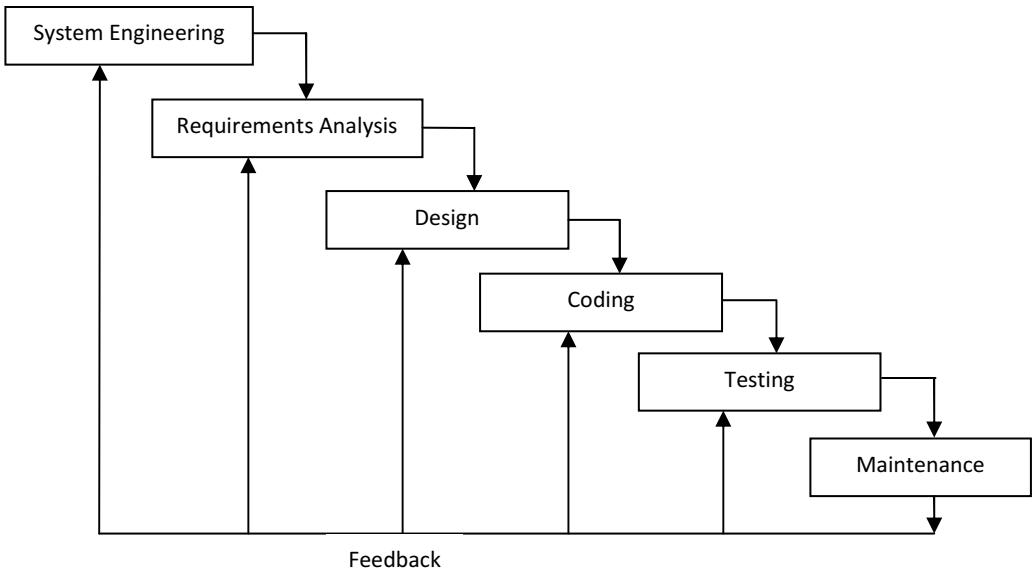
Flowchart merupakan representasi grafik dari algritma dengan menggunakan simbol-simbol tertentu yang masing-masing mempunyai fungsi yang khusus. *Flowchart* menggambarkan atau langkah dari sistem (proses, operasi, fungsi atau aktifitas). *Flowchart* mempunyai beberapa simbol yang digunakan. Beberapa simbol lain yang dipakai dan penjelasannya dapat dilihat pada tabel.

Tabel 2.2 simbol *flowchart*, arti dan keterangannya

Simbol	Arti	Keterangan
	Terminator	Untuk mulai atau mengakhiri program
	Garis Alir	Arah aliran suatu proses dalam program
	*Preparation*	Proses inisialisasi atau pemberian harga awal
	Proses	Proses perhitungan atau pengolahan data
	*Input/Output* data	Merepresentasikan *Input data* atau *Output data* yang diproses atau Informasi.
	*Predefined* proses (Sub Program)	Permulaan subprogram/proses menjalankan subprogram
	*Decision* (keputusan dalam pengujian)	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
	*On Page Connector*	Penghubung bagian *flowchart* yang terpisah tetapi masih dalam satu halaman
	*Off Page Connector*	Penghubung bagian *flowchart* yang terpisah pada halaman berbeda

2.2.5 Metodologi *Waterfall*

Disebut juga siklus klasik (1970-an) dan sekarang ini lebih dikenal dengan sekuensial linier. Metode ini membutuhkan pendekatan sistematis dan sekuensial dalam pengembangan *software*, dimulai dari *system engineering, analisys, design* (perancangan), *coding, testing, dan maintenance.*



Gambar 2.9 *Metode waterfall*

a. System / Information Engineering

Merupakan bagian dari sistem yang terbesar dalam pengerjaan suatu proyek, dimulai dengan menetapkan berbagai kebutuhan dari semua elemen yang diperlukan sistem dan mengalokasikannya kedalam pembentukan perangkat lunak.

b. Analisis

Merupakan tahap menganalisis hal-hal yang diperlukan dalam pelaksanaan proyek pembuatan perangkat lunak.

c. Design

Tahap penerjemahan dari data yang dianalisis kedalam bentuk yang mudah dimengerti oleh *user*.

d. Coding

Tahap penerjemahan data atau pemecahan masalah yang telah dirancang keadalam bahasa pemrograman tertentu.

e. Testing

Merupakan tahap pengujian terhadap perangkat lunak yang dibangun.

f. Maintenance

Tahap akhir dimana suatu perangkat lunak yang sudah selesai dapat mengalami perubahan–perubahan atau penambahan sesuai dengan permintaan *user*.