BAB 2
TINJAUAN PUSTAKA

2.1 Profil PT. Blantickindo Aneka

PT. Blantickindo Aneka yang berlokasi di Wijaya Grand Center Blok B-17, Jalan Wijaya II Kel.Pulo Kec.Kebayoran Baru, Jakarta Selatan. Didirikan sejak tahun 1986 sebagai jawaban atas peningkatan kebutuhan keahlian dalam bidang teknik, lingkungan, bio-fisik, sosial ekonomi dan institusi, seiring dengan pertumbuhan ekonomi Indonesia yang berkembang pesat. Kami telah menjadi anggota Ikatan Konsultan Indonesia (INKINDO) sejak tahun 1986 dan telah melakukan pekerjaan pelayanan jasa konsultan, baik dari pemerintah maupun swasta. Wilayah kerja perusahaan ini telah menjangkau beberapa kota-kota besar di Indonesia, antara lain: DKI Jakarta, Padang, Surabaya, Bali, NTT, NTB, Makasar, Nias, Riau, Maluku dan kota-kota besar lainnya.

2.1.1 Logo PT. Blantickindo Aneka

Logo dari PT. Blantickindo Aneka dapat dilihat pada gambar dibawah ini.

Gambar 2.1 Logo PT. Blantickindo Aneka

2.1.2 Visi dan Misi PT. Blantickindo Aneka

Visi dan misi dari PT. Blantickindo Aneka adalah sebagai berikut:

1. Visi

   Visi dari PT. Blantickindo Aneka saat ini adalah:

   "Menjadi perusahaan yang senantiasa mampu bersaing dan tumbuh berkembang di pasar infrastruktur Indonesia dengan bijak serta terpercaya dalam menjalankan usahanya".

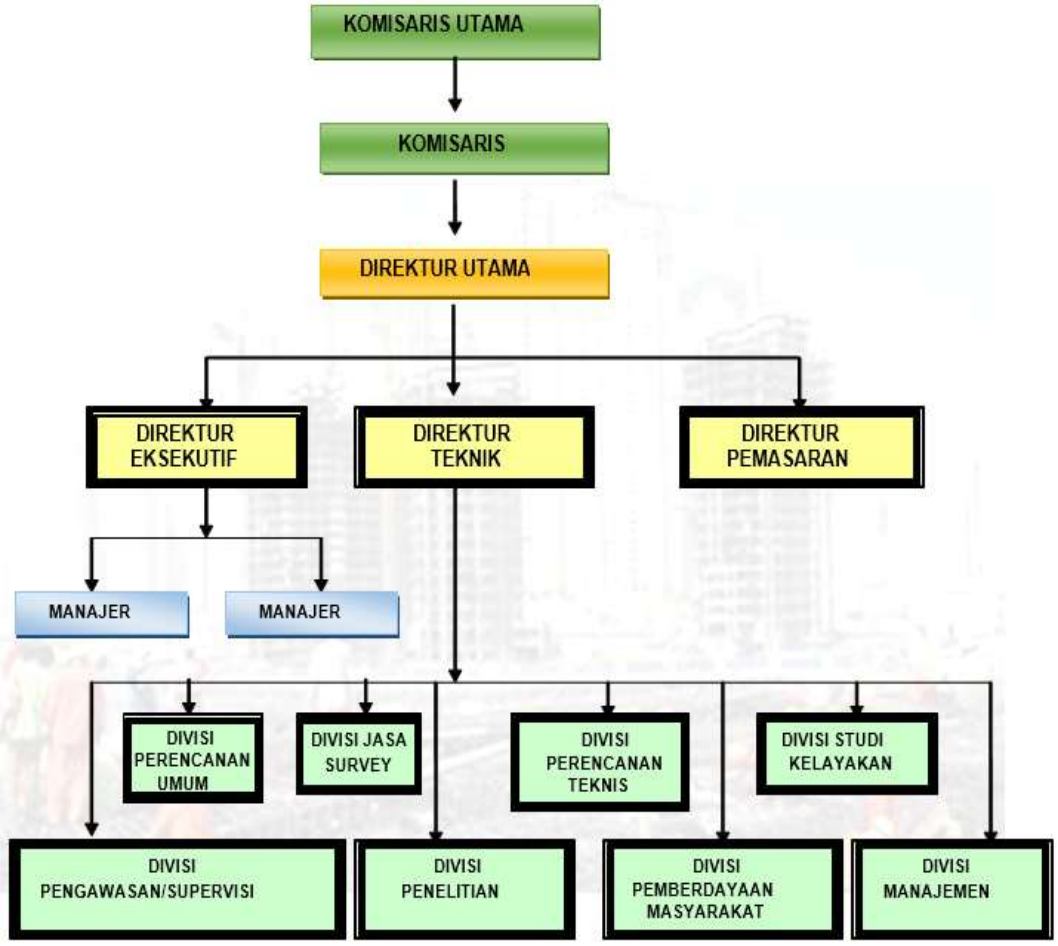
2. Misi

   Misi dari PT. Blantickindo Aneka adalah sebagai berikut:

   1. Mengelola perusahaan secara profesional, terbuka dan mematuhi peraturan perundangan yang berlaku untuk menghasilkan kinerja yang ekselen.
   2. Meningkatkan nilai tambah dan kinerja finansial perusahaan dengan memperhatikan efisiensi biaya dan efektifitas program.
   3. Menghasilkan laba yang pantas untuk mendukung pengembangan perusahaan.
   4. Memberikan kepercayaan penuh kepada *Client* dalam menjalin kerja sama dibidang jasa.

2.1.3 Struktur Organisasi PT. Blantickindo Aneka

Susunan organisasi PT. Blantickindo Aneka dapat dilihat pada gambar 2.2 dibawah ini.



Gambar 2.2 Struktur Organisasi PT. Blantickindo Aneka

2.2 Landasan Teori

Landasan teori membahas mengenai materi atau teori apa saja yang digunakan sebagai acuan dalam membuat tugas akhir ini. Landasan teori yang diuraikan merupakan hasil dari studi literatur yaitu pengumpulan data dengan cara mempelajari jurnal, browsing internet, referensi buku dan bacaan-bacaan lainnya yang berhubungan erat dengan penelitian yang dilakukan.

2.2.1 *Database* (Basis Data)

Basis data atau *database* merupakan sekumpulan data yang tersimpan secara sistematik, di mana data tersebut dapat diolah menjadi sebuah informasi yang berguna. *Database* dikelola dan diolah oleh sebuah sistem yang disebut *DBMS (Database Management System).* Sedangkan *Database Management System* (DBMS) itu sendiri

atau yang dikenal dengan sistem manajemen basis data berfungsi sebagai sistem dalam pengolahan basis data sehingga menjadikan sebuah informasi baru. Sistem ini memungkinkan untuk menyusun, mengolah dan memperbaharui *item* dalam suatu basis data. Sistem ini memiliki kemampuan untuk mengolah data dalam jumlah yang besar, dan juga dapat melakukan manipulasi data dengan cepat dan mudah. Tujuan utama dari DBMS adalah menyediakan cara untuk menyimpan dan mengambil informasi dari *database* dengan baik, nyaman, dan efisien **.** [9]

2.2.2 Data *Warehouse*

Data warehouse atau gudang data adalah suatu sistem pengoleksian data yang mempunyai sifat seperti, berorientasi pada subjek, ter integrasi, *time –variant*, dan *non-volatile* untuk kebutuhan pengambilan keputusan suatu perusahaan. Dalam penggunaannya, data *warehouse* menunjang akan sistem DSS (*Decission Support System)* dan EIS (*Executive Information System).* Data *warehouse* didesain untuk menganalisis data berdasarkan subjek – subjek tertentu dalam suatu organisasi. Dalam data *warehouse*, data yang berasal dari sumber yang terpisah kemudian disimpan dalam suatu format yang konsisten dan saling ter integrasi satu dengan yang lainnya, oleh sebab itu data tidak dapat dipecah karena data yang telah ada merupakan suatu kesatuan yang menunjang keseluruhan konsep dari *data*

warehouse tersebut. Seluruh data yang ada pada data *warehouse* dikatakan akurat dan valid hanya pada rentang waktu tertentu, dan data yang ada pada data *warehouse* tidak dapat dilakukan *update* data secara *real-time* tetapi dengan melakukan *refresh* dari sistem operasional secara reguler, data yang baru akan selalu ditambahkan sebagai suatu "suplemen" bagi *database* itu sendiri, untuk kemudian secara *incremental* disatukan dengan data yang sudah ada sebelumnya. Berdasarkan definisi tersebut, karakteristik dari data *Warehouse* ini sebagai berikut [11] :

1. *Subject oriented* (berorientasi subjek)

   Data *Warehouse* dirancang untuk membantu *User* dalam pengambilan keputusan. Contohnya untuk mengetahui tentang data penjualan perusahaan. Kita bisa membangun data *Warehouse* yang berfokus pada penjualan. Dengan menggunakan data *Warehouse*, kita bisa menjawab pertanyaan seperti "Siapa pembeli terbaik untuk barang ini tahun lalu?". Kemampuan untuk mendefinisikan sebuah data *Warehouse* sebagai sebuah subjek, dalam hal ini penjualan, membuat data *Warehouse subject oriented*.
2. *Integrated* (terintegrasi)

   Data *Warehouse* dikonstruksikan dengan cara mengintegrasikan berbagai sumber data yang berbeda. Data yang terintegrasi menyebabkan data tersebut lebih konsisten, sehingga lebih mudah dipahami oleh para pembuat keputusan.
3. *Time-variant*

   Data *Warehouse* harus bisa menghasilkan informasi dari sudut pandang historical (misalnya informasi 5-10 tahun yang lalu atau bahkan lebih). Atau bisa dikatakan bahwa data *Warehouse* berfokus pada perubahan setiap waktunya.
4. *Non-volatile*

   Data yang ada dalam data *Warehouse* tidak bisa di-*edit* ataupun di-*update*.

   Data *Warehouse* dibuat untuk melayani *User* (*analyst* dan pengambil keputusan). Sehingga data *Warehouse* wajib dirancang sesuai dengan persyaratan berikut :

   a. Harus bisa memberikan kepuasan kepada setiap *User*.

b. Memiliki *function* sendiri tanpa mengganggu OLTP *systems*.
c. Menyediakan pusat tempat penyimpanan data yang konsisten.
d. Menjawab setiap *complex queries* dengan cepat.
e. Menyediakan berbagai analisis *tools* yang kuat, seperti OLAP dan data *mining*.

Sebagian besar data *Warehouse* yang sukses selain memenuhi persyaratan di atas juga memiliki beberapa karakteristik seperti :

1) Berdasarkan model dimensional.
2) Mengandung *historical* data.
3) Terdiri dari *detailed* dan *summarized* data.
4) Tetap mempertahankan konsistensi data walaupun berasal dari sumber yang berbeda.
5) Fokus dalam *single subject*, seperti penjualan, keuangan, atau inventarisasi.

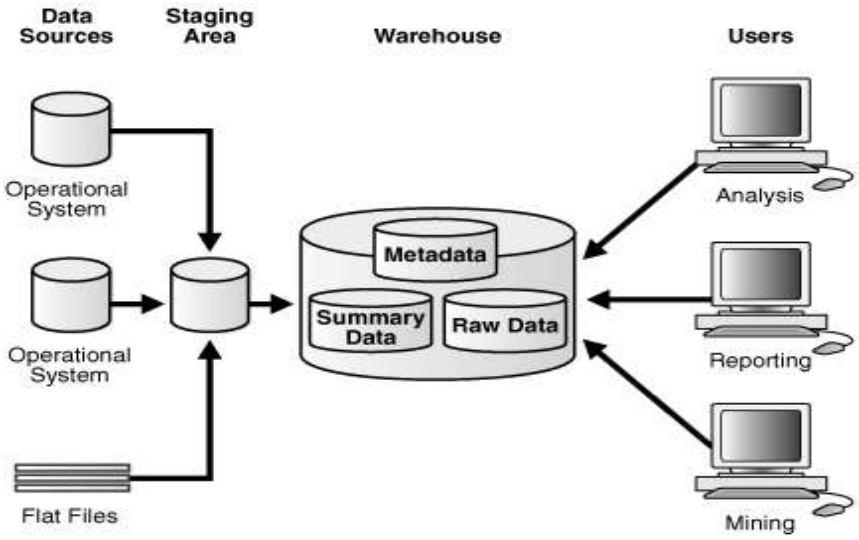
Tabel 2.1 Perbedaan OLTP dan Data *Warehouse*

OLTP	Data *Warehouse*
Dirancang untuk operasi *real-time* bisnis	Dirancang untuk analisis dari suatu bisnis berdasarkan atribut dan kategori
Menangani data saat ini	Menangani data saat ini dan data masa lalu
Data disimpan pada beberapa *platform*	Data disimpan pada satu platform saja
Data diorganisir berdasarkan fungsi atau operasinya	Data diorganisir berdasarkan subjek
Prosesnya bersifat berulang (*loop*)	Prosesnya dilakukan setiap saat dan harus berorientasikan waktu (*historical*)
Untuk operasional	Untuk managerial
Berorientasi pada transaksi	Berorientasi pada analisis

2.2.2.1 Arsitektur Data *Warehouse*

Arsitektur data warehouse merupakan struktur yang menyajikan semua komponen yang terlibat di dalam data *warehouse* secara bersamaan. Di dalam data *warehouse*, arsitektur termasuk data yang ter integrasi sebagai satuan yang terpusat, semua kebutuhan untuk persiapan data dan penyimpanannya, dan arah penyajian informasi dari data *warehouse* sehingga menghasilkan suatu aturan, prosedur, dan fungsional untuk memungkinkan data *warehouse* bekerja dan memenuhi kebutuhan bisnis.

Salah satu arsitektur yang dapat digunakan adalah arsitektur *three major areas* yang meliputi data *acquisition*, data *storage*, dan *information delivery*. Untuk lebih jelasnya terdapat pada [12].

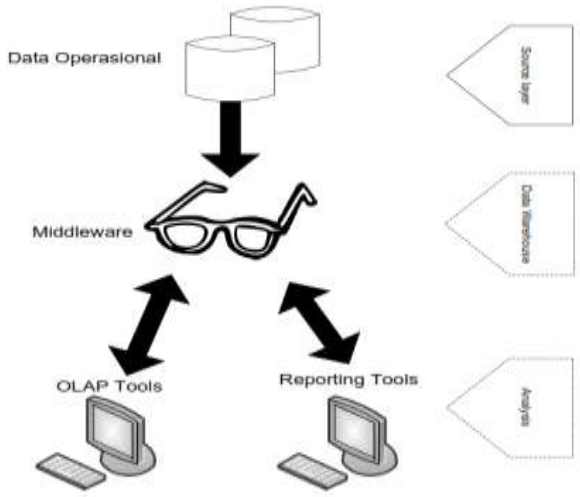


Gambar 2.3 *Three Major Areas* Arsitektur Data *Warehouse*

Arsitektur Data *Warehouse* menurut Matteo Golfarelli dan Stefano Rizzi dalam bukunya yang berjudul Data *Warehouse Design: Modern Principles and Methodologies*, mengelompokkan arsitektur data *warehouse* menjadi 3 kelompok, terdiri dari *single layer architechture, two layer architecture,* dan *three layer architecture.*

a. Single Layer Architechture

Single Layer Architechture pada umumnya tidak sering digunakan dalam suatu kasus. Tujuannya adalah untuk meminimalkan jumlah data yang disimpan, untuk mencapai tujuan ini, ia bisa menghilangkan redudansi data. Gambar 2.4 menunjukkan hanya lapisan fisik yang tersedia. ini berarti bahwa data *warehouse* di implementasikan sebagai pandangan multidimensi data operasional yang dibuat oleh *middleware* tertentu, atau lapisan pengolahan menengah. *Middleware* merupakan komponen perantara yang memungkinkan *client* dan (lapisan aplikasi dan sistem operasi) saling terhubung dan berkomunikasi satu sama lain.



Gambar 2.4 Single Layer Architecture

Kelemahan arsitektur ini terletak pada kegagalan untuk memenuhi persyaratan untuk pemisahan antara pengolahan analisis dan transaksional. Query analisis yang disampaikan kepada data operasional setelah middleware menganalisis. Untuk alasan ini, pendekatan virtual untuk data warehouse dapat berhasil hanya jika analisis kebutuhan utama dibatasi dan volume data untuk tujuan analisis sangat besar.

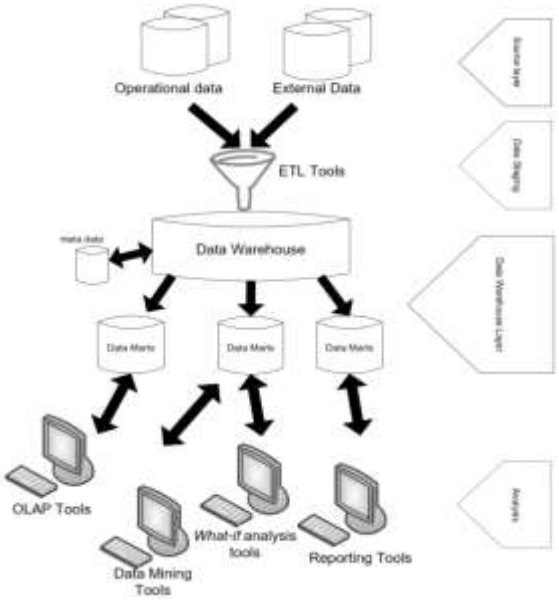
b. Two layer architechture

Two layer architechture yang di tunjukan pada Gambar 2.5 merupakan arsitektur dari data *warehouse* yang menggunakan dua *layer* atau lapisan sebagai proses pemisahan antara sumber yang tersedia dengan data *warehouse*. Arsitektur ini memiliki empat tahap aliran data, yaitu:

1. *Source Layer*, Sebuah sistem data *warehouse* menggunakan sumber data heterogen. Data yang awalnya disimpan ke *database* relasional perusahaan atau mungkin berasal dari sistem informasi perusahaan.
2. *Staging* Data, Data yang disimpan ke sumber harus diekstrak, kemudian dibersihkan untuk menghilangkan inkonsistensi, dan terintegrasi untuk menggabungkan sumber yang heterogen menjadi satu skema umum. Skema umum tersebut adalah Ekstraksi, Transformasi, dan *Loading* (ETL).
3. Data *warehouse layer*, pada lapisan ini data dapat diakses secara langsung, tetapi juga dapat digunakan sebagai sumber untuk membuat data

mart dan dirancang untuk departemen perusahaan tertentu. Meta data sebagai repository yang menyimpan informasi pada sumber, *staging* data, dan sebagainya.

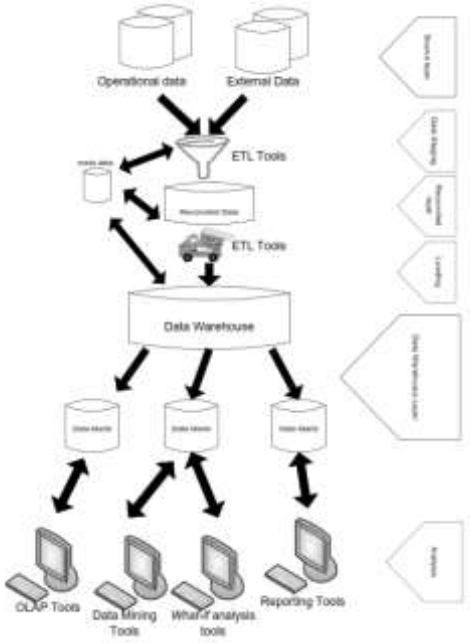
4. *Analysis*, data yang terintegrasi secara efisien dan fleksibel diakses untuk mencetak laporan, menganalisis informasi yang di butuhkan dan mensimulasikan hipotesis skenario bisnis yang membahas berbagai jenis pengambilan keputusan perusahaan.



Gambar 2.5 Two Layer Architecture

c. Three Layer Architechture

Dalam arsitektur ini, lapisan ketiga adalah lapisan penyesuaian data (*reconciled* data). Lapisan ini terwujud dari data operasional yang diperoleh setelah mengintegrasikan dan membersihkan sumber data. Akibatnya, data tersebut adalah terintegrasi, konsisten, dan rinci. Gambar 2.6 menunjukkan sebuah data *warehouse* yang tidak di dapat dari sumbernya secara langsung, tetapi diperoleh dari *reconciled* data.



Gambar 2.6 Three Layer Architecture

Berdasarkan arsitekturnya, terdapat tiga tugas umum yang dapat dilakukan dengan adanya data *warehouse*, ketiga tugas tersebut yaitu :

1. Pembuatan Laporan

   Pembuatan laporan merupakan salah satu kegunaan data *warehouse* yang paling umum dilakukan. Dengan menggunakan *query* sederhana laporan harian, bulanan, tahunan atau jangka waktu tertentu dapat didapatkan kapanpun sesuai dengan kebutuhan untuk menunjang proses pendukung keputusan bisnis.

2. OLAP *(Online Analytical Processing)*

   Data yang terdapat dalam data *warehouse* tidak akan bermanfaat secara optimal apabila tidak dapat di optimalkan dengan baik, OLAP merupakan suatu *tools* untuk membantu proses analisis data yang terdapat dalam data *warehouse* dengan menggunakan konsep data multidimensi yang memungkinkan para penggunanya menganalisa data dengan berbagai sudut pandang dan aspek bisnis yang dinamis dengan lebih detail tanpa perlu mengetikan perintah SQL . Hal ini dapat memungkinkan untuk mendapatkan informasi yang lebih berguna terhadap data yang ada dalam

data *warehouse* untuk menjadi informasi yang bersifat strategis dan taktikal untuk menunjang keputusan bisnis secara lebih baik.

3. Proses informasi *executive*

   Data *warehouse* dapat menunjang akan sistem DSS *(Decission Support System)* dan EIS *(Executive Information System),* data *mart* juga didesain agar menjadi sebuah skema data yang dapat mudah untuk di analisis karena sama-sama menggunakan skema data yang bersifat *multidimensional*.

   Kumpulan data yang ada dalam data *warehouse* dapat di olah dan disajikan kedalam bentuk informasi berupa visualisasi data yang dapat lebih mudah untuk di analisis dan dimengerti oleh para penggunanya yang berfungsi seperti layaknya sistem DSS *(Decission Support System),* informasi tersebut dapat disajikan kedalam bentuk *pivot table* ataupun grafik dan *chart* statistik yang dapat menunjang para eksekutif atau karyawan dengan tingkatan *top level management* untuk melakukan analisis terhadap kumpulan data perusahaan dengan lebih mudah tanpa harus menganalisa kesuluruhan data yang tidak dibutuhkan untuk keputusan bisnis.

Selain tiga tugas diatas, data *warehouse* juga menunjang proses untuk penggalian data atau biasa disebut data *mining*. Data *mining* merupakan proses untuk menggali *(mining)* pengetahuan dan informasi baru dari data yang berjumlah banyak pada data *mart* dengan menggunakan kecerdasan buatan *(Artificial Inteligence),* statistik dan model matimatis. Apabila tugas-tugas umum yang telah disebutkan sebelumnya digabungkan dengan data *mining* sistem aplikasi tersebut dapat dikatakan sebuah BPM *(Business Process Management)* dan bukan lagi hanya sebatas *reporting tools* umum.

2.2.2.2 *Dimensional Modeling* Data *Warehouse*

Kebutuhan *User* dan realitas data yang menjadi faktor penentu untuk merancang dimensional model data *Warehouse*, seperti bisnis apa yang paling diperlukan, detailnya seperti apa dan dimensi-dimensi serta fakta-fakta apa yang harus diikutkan.

Maka dimensional model harus disesuai dengan kebutuhan dari *User*. Model juga harus dirancang sedemikian rupa agar dapat bertahan dan dapat beradaptasi dari segala perubahan yang akan terjadi. Desain modelnya yang dihasilkan dibentuk menjadi *database* relasional yang mendukung OLAP *cubes* untuk menyediakan secara "instant" hasil *query* untuk analis.

Sembilan tahap metodologi dalam perancangan database untuk *data warehouse,* yaitu [10]:

1. Langkah 1 : Pemilihan Proses
    a. *Data Mart* yang pertama kali dibangun haruslah *data mart* yang dapat dikirim tepat waktu dan dapat menjawab semua pertanyaan bisnis yang penting.
    b. Pilihan terbaik untuk *data mart* yang pertama adalah yang berhubungan dengan sales, misal *property salesa*, *property leasing*, *property adversting*.
2. Langkah 2 : Pemilihan Sumber
    a. Proses pemilihan secara pasti apa yang diwakili atau direpresentasikan oleh sebuah tabel fakta.
    b. Misal, jika sumber dari sebuah tabel fakta *property sale* adalah *property sale* individual maka sumber dari sebuah dimensi pelanggan berisi rincian pelanggan yang membeli *property* utama.
3. Langkah 3 : Mengidentifikasi Dimensi
    a. Set dimensi yang dibangun dengan baik, memberikan kemudahan untuk memahami dan enggunakan *data mart*.
    b. Dimensi ini penting untuk menggambarkan fakta – fakta yang terdapat pada tabel fakta. Misal, pada setiap data pemesanan pada tabel dimensi pemesanan dilengkapi dengan kd_pemesanan, kd_pelanggan, total_bayar dll.
    c. Jika ada dimensi yang muncul pada dua *data mart*, kedua *data mart* tersebut harus berdimensi sama, atau paling tidak salah satunya berupa subset matematis dari yang lainnya.
    d. Jika sebuah dimensi digunakan pada dua *data mart* atau lebih, dan dimensi ini tidak disinkronisasi, maka keseluruhan *data warehouse* akan gagal, karena dua *data mart* tidak bisa digunakan secara bersama – sama.

4. Langkah 4 : Pemilihan Fakta
    a. Sumber dari sebuah tabel fakta menentukan fakta mana yang bisa digunakan *data mart*.
    b. Semua fakta harus diekpresikan pada tingkat yang telah ditentukan oleh sumber.
5. Langkah 5 : Menyimpan pre-kalkulasi di Tabel Fakta
    a. Hal ini terjadi apabila fakta kehilangan statement.
6. Langkah 6 : Melengkapi Tabel Dimensi
    a. Pada tahapan ini kita menambahkan keterangan selengkap – sengkapnya pada tabel dimensi.
7. Langkah 7 : Pemilihan Durasi Database
    a. Misalnya pada suatu perusahaan asuransi mengharuskan data disimpan selama 5 tahun atau lebih.
8. Langkah 8 : Menulusuri Perubahan Dimensi yang Perlahan
    a. Atribut dimensi yang telah berubah tertulis ulang.
    b. Atribut dimensi yang telah berubah menimbulkan sebuah dimensi baru.
    c. Atribut dimensi yang telah berubah menimbulkan alternatif sehingga nilai atribut lama dan yang telah berubah menimbulkan alternatif sehingga nilai atribut lama dan harus dapat diakses secara bersama pada dimensi yang sama.
9. Langkah 9 : Menentukan Prioritas dan Mode Query
    a. Pada tahap ini lebih menggunakan perancangan fisik.

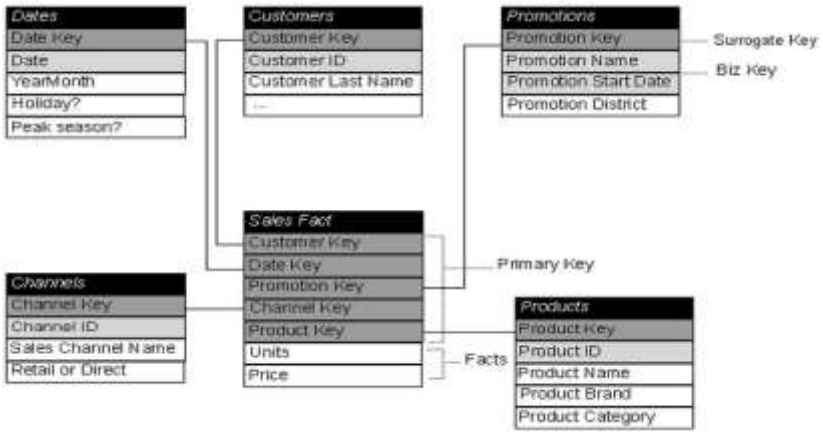
2.2.2.3 Skema Data Warehouse

Ada beberapa konsep permodelan data *warehouse* pada *dimensionality modelling* yang dikenal umum pada saat ini, konsep-konsep tersebut antara lain *star schema, snowflake* dan *fact constellation schema.*

1. Skema S*tar*

Skema bintang adalah skema *data warehouse* yang paling sederhana. Skema ini disebut skema bintang karena hubungan antar tabel dimensi dan tabel fakta menyerupai bintang, dimana satu tabel fakta dihubungkan dengan beberapa

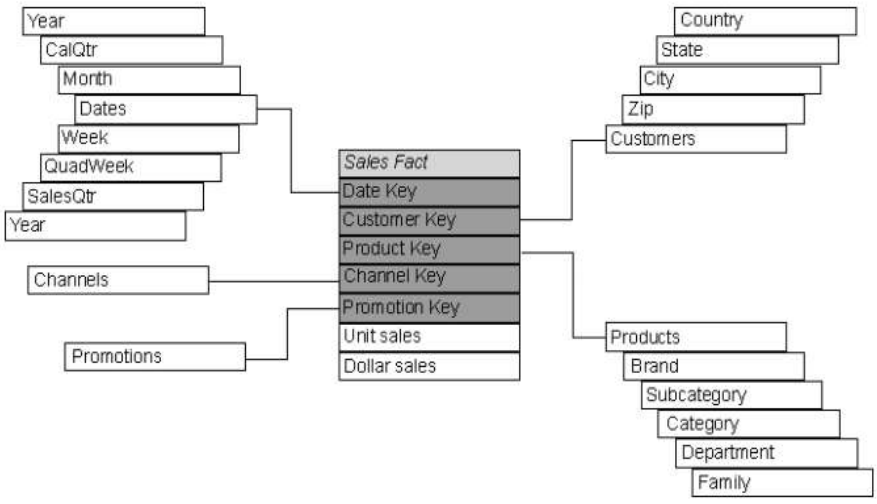
tabel dimensi. Titik tengah skema bintang adalah satu tabel fakta besar dan sudut-sudutnya adalah tabel – tabel dimensi. Keuntungan yang didapat jika menggunakan skema ini adalah peningkatan kinerja *data warehouse*, pemrosesan *query* yang lebih efisien, dan waktu respon yang cepat.



Gambar 2.7 Skema Star

2. Skema *Snowflake*

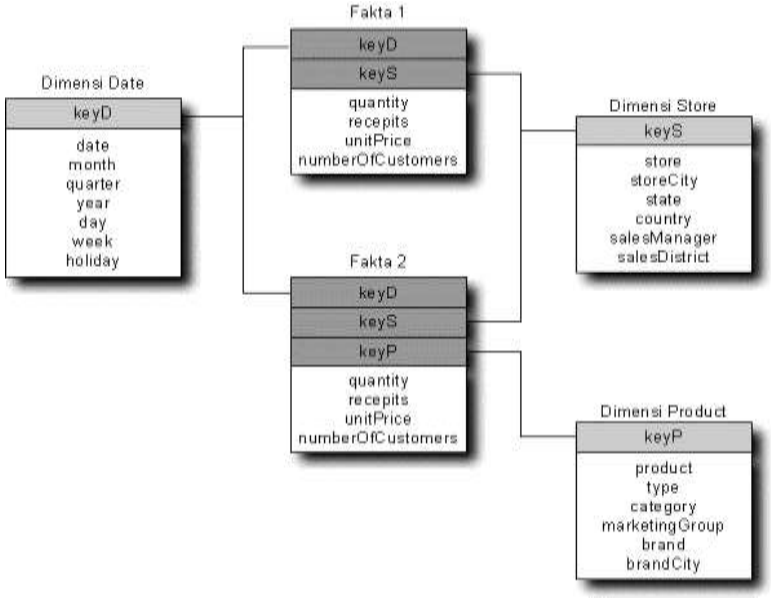
Suatu skema disebut skema *snowflake* jika satu atau lebih tabel dimensi tidak berhubungan langsung dengan tabel fakta tetapi harus berhubungan melalui tabel dimensi lain. Keuntungan yang didapat dengan menggunakan skema ini adalah penghematan *memory*, tapi waktu yang dibutuhkan untuk pemrosesan *query* menjadi lebih lama.



Gambar 2.8 Skema Snowflake

3. Skema Constellation

Suatu skema dikatakan sebagai skema *constellation* jika ada satu tabel dimensi yang dipakai bersamaan oleh satu atau lebih tabel fakta, Keuntungan menggunakan skema ini adalah menghemat *memory* dan mengurangi kesalahan yang mungkin terjadi. [5].



Gambar 2.9 Skema *Constellation*

2.2.2.4 Proses ETL Data *Warehouse*

ETL merupakan himpunan fungsi yang dilakukan untuk mengubah dan membentuk kembali data ke dalam bentuk yang berbeda pada data di dalam sistem operasional yang disimpan di dalam data *Warehouse* sebagai informasi yang relevan dan strategis. Adapun kelompok himpunan *ETL* adalah ekstraksi data, transformasi, dan *loading* yang menjadi tahapan proses pengubahan dan pembentukan ulang data yang digunakan di dalam data *Warehouse*.

Proses ETL atau biasa disebut Extract, Transform, dan Load merupakan proses pengubahan data dari OLTP database menjadi data *Warehouse*. Jika dilihat dari arstitektur data *Warehouse*, proses ETL ini merupakan proses yang berada di data staging. Proses ETL merupakan proses untuk mengubah, memformat ulang serta mengintegrasikan data yang berasal dari satu atau beberapa OLTP sistem [3].

a. Extraction

Data Extraction adalah proses pengambilan data yang diperlukan dari sumber data warehouse dan selanjutnya di masukan pada staging area untuk diproses pada tahap berikutnya. Pada fungsi ini banyak berhubungan dengan berbagai tipe sumber data seperti : format data, mesin yang berbeda, software dan arsitektur yang tidak sama. Sehingga sebelum proses ini dilakukan, sebaiknya perlu didefinisikan requirement terhadap sumber data yang yang akan digunakan untuk proses berikutnya. Adapun fungsi ekstrasi diantaranya, yaitu :

1. *Ekstraksi* data secara otomatis dari aplikasi sumber.
2. Penyaringan atau seleksi data hasil *ekstraksi*.
3. Pengiriman data dari berbagai *platform* aplikasi ke sumber data.
4. Perubahan format *layout* data dari format aslinya.
5. Penyimpanan dalam *file* sementara untuk penggabungan dengan hasil *ekstraksi* dari sumber lain.

b. Tranformation

Pada kenyataannya, pada proses transaksional data disimpan dalam berbagai format sehingga jarang kita temui data yang konsisten antara aplikasi-aplikasi yang ada. Transformasi data ditujukan untuk mengatasi masalah ini. Dengan proses transformasi data ini, kita melakukan standarisasi terhadap data pada satu format yang konsisten. Beberapa contoh ketidak konsistenan data tersebut dapat diakibatkan oleh tipe data yang berbeda, data length dan lain sebagainya.. Langkah-langkah dalam transformasi data adalah sebagai berikut :

1. Memetakan data input dari skema data aslinya ke skema *data warehouse*.
2. Melakukan konversi tipe data atau format data.
3. Pembersihan serta pembuangan duplikasi dan kesalahan data.
4. Penghitungan nilai-nilai derivat atau mula-mula.
5. Penghitungan nilai-nilai agregat atau rangkuman.
6. Pemerikasaan integritas referensi data.
7. Pengisian nilai-nilai kosong dengan nilai default.
8. Penggabungan data.

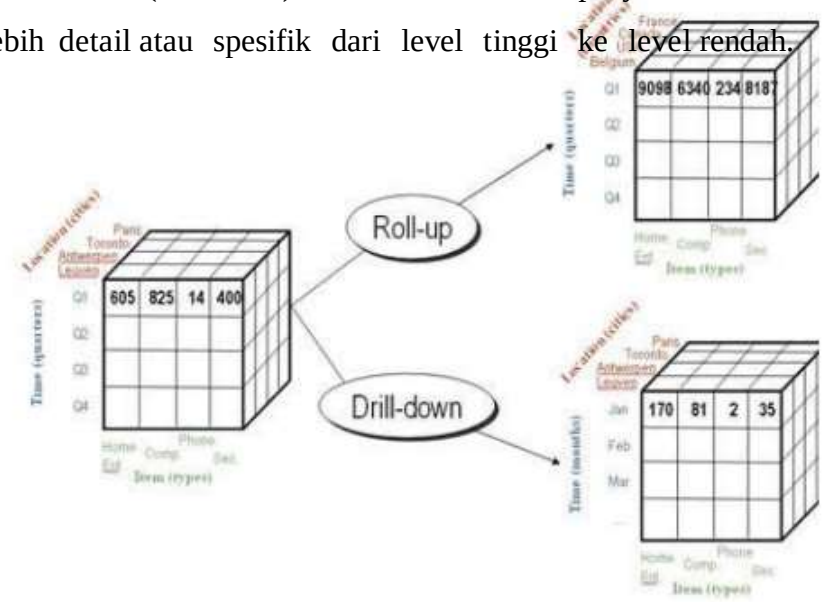
c. Loading

Data load adalah memindahkan data ke data warehouse. Ada dua loading data yang dilakukan pada data warehouse. Pertama adalah inisial load, proses ini dilakukan pada saat telah selesai mendisain dan membangun data warehouse. Data yang di masukan akan sangat besar dan memakan waktu yang relatif lebih lama. Kedua Incremental load, dilakukan ketika data warehouse telah dioperasikan. Incremental load ini dapat dilakukan sesuai dengan sistem yang dibangun.

2.2.2.5 OLAP (*On-Line Analytical Processing*)

OnlineAnalytical Processing (OLAP) terdiri atas seperangkat *tool* untuk membantu proses analisis dan perbandingan data dalam *database*. *Tool* dan metode OLAP membantu pengguna menganalisis data pada sebuah *data warehouse* dengan menyediakan berbagai tampilan data, dan didukung dengan representasi data grafik yang dinamis. [7]

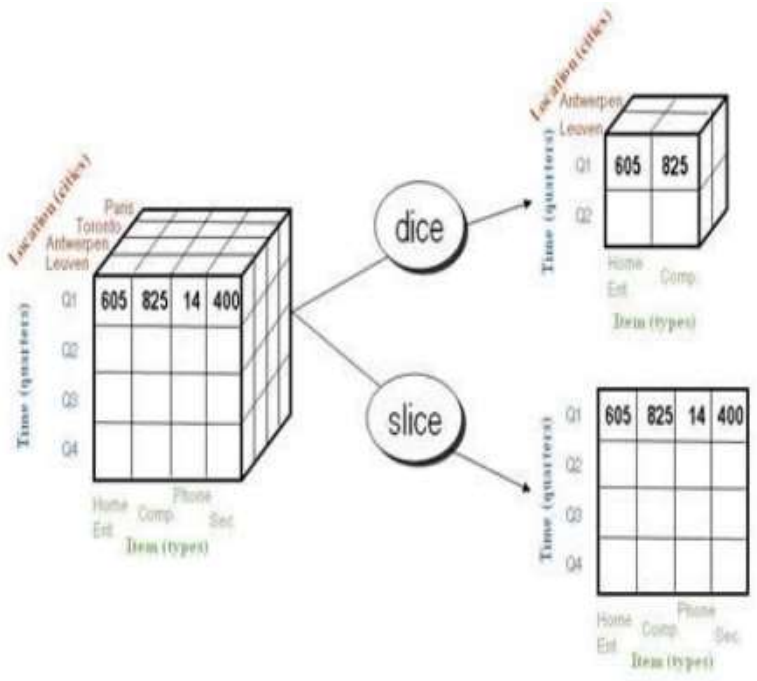
1. Beberapa operasi OLAP yaitu:
   a. *Drill Up (roll-up*) ringkasan data, yaitu dengan menaikkan konsep hirarki atau mereduksi dimensi.
   b. *Drill Down* (*roll-down*) kebalikan dari *roll-up*, yaitu melihat data secara lebih detail atau spesifik dari level tinggi ke level rendah.



Gambar 2.10 Roll-up dan Drill-down

c. Slicing and dicing

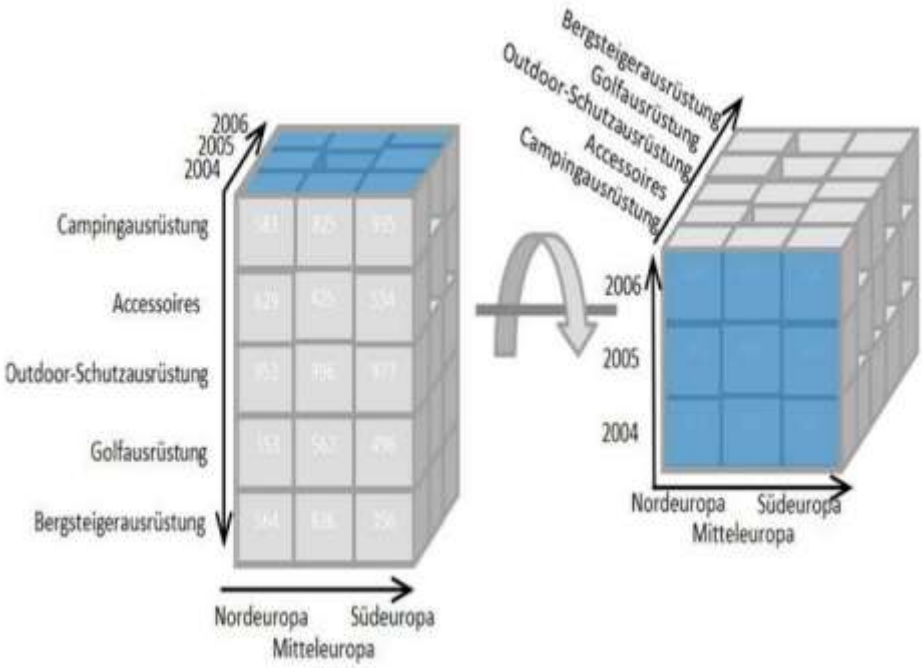
Slicing dan dicing adalah operasi untuk melihat data sebagai visualisasi dari kubus. Dengan slicing dan dicing pengguna dapat melihat data dari beberapa perspektif. Pengguna dapat mengekstrak bagian dari data agregrated dan dapat memeriksa dengan detail berdasarkan dimensi-dimensi yang diinginkan. Data Agregrated merupakan data praperhitungan (precalculated) dalam bentuk rangkuman data (data summarized) sehingga query pada kubus (cube) lebih cepat. Slicing memotong kubus sehingga dapat memfokuskan pada perspektif yang spesifik (pada suatu dimensi). Sedangkan dicing memberikan kemampuan untuk melihat pemilihan data pada dua dimensi atau lebih. Yaitu dengan merotasi cube pada perspektif yang lain sehingga pengguna dapat melihat lebih spesifik terhadap data yang dianalisis. Untuk lebih jelasnya slicing and dicing bisa dilihat pada Gambar 2.11.



Gambar 2.11 Slicing and Dicing

d. Pivot

Menampilkan nilai-nilai ukuran dalam tata letak tabel yang berbeda dan juga bisa mengatur kembali dimensi dalam OLAP cube. Untuk lebih jelasnya pivot bisa dilihat pada Gambar 2.12



Gambar 2.12 *Pivot*

2.2.2.6 Visualisasi Data

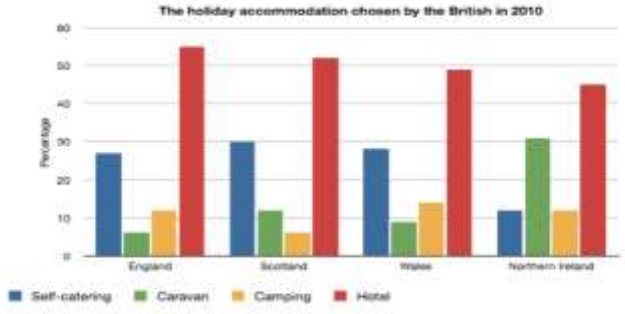
Pengertian Visualisasi adalah rekayasa dalam pembuatan gambar, diagram, atau animasi untuk menampilkan sebuah informasi. Secara umum, visualisasi digambarkan dalam bentuk gambar yang bersifat abstrak maupun nyata. Pada saat ini visualisasi telah berkembang dan banyak dipakai untuk keperluan ilmu pengetahuan, rekayasa, visualisasi desain produk, pendidikan, multimedia interaktif dll. Visualisasi data memiliki tujuan yaitu [8]:

1. Mengeksplor
2. Menghitung
3. Menyampaikan

Data numerik, skema, gambar umum, tabel, dan lainnya dapat divisualisasikan dalam bentuk media dua dimensi non projeksi. Yang biasa digunakan antara lain :

1. *Bar Chart*

   Bar chart sering digunakan untuk menunjukkan data berdasarkan kategori tertentu dimana tidak ada penekanan total presentase pada setiap kategori. *Bar chart* dapat disajikan secara vertikal maupun horizontal. Skala pengukuran adalah nominal atau ordinal. *Bar chart* dapat digunakan untuk menampilkan data kontinu seperti ukuran sepatu atau warna mata dan data diskontinu seperti tinggi badan atau berat badan.



Gambar 2.13 Contoh Bar Chart

2. *Line Chart*

   Line Chart sering digunakan untuk menampilkan informasi dalam rangkaian titik data yang dihubungkan dengan segmen garis lurus. *Line chart* sering digunakan untuk memvisualisasikan trend data dalam interval waktu atau dalam kurun waktu tertentu.

Gambar 2.14 Contoh Line Chart

2.2.3 *Object Oriented Analysis and Design* (OOAD)

OOAD merupakan teknik dan metode dalam mengembangkan sistem yang berbasis pada objek, di mana OOAD menggabungkan data dan proses menjadi

suatu entitas tunggal yang disebut objek. Tujuan dari OOAD adalah membuat elemen sistem yang lebih *reusable, improving system quality*, dan produktivitas terhadap analisis dan desain sistem. *Tools* yang digunakan dalam melakukan pemodelan pada OOAD adalah UML (*Unified Modeling Language).* Berikut adalah diagram yang digunakan dalam UML [6].

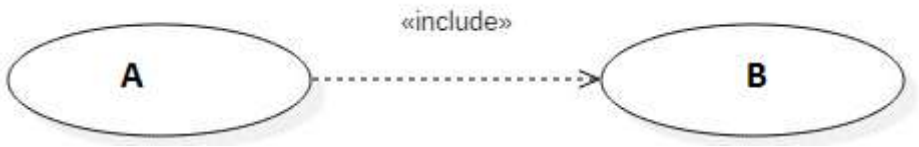
1. *Use Case* Diagram

Use case adalah rangkaian/uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor. Umumnya *use case* digambarkan dengan sebuah elips dengan garis yang solid, biasanya mengandung nama.*Use case* menangkap interaksi yang terjadi antara produsen dan konsumen informasi dan sistem itu sendiri. Pada bagian ini mengkaji bagaimana kasus penggunaan dikembangkan sebagai bagian dari kegiatan modeling persyaratan. Gunakan kasus adalah bagian penting dari analisis pemodelan untuk antarmuka pengguna.*Use case* menggambarkan kebutuhan system dari sudut pandang *User*. Gambar 2.10 Merupakan symbol *use case.*

Relasi-relasi yang berada pada *use case* adalah sebagai berikut :

a. Include

Suatu *use case* bisa meng-*include use case* lainnya. Jika *use case* A melakukan *include* terhadap *use case* B harus diimplementasikan setiap kali *use case* A dipanggil. Direpresentasikan dengan garis panah putus-putus bertuliskan <<include>> kearah *use case* yang di *include*. Contohnya dapat dilihat pada gambar 2.15 sebagai berikut:



Gambar 2.15 Relasi *Include*

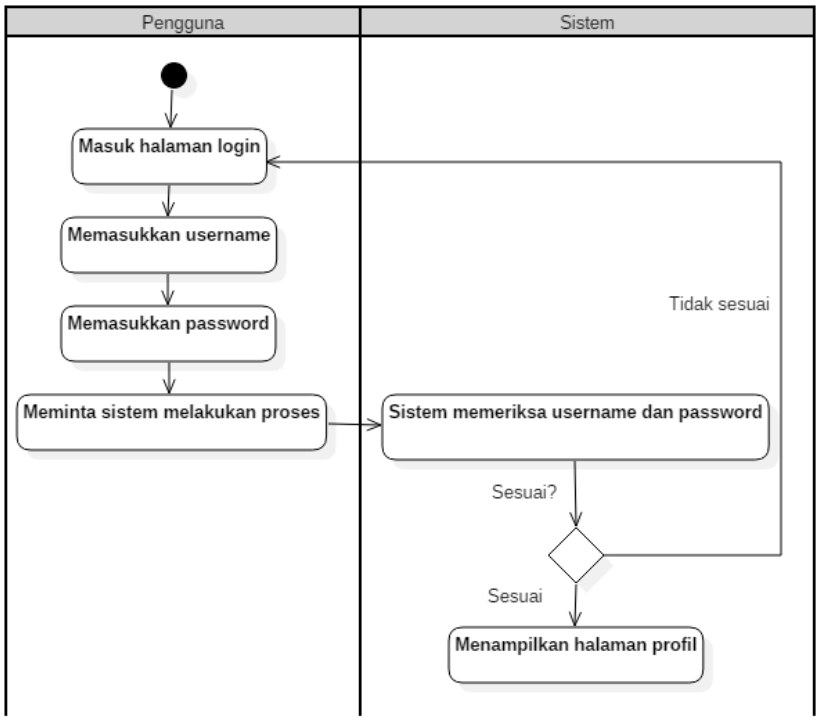
b. Extend

Suatu *use case* bisa di-*extend* oleh *use case* lain. Jika *use case* A di-*extend* oleh *use case* B, maka antara A dan B dapat diimplementasikan dan digunakan secara bebas. Direpresentasikan dengan garis panah putus-putus bertuliskan <<extend>>. A tidak harus selalu memanggil B dalam beberapa batasan. Contohnya dapat dilihat pada gambar 2.16 sebagai berikut:

Gambar 2.16 Relasi *Extend*

2. Activity Diagram

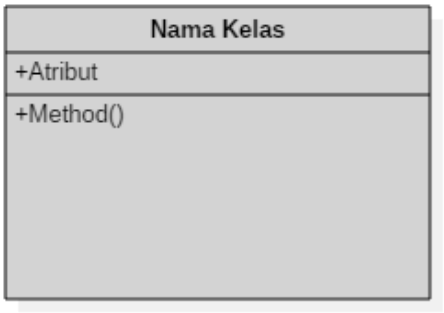
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Activity diagram merupakan state diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (internal processing). Oleh karena itu activity diagram tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Dipakai pada business modeling untuk memperlihatkan urutan aktifitas proses bisnis. Struktur diagram ini mirip *flowchart* atau *Data Flow Diagram* pada perancangan terstruktur. Sangat bermanfaat apabila kita membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan. Activity diagram dibuat berdasarkan sebuah atau beberapa use case pada use case diagram. Berikut gambar 2.17 menjelaskan contoh *activity diagram* :



Gambar 2.17 Contoh *Activity Diagram*

3. Class Diagram

Class diagram atau Diagram Kelas adalah alat perancangan terbaik untuk tim pengembang. Diagram tersebut membantu pengembang mendapatkan struktur system sebelum kode ditulis, dan membantu untuk memastikan bahwa system adalah desain terbaik. Diagram kelas merupakan Sebuah diagram yang memberikan pandangan statis atau struktural dari sebuah sistem. Tidak menunjukkan sifat dinamis dari komunikasi antara objek dari kelas dalam diagram. Unsur-unsur utama dari diagram kelas yaitu kotak, yang merupakan ikon yang digunakan untuk mewakili kelas dan interface. Setiap kotak dibagi menjadi bagian horizontal, bagian atas berisi nama kelas, bagian tengah berisi daftar atribut kelas. Atribut mengacu pada sesuatu yang merupakan objek dari kelas yang tahu atau dapat menyediakan semua waktu. Mereka bisa menjadi nilai-nilai yang kelas dapat menghitung dari variabel misalnya atau nilai-nilai yang kelas dapatkan dari benda-benda lain dari yang terdiri. Bagian ketiga dari diagram kelas berisi operasi atau Method. Operasi mengacu pada apa objek dari kelas yang bisa dilakukan. Gambar 2.18 dihalaman berikutnya merupakan tampilan dari class diagram :



Gambar 2.18 Tampilan *Class Diagram*

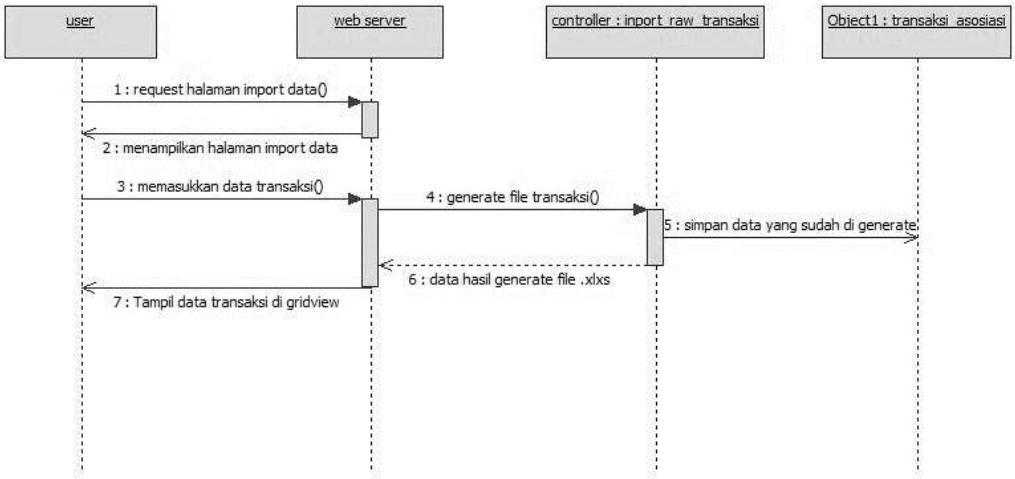
Penjelasannya sebagai berikut:

1. Nama Kelas : Nama kelas haruslah unik, karena ini adalah identitas yang dimiliki oleh setiap Kelas.
2. Atribut : data item menegaskan Kelas.
3. Method : Pelaksanaan prosedur (badan dari kode yang mengeksekusi respon terhadap permintaan objek lain dalam sistem).

4. Diagram Sequence

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan *sequence diagram* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya *sequence diagram* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada *sequence diagram* sehingga semakin banyak *use case* yang didefinisikan maka *sequence diagram* yang harus dibuat juga semakin banyak. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu. Contoh *diagram sequence* dapat dilihat pada gambar 2.19 berikut ini:



Gambar 2.19 ***Contoh Sequence Diagram***