BAB 2

LANDASAN TEORI

2.1 Teori *Data Warehouse*

2.1.1 Definisi *Database*

Menurut Thomas Connolly dan Carolyn Begg (2002, p14), "*Database is a shared collection of logically related data, and a description of this data, designed to meet the information needs if an organization*", yang berarti *database* merupakan kumpulan data yang berhubungan secara logic, dan gambaran data tersebut dirancang untuk memenuhi kebutuhan informasi suatu perusahaan.

Menurut E.G. Mallach (2000, p95), "*Data is what the information system department creates, stores, and provides*", yang berarti data merupakan sesuatu yang diciptakan, disimpan dan disediakan oleh departemen sistem informasi.

Jadi, *database* merupakan kumpulan data yang saling berhubungan yang diciptakan, disimpan, dan disediakan untuk memenuhi kebutuhan informasi suatu perusahaan.

2.1.2 Definisi *Data Warehouse*

Menurut W.H. Inmon (2002, p389), *"A data warehouse is a collection of integrated, subject oriented database designed to support the DSS function, where each unit of data is relevant to some moment in time"* yang berarti *data warehouse* merupakan sekumpulan *database* yang terintegrasi dan berorientasi

subjek yang dirancang untuk mendukung sistem pendukung keputusan dimana masing-masing unit data berkaitan dengan beberapa kejadian pada suatu waktu.

Menurut Vidette Poe (1996, p6), "*A data warehouse is a read-only analytical database that used as the foundation of a decision support system*", yang berarti *data warehouse* merupakan *database* yang bersifat analisis dan hanya dapat dibaca saja, yang digunakan sebagai dasar dari sistem penunjang keputusan.

Menurut Jose Ramalho (2001, p206), *data warehouse* merupakan sebuah *database* yang mengandung data yang biasanya mewakili sejarah bisnis dari suatu 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 sistem *Online Transaction Processing* (OLTP). Menurut Jose Ramalho (2001, p204), *data warehouse* adalah pendekatan untuk penyimpanan data dimana sumber data yang heterogen (yang biasa tersebar pada beberapa *database* OLTP) dan terpisah dimigrasikan untuk penyimpanan data yang homogen.

Menurut Ralph Kimball, "*A data warehouse is a copy of transaction data specifically structured for querying and reporting*." yang berarti *data warehouse* merupakan salinan *(copy)* dari data transaksi yang terstruktur dan digunakan untuk laporan dan *query***.**

Jadi, *data warehouse* merupakan sekumpulan data yang saling berhubungan dan bersifat analisis, yang digunakan sebagai dasar dari sistem penunjang keputusan.

2.1.3 Definisi *Data Mart*

Menurut Thomas Connoly (2002,p1067), *" Data mart is a subset of a data warehouse that support the requirements of a particular department of business function"*, yang berarti *data mart* merupakan *subset* dari *data warehouse* yang mendukung kebutuhan informasi dari suatu departemen atau fungsi bisnis tertentu.

Perbedaan antara *data mart* dengan *data warehouse* adalah :

- *Data mart* hanya berfokus pada kebutuhan *user* yang berkaitan dengan suatu departemen atau fungsi bisnis
- *Data mart* tidak mengandung data operasional secara detil, tidak seperti *data warehouse*
- Data yang ada dalam *data mart* lebih sedikit daripada yang ada dalam *data warehouse*, *data mart* juga lebih mudah dimengerti karena lebih sederhana.

2.1.4 Karakteristik *Data Warehouse*

Menurut W.H. Inmon (2000, p467), "*A subject-oriented, integrated, non-volatile, and time-variant collection of data in support of management's decision.*" Berdasarkan definisi menurut W.H. Immon, karakteristik *data warehouse* adalah sebagai berikut:

2.1.4.1 *Subject oriented* (berorientasi subjek)

Karakteristik pertama dari *data warehouse* yaitu berorientasi pada subjek yang berarti data dikelompokkan berdasarkan fungsi utama dalam bisnis, bukan berdasarkan data transaksi. Contohnya, dalam dunia

perbankan *data warehouse* akan berorientasi pada subjek utama antara lain *customer*, *vendor*, dan *product*, sedangkan OLTP berorientasi pada proses / fungsinya, antara lain tabungan, peminjaman, dan kartu kredit.

2.1.4.2 *Time variant* (variasi waktu)

Data dalam *data warehouse* berhubungan dengan suatu titik atau point dalam suatu periode waktu, dan data dalam *data warehouse* akurat selama periode waktu tertentu, sehingga dapat dikatakan memiliki rentang waktu *(time variant)*. *Data warehouse* juga memiliki tempat untuk penyimpanan data untuk 5 tahun yang lalu atau lebih lama lagi, yang mungkin nantinya dapat digunakan untuk perbandingan *trend* dan *forecasting*.

2.1.4.3 *Integrated* (saling terintegrasi)

Data disimpan sebagai satu kesatuan tunggal, bukan sebagai kumpulan file yang berbeda dalam pengelompokan atau struktur data.

Konsistensi yang ditunjukkan *data warehouse* dapat dilihat pada :

i. *Encoding* (pengkodean)

   Sebagai contoh, *software developer* harus memberi kode “m” untuk jenis kelamin pria, “f” untuk jenis kelamin wanita. Dapat juga memberi kode “1” atau *“male”* untuk pria serta “0” atau *“female”* untuk wanita.

ii. *Attribute measurement* (pengukuran atribut)

   Sebagai contoh, ada beberapa satuan ukuran yang digunakan untuk satuan panjang dalam *database* seperti cm, inchi, meter, dan yard.

Dengan karakteristik integrasi data, maka ukuran tersebut harus konsisten seperti menetapkan ukuran satuan panjang yaitu cm.

iii. *Multiple source* (banyak sumber)

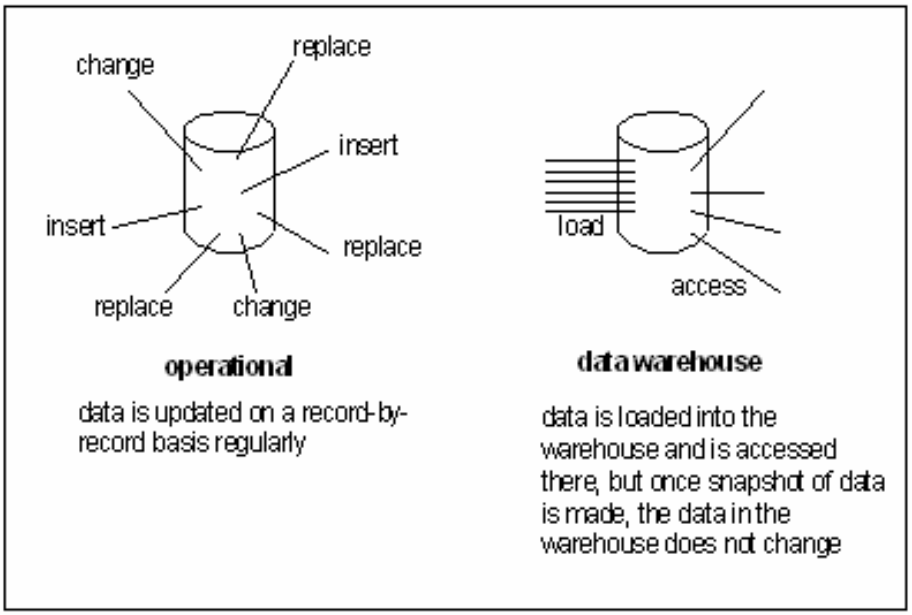
Dalam *database* ada kemungkinan banyaknya deskripsi dari suatu informasi, namun dengan prinsip integrasi data, seluruh informasi tersebut harus memiliki kesamaan deskripsi yang konsisten.

iv. *Conflicting Keys* (kunci yang berbeda)

Sebagai contoh, dalam *database* ada beberapa tipe data yang berbeda dalam *field* yang sama seperti *field* kode barang dalam tabel penjualan memiliki tipe data *character* (char) dengan *field size* 10 sedangkan dalam tabel lain berbeda, misal char (12). Semua perbedaan itu harus diintegrasikan menjadi satu tipe data yaitu char dengan ukuran 12.

2.1.4.4 *Nonvolatile* (tidak berubah-ubah)

Data pada *data warehouse* tidak dapat mengalami perubahan, lain halnya pada *database* operasional dimana dapat dilakukan operasi *update*, *insert*, *delete* terhadap data yang menyebabkan perubahan isi pada *database* namun pada *data warehouse* hanya ada dua kegiatan memanipulasi data yaitu *loading* data (mengambil data yang dibutuhkan *data warehouse*) dan akses data (proses mengakses *data warehouse*, seperti melakukan *query* atau menampilkan laporan yang dibutuhkan), tidak ada kegiatan *updating* data.



Aspek Nonvolatile Data Warehouse
Sumber : http://www.cait/wustl/educait/papers/prism/vol1_no1/

Gambar 2.1 Aspek *Nonvolatile Data Warehouse*

Beberapa perbedaan data operasional dengan *data warehouse* dapat dilihat pada tabel dibawah ini :

Data Operasional	**Data Warehouse**
Berorientasi aplikasi	Berorientasi subjek
Detail	Ringkas
Dapat diupdate	Tidak dapat diupdate
Transaction driven	*Analysis driven*
Strukturnya tetap	Strukturnya fleksibel
Jumlah data yang diproses kecil	Jumlah data yang diproses besar
Non-redudancy	*Redudancy*
Untuk komunitas karyawan	Untuk komunitas manajerial
Current value data	*Historical data*

Tabel 2.1 Perbedaan data operasional dan *data warehouse*

2.1.5 *Granularity*

Salah satu aspek terpenting dalam perancangan *data warehouse* adalah *granularity*. *Granularity* menunjukkan tingkat kerincian atau keringkasan data dalam *data warehouse*. Semakin rinci atau detil suatu data, maka tingkat *granularity*nya semakin rendah, dan juga sebaliknya. Contoh : Sebuah transaksi memiliki tingkat *granularity* yang rendah. Dengan meringkas seluruh transaksi tersebut selama sebulan akan menyebabkan tingkat granularitynya menjadi tinggi.

2.1.6 Struktur *Data Warehouse*

Menurut Vidette Poe (1997, p96-97), *data warehouse* memiliki struktur yang spesifik serta memiliki perbedaan dalam tingkatan ringkasan, detil data, dan umur data. Struktur tersebut terdiri dari:

2.1.6.1 *Current Detail Data* (Detil data saat ini)

Current detail data adalah data detail yang sedang aktif saat ini, mencerminkan keadaan yang sedang berjalan saat ini dan merupakan tingkat terendah dalam *data warehouse*. *Current detail data* ini biasanya memerlukan media penyimpanan data yang cukup besar. Alasan perlu diperhatikannya *current detail data* adalah sebagai berikut:

- Menggambarkan kejadian yang baru terjadi dan selalu menjadi perhatian utama
- Hampir selalu disimpan di media penyimpanan karena diperlukan askes yang cepat tetapi mahal dan kompleks dalam pengaturannya

- Dapat digunakan dalam membuat rekapitulasi sehingga *current detail data* harus akurat
- Jumlahnya sangat banyak dan disimpan pada tingkat penyimpanan terendah.

2.1.6.2 *Old Detail Data* (Detil data historis)

Old detail data merupakan data historis, dapat berupa hasil *back up* yang dapat disimpan dalam media penyimpanan yang terpisah dan dapat diakses kembali pada saat tertentu. Data ini jarang diakses sehingga disimpan dalam media penyimpanan alternatif seperti *tape disk*. Penyusunan direktori untuk data ini harus menggambarkan umur dari data agar memudahkan pengaksesan kembali.

2.1.6.3 *Lightly Summarized Data* (Ringkasan data level menengah)

Lightly summarized data merupakan ringkasan dari *current detail data*. Di dalam tahap ini data belum dapat digunakan untuk pengambilan keputusan karena sifat data belum "*total summary*" yang artinya data masih bersifat detil. *Lightly summarized data* seringkali digunakan sebagai gambaran dari keadaan yang sedang berlangsung maupun yang belum berlangsung.

2.1.6.4 *Highly Summarized Data* (Ringkasan data level tinggi)

Highly summarized data merupakan hasil proses ringkasan yang bersifat "*total summary*". Pada tingkat ini data sangat mudah diakses dan pada akhirnya dapat digunakan sebagai pengambil keputusan bagi para eksekutif perusahaan. Bagi para eksekutif, hal ini sangatlah memudahkan

karena mereka hanya perlu membaca atau melakukan analisis dalam waktu yang singkat.

2.1.6.5 *Metadata*

Metadata bukanlah merupakan hasil kegiatan operasional seperti keempat jenis data diatas, namun merupakan komponen penting dalam *data warehouse*. *Metadata* seringkali disebut ‘data tentang data’. *Metadata* memberikan peranan yang penting untuk keefektifan penggunaan *data warehouse* karena dengan adanya *metadata*, maka akan mempermudah *end user* dalam melakukan analisis dan menghemat waktu. *Metadata* bertindak seperti indeks mengenai isi dari *data warehouse*. *Metadata* mengandung :

i. Struktur data

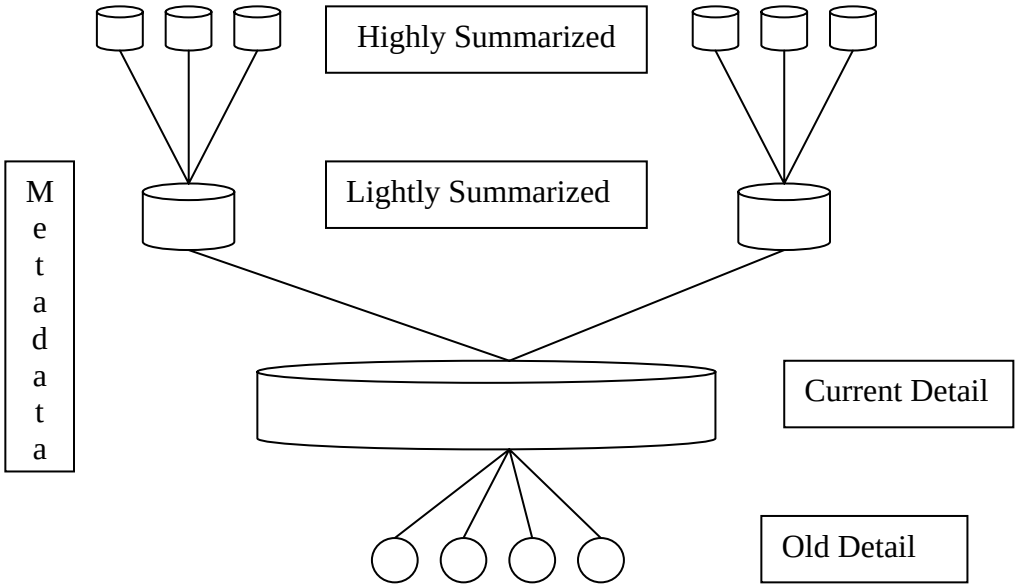
   Suatu direktori untuk membantu analisis DSS *(Decision Support System)* untuk mencari lokasi/tempat data dalam *data warehouse*.

ii. Algoritma untuk meringkas data

   Suatu algoritma untuk proses *summary* data antara *current detail data* dengan *lightly summarized data* dan antara *lightly summarized data* dengan *highly summarized data*, dan lain-lain.

iii. Pemetaan dari data operasional ke *data warehouse*

   Suatu panduan pemetaan data pada saat data ditransformasi/diubah dari lingkup data operasional menjadi lingkup *data warehouse*.



Gambar 2.2 Struktur *Data warehouse*
(Sumber : W.H. Inmon, 2002, p36)

2.1.7 Anatomi *Data Warehouse*

Dalam menentukan bentuk *data warehouse* yang akan digunakan oleh suatu perusahaan, terlebih dahulu harus diketahui kebutuhan yang diperlukan dalam menjalankan aplikasi yang dirancang. Bentuk umum yang sering digunakan dalam *data warehouse* adalah :

2.1.7.1 *Functional Data Warehouse* (*Data Warehouse* Fungsional)

Data warehouse dibuat lebih dari satu dan dikelompokkan berdasarkan masing–masing fungsinya seperti fungsi keuangan *(financial),* fungsi *marketing,* fungsi kinerja personalia dan lain- lainnya. Keuntungan dari bentuk ini adalah dapat dengan mudah dibangun dengan biaya yang relatif murah, sedangkan kerugiannya adalah terbatasnya kemampuan dalam pengumpulan data bagi pengguna. Penerapan jenis

sistem pengumpulan data seperti ini beresiko kehilangan konsistensi data di luar lingkungan bisnis bersangkutan. Apabila pendekatan ini lingkupnya diperbesar dari lingkungan fungsional menjadi lingkup perusahaan, konsistensi data perusahaan tidak lagi dapat dijamin.

2.1.7.2 *Centralized Data Warehouse* (*Data Warehouse* Terpusat)

Bentuk ini terlihat seperti bentuk *functional data warehouse*, namun disini sumber data terlebih dahulu dikumpulkan dan diintegrasikan pada suatu tempat terpusat, kemudian barulah data tersebut dibagi – bagi berdasarkan fungsi – fungsi yang dibutuhkan oleh perusahaan dan bentuk ini sering digunakan oleh perusahaan yang belum memiliki jaringan eksternal. Keuntungan bentuk ini dibandingkan dengan *data warehouse* fungsional adalah bahwa data benar-benar terpadu.

Bentuk ini mengharuskan pemasok data mengirimkan data tepat pada waktunya supaya tetap konsisten dengan pemasok lainnya. Di samping itu, pemakai hanya dapat mengambil data dari bagian pengumpulan saja, dan tidak dapat secara langsung berhubungan dengan pemasok datanya sendiri.

2.1.7.3 *Distributed Data Warehouse* (*Data Warehouse* Terdistribusi)

Bentuk ini dikembangkan berdasarkan konsep *data warehouse gateway* yang memungkinkan pemakai untuk langsung berhubungan dengan sumber data maupun dengan pusat pengumpul data lainnya. Gambaran pemakai atas data adalah berupa gambaran logik karena data mungkin diambil dari berbagai sumber yang berbeda. Pendekatan ini mengandalkan keunggulan teknologi "*client-server*" untuk mengambil

data dari berbagai sumber. Pendekatan ini memungkinkan tiap departemen atau divisi untuk membangun pengumpul data fungsionalnya masing-masing atau bahkan sistem operasionalnya dan memadukan bagian-bagian tersebut dengan teknologi *client-server*. Pendekatan ini memerlukan biaya yang sangat besar karena setiap sistem pengumpulan data fungsional dan sistem operasinya dikelola secara terpisah. Di samping itu, supaya berguna bagi perusahaan data harus disinkronisasikan untuk memelihara keterpaduannya. Metode ini akan sangat efektif apabila data telah tersedia dalam bentuk yang konsisten dan pemakai dapat menambah data tersebut dengan informasi baru apabila ingin memperoleh gambaran baru atas informasi.

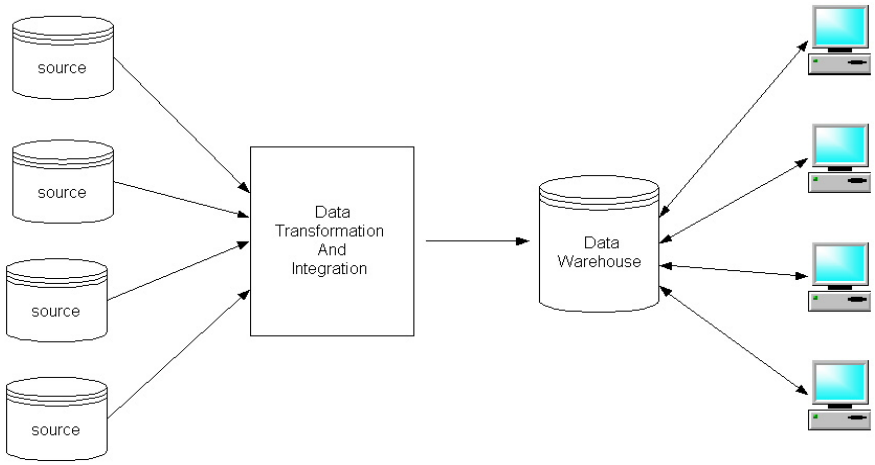
2.1.8 Arsitektur *Data Warehouse*

Menurut Vidette Poe (1996,p24), arsitektur merupakan sekumpulan aturan atau struktur yang memberikan kerangka untuk keseluruhan rancangan suatu sistem atau produk.

Arsitektur data menyediakan kerangka dengan mengidentifikasikan dan memahami bagaimana data akan dipindahkan melalui sistem dan digunakan dalam perusahaan. Arsitektur *data warehouse* mempunyai komponen utama yaitu *database* yang hanya dapat dibaca. Karakteristik arsitektur *data warehouse* (Poe,1996, p40-41) :

- Data diambil dari sistem asal (sistem informasi yang ada), *database* dan file
- Data dari sistem asal diintegrasikan dan ditransformasi sebelum disimpan kedalam *data warehouse*

- *Data warehouse* adalah jenis *database read-only* yang diciptakan untuk mengambil keputusan
- *User* mengakses *data warehouse* via *front-end tool* atau aplikasi.



Gambar 2.3 Arsitektur *Data warehouse*
(Sumber : Poe,1996,p41)

2.2 Teori Perancangan *Data Warehouse*

2.2.1 Perancangan *Data Warehouse*

Menurut Vidette Poe (1996, p120), alat yang digunakan untuk merancang *data warehouse* adalah skema bintang (*star schema*). Skema bintang mempunyai struktur yang sederhana dengan tabel yang relative sedikit dan hubungan antar tabel yang terlihat jelas. Rancangan ini dapat melakukan *query* dengan cepat serta mudah dimengerti oleh *analyst* dan pengguna akhir.

2.2.2 Definisi Skema Bintang

Menurut Thomas Connoly (2002,p1079), "*star schema is a logical structure that has a fact table containing factual data in the center, surrounded by dimension tables containing reference data* (*which can be denormalized*).",

yang berarti skema bintang merupakan struktur *logical* yang memiliki tabel fakta yang berisi data fakta, dikelilingi oleh tabel dimensi yang berisi data referensi (yang dapat didenormalisasikan). Teori Connolly ini didukung oleh A. Silberschatz., Korth, H. F. Sudarshan, S.(2003) dan Ralph Kimball (1996).

2.2.3 Keuntungan Skema Bintang

Keuntungan menggunakan skema bintang menurut Connoly adalah :

- Efisiensi, struktur *database* yang konsisten sehingga lebih efisien dalam mengakses data dengan menggunakan alat/*tool* untuk menampilkan data termasuk laporan tertulis dan *query*
- Kemampuan untuk mengatasi perubahan kebutuhan, skema bintang dapat beradaptasi terhadap perubahan kebutuhan pengguna, karena semua tabel dimensi memiliki kesamaan dalam hal menyediakan akses ke tabel fakta
- *Extensibility,* model dimensional dapat dikembangkan. Seperti menambah tabel fakta selama data masih konsisten, menambah tabel dimensi selama ada nilai tunggal di tabel dimensi tersebut yang mendefinisikan setiap *record* tabel fakta yang ada, menambahkan *attribute* tabel dimensi, dan memecah record tabel dimensi yang ada menjadi level yang lebih rendah dari level sebelumnya
- Kemampuan untuk menggambarkan situasi bisnis pada umumnya, pendekatan standar untuk menangani situasi umum didunia bisnis yang terus bertambah

- Proses *query* yang bisa diprediksi, aplikasi *data warehouse* yang mencari data dari level yang dibawahnya akan dengan mudah menambah jumlah atribut pada tabel dimensi dari sebuah skema bintang. Aplikasi yang mencari data dari level yang setara akan menghubungkan tabel fakta yang terpisah melalui tabel dimensi yang dapat diakses bersama.

2.2.4 Tipe Tabel Skema Bintang

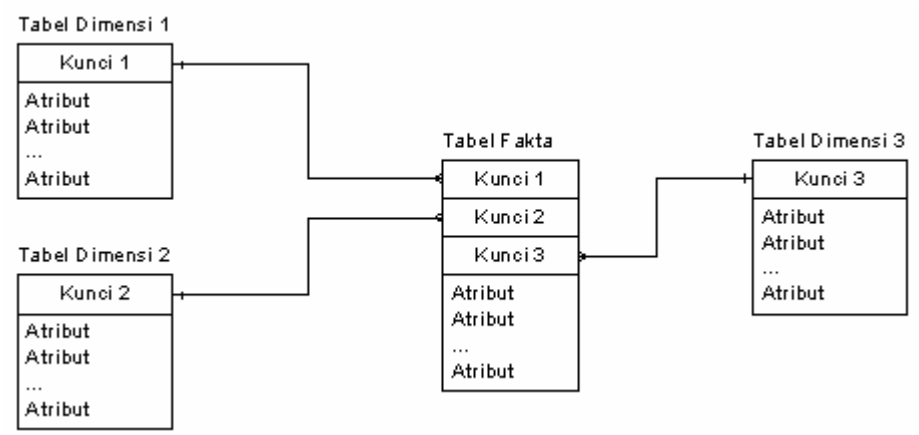
Dalam skema bintang ada dua tipe tabel, yaitu tabel fakta dan tabel dimensi. Tabel fakta dapat disebut juga sebagai tabel *mayor*, terdiri dari data kuantitatif atau data fakta mengenai bisnis, informasi yang di*query*. Infomasi ini selalu diukur secara statistik dan dapat mengandung banyak kolom dan baris. Tabel dimensi disebut juga sebagai tabel *minor*, karena lebih kecil dan mencerminkan dimensi bisnis.

2.2.5 Jenis Skema Bintang

2.2.5.1 Skema Bintang Sederhana

Dalam skema ini, setiap tabel harus memiliki *primary key* yang terdiri dari satu kolom atau lebih dan *primary key* tersebut harus bersifat unik. *Primary key* dari tabel fakta terdiri dari satu atau lebih *foreign key*. *Foreign key* adalah kolom pada satu tabel yang nilainya didefinisikan oleh *primary key* pada tabel yang lain.

Gambar dibawah ini menggambarkan hubungan antara tabel fakta dan tabel dimensi. Tabel fakta memiliki tiga *foreign key*, dimana masing-masing *foreign key* itu merupakan *primary key* pada tabel dimensi.

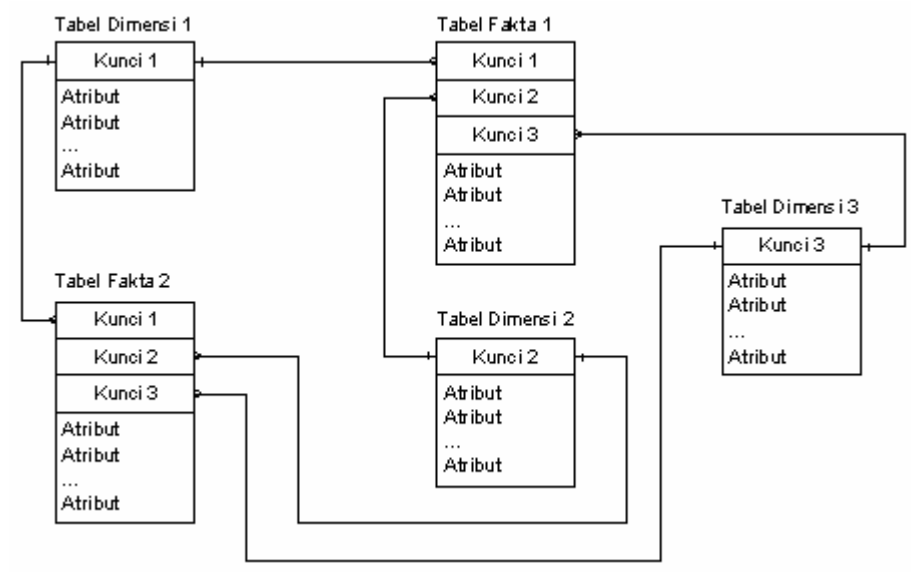


Gambar 2.4 Skema Bintang Sederhana
Sumber : Poe, 1996, p124

Skema Bintang dengan banyak Tabel Fakta

Dalam sebuah skema bintang, dapat juga memiliki lebih dari satu tabel fakta, karena adanya fakta yang tidak saling berhubungan. Misalnya disamping penjualan terdapat tabel fakta *forecasting* dan tabel fakta *result*. Tetapi walaupun terdapat banyak tabel fakta, tabel dimensinya tetap digunakan secara bersama-sama.

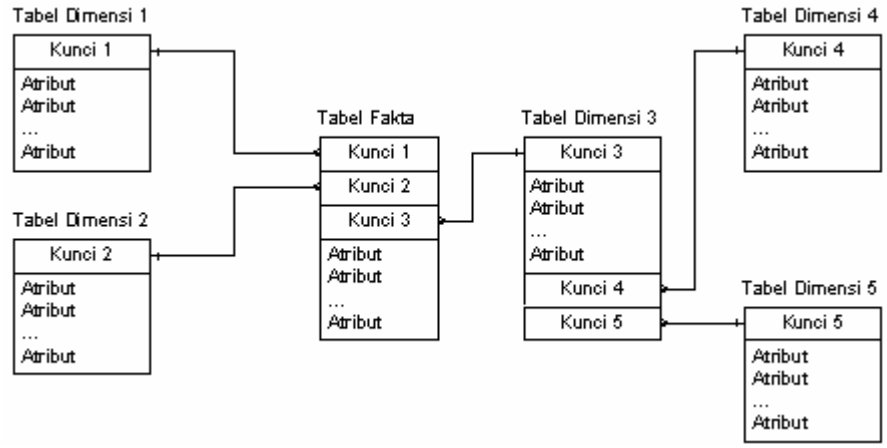
Gambar di bawah ini menunjukkan adanya dua tabel fakta dan tiga tabel dimensi yang memperlihatkan hubungan *many to one* antara *foreign key* pada kedua tabel fakta tersebut dengan *primary key* pada masing masing tabel dimensi.



Gambar 2.5 Skema Bintang Dengan Beberapa Tabel Fakta
Sumber : Poe, 1996, p126

Skema Bintang dengan Tabel Dimensi Tambahan

Tabel dimensi mungkin juga mengandung *foreign key* yang mereferensikan *primary key* di tabel dimensi lain. Tabel dimensi yang direferensikan ini dinamakan *outboard* atau *secondary dimension table.*



Gambar 2.6 Skema Bintang Dengan Tabel Dimensi Tambahan
(Sumber : Poe, 1996, p128)

2.2.5.2 Skema *Snowflake*

Menurut Thomas Connoly (2002, p1080), "*snowflake schema is a variant of the star schema where dimension table do not contain denormalized data.*", yang berarti skema *snowflake* merupakan bentuk lain dari skema bintang dimana tabel dimensi mengandung data yang telah dinormalisasi. Suatu tabel dimensi dapat memiliki tabel dimensi lainnya.

Ciri-ciri *snowflake* adalah:

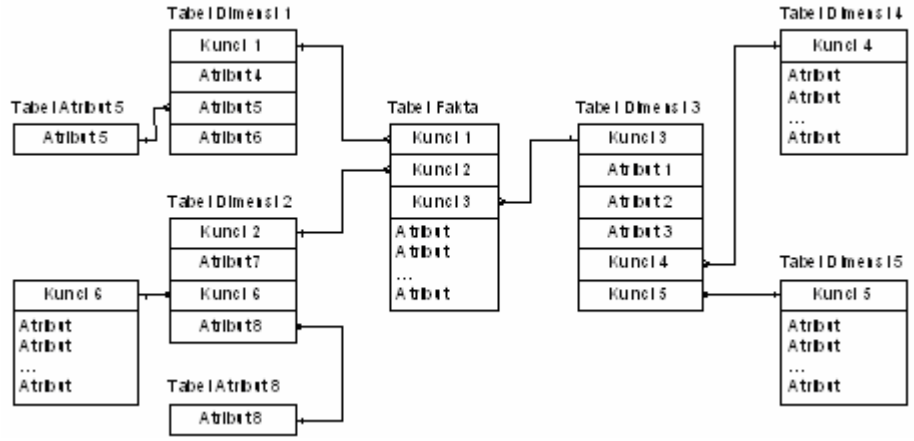
1. Tabel dimensi dinormalisasi dengan dekomposisi pada level *attribute*.
2. Setiap dimensi mempunyai satu *key* untuk setiap level pada hirarki dimensi
3. Kunci level terendah menghubungkan tabel dimensi dengan tabel fakta dan tabel atribut berlevel rendah

Keuntungan dari skema *snowflake* adalah :

- Kecepatan memindahkan data dari data OLTP kedalam *metadata*
- Sebagai kebutuhan dari alat pengambil keputusan tingkat tinggi dimana dengan tipe yang seperti ini seluruh struktur dapat digunakan sepenuhnya
- Banyak yang beranggapan lebih nyaman merancang dalam bentuk normal ketiga.

Sedangkan kerugiannya adalah mempunyai masalah yang besar dalam hal kinerja (*performance*), hal ini disebabkan semakin banyaknya *join*

antar tabel-tabel yang dilakukan dalam skema *snowflake* ini, maka semakin lambat juga kinerja yang dilakukan.



Gambar 2.7 Skema *Snowflake*
Sumber : Poe, 1996, p129

2.2.6 Normalisasi vs Denormalisasi

2.2.6.1 Normalisasi

Menurut Thomas Connolly dan Carolyn Begg (2002, p376), "*Normalisasi is a technique for producing a set of relation with desirable properties, given the data requirements of an enterprise*", yang berarti normalisasi merupakan suatu teknik untuk menghasilkan sekumpulan hubungan dengan properti yang diinginkan, memberikan kebutuhan data dari sebuah perusahaan.

Ada 5 tingkatan normalisasi, tetapi yang paling sering digunakan dalam melakukan normalisasi adalah tiga tingkat pertama dalam normalisasi, sedangkan tingkatan lainnya dapat terjadi tergantung pada data yang ada. Tingkat pertama dalam normalisasi disebut dengan *First Normal Form* (1NF), tingkat kedua disebut dengan *Second Normal Form*

(2NF), dan *Third Normal Form* (3NF), ketiga tingkat normalisasi tersebut berdasarkan pada *functional dependency* antar atribut pada suatu hubungan. Tingkatan selanjutnya adalah *Fourth Normal Form* (4NF) dan *Fifth Normal Form* (5NF), dimana tingkatan pada normalisasi ini sangat jarang terjadi.

First Normal Form (1NF) menyatakan bahwa tabel tidak bisa memuat nilai yang berulang, atau kolom multi nilai. *Second Normal Form* (2NF) menyebutkan bahwa tabel harus berisi entitas tunggal dan setiap kolom *non-primary key* bergantung pada seluruh *primary key*. *Third Normal Form* (3NF) menyatakan bahwa semua kolom *non-primary key* independen, dengan kata lain, sebuah kolom *non-primary key* bisa tidak bergantung pada kolom *non-primary key* lainnya.

Langkah pertama dalam normalisasi *database* adalah memastikan setiap tabel memiliki sebuah kunci utama. Saat menerapkan normalisasi dalam sebuah *database* maka jumlah tabel dan kolom bertambah, yang juga menambah jumlah dan kompleksitas penyertaan (*join*) yang dibutuhkan untuk mengambil data dari *database*. Normalisasi meningkatkan efisiensi dan integritas *database*, khususnya untuk meng-*update* data.

2.2.6.2 Denormalisasi

Normalisasi memang dapat meningkatkan efisiensi dan integritas suatu *database*, namun ada saat-saat tertentu perlu dilakukan denormalisasi untuk meningkatkan kinerja *query*. Denormalisasi

dibutuhkan ketika jumlah penyertaan (*join*) relasional yang diperlukan untuk mengambil informasi memerlukan waktu yang banyak.

Dalam melakukan denormalisasi penyimpanan data dalam *database* akan melanggar ketentuan dalam normalisasi terutama *Third Normal Form* (3NF) yang bertujuan untuk menghilangkan redudansi data. Namun jika normalisasi menghabiskan waktu dalam memberikan suatu informasi dari tabel yang diinginkan akan lebih efisien jika disimpan dalam sebuah tabel.

2.3 *Performance Tuning*

Performance tuning merupakan teknik yang dilakukan untuk meningkatkan kinerja *hardware* maupun *software*. Bukanlah hal yang mudah untuk melakukan *performance tuning*, karena banyaknya faktor yang harus diperhatikan dan tidak semua faktor itu saling mendukung satu sama lain. Contoh : seorang DBA menginginkan data tersimpan dengan lengkap dan dapat diakses dengan cepat, sedangkan pada kenyataannya semakin banyak data yang tersimpan maka pengaksesan akan semakin lambat.*Performance tuning* juga sangat diperlukan dalam *data warehouse* karena data pada *data warehouse* semakin lama semakin bertambah, sehingga proses pengaksesan dan pencarian menjadi lambat. Ada beberapa strategi yang digunakan dalam melakukan *performance tuning*, antara lain agregasi, pemberian index, dan partisi. Strategi inilah yang dapat digunakan untuk mengatasi masalah *data warehouse* tersebut.

2.4 OLAP (*On – Line Analytical Processing*)

Menurut E.G. Mallach (2000, p531), OLAP adalah *" A category of software that enables analyst, managers, and executive to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information that has been transformed from raw data to reflect the real dimensionality of the enterprise as understood by the user"*, yang berarti OLAP merupakan kategori *software* yang memungkinkan seorang *system analyst*, manager, dan eksekutif untuk memperoleh data dengan akses yang cepat, konsisten dan interaktif sehingga informasi dapat dilihat dari berbagai sudut pandang dimana data telah ditranformasikan dari data mentah untuk menggambarkan dimensi perusahaan yang dapat dimengerti oleh *user*.

Keuntungan dari aplikasi OLAP, antara lain :

- Meningkatkan produktivitas bisnis dan *user*, pegembang IT, seluruh organisasi secara konsekuen
- Meningkatkan pendapatan dan keuntungan dengan memperbolehkan organisasi memberikan respon lebih cepat terhadap permintaan pasar
- Mengurangi pemindahan *query* dan lalu lintas jaringan pada sistem OLTP ataupun pada *data warehouse*

2.5 Metodologi Perancangan *Data Warehouse*

Metode yang digunakan dalam penulisan skripsi ini meliputi :

1. *Requirement Collection and Analysis*

i. Menentukan *owner* dan *user (stakeholder)*, pada tahap ini dilakukan penentuan siapa yang berperan sebagai *owner* dan *user*. *User* yang dimaksud yaitu *user* yang menggunakan sistem dan *user* yang memaintain sistem.

ii. *Fact Finding*, merupakan teknik yang digunakan untuk mengumpulkan fakta atau informasi. Teknik-teknik itu antara lain:

- *Interview*, teknik ini dilakukan dengan memberikan beberapa pertanyaan yang berhubungan dengan topik yang dibicarakan. Pertanyaannya ditujukan hanya pada satu orang
- *Questionare*, teknik ini dilakukan dengan memberikan angket yang berisi pertanyaan-pertanyaan yang berhubungan dengan topik. *Questionare* ini diberikan kepada sejumlah orang (±100) dengan jawaban yang beragam
- *Document Examination*, teknik ini dilakukan dengan menguji semua dokumen-dokumen yang berkaitan. Jika ada hal-hal yang ambigu maka dapat langsung ditanyakan kepada pengguna dokumen
- *Joint Requirement Planning*, teknik ini dilakukan dengan mengumpulkan orang-orang (±10) untuk berdiskusi secara langsung.

2. Perancangan *Logical*, terdiri dari :

i. Penentuan *Grain*

Dengan menentukan *grain*, berarti secara langsung juga menentukan apa yang akan ditampilkan pada tiap *record* dari tabel fakta. Ketika *grain* untuk tabel fakta telah dipilih, maka dapat diidentifikasikan dimensi untuk tabel fakta tersebut.

ii. Penentuan *Measure*

Measure merupakan suatu nilai yang ingin dihitung atau diukur dan tipe datanya harus angka *(numeric/currency).*

iii. Penentuan Dimensi

Dimensi merupakan sudut pandang *user* terhadap data dan tergantung kebutuhan masing-masing *user*.

iv. Rancangan Skema Bintang

Skema bintang merupakan struktur *logical* yang memiliki tabel fakta yang berisi data fakta, dikelilingi oleh tabel dimensi yang berisi data referensi (yang dapat didenormalisasikan).

v. Pemetaan *data warehouse* dan OLTP

Kegiatan memetakan atribut-atribut tabel *data warehouse* dari atribut-atribut tabel OLTP.

3. Pemilihan DBMS

Pada tahap ini dilakukan pemilihan bahasa pemrograman untuk menciptakan *database* baru untuk *data warehouse*. Misalnya : *Microsoft SQL Server, MySQL, Oracle, Microsoft Access.*

4. Perancangan *Physical*, terdiri dari :

a. Perancangan *base table*

b. Perancangan representasi dari *derived data*

c. Analisis transaksi

d. Pemilihan file organisasi

e. Pemilihan index

f. Analisis pertumbuhan data dan kapasitas media penyimpanan

g. Perancangan *user view*

h. Mekanisme keamanan

2.6 Teori Persediaan dan Penjualan

2.6.1 Teori Persediaan

Persediaan menurut Mulyadi (1997, p555) terbagi menjadi persediaan produk jadi, persediaan produk dalam proses, persediaan bahan baku, persediaan bahan penolong, persediaan bahan habis pakai pabrik, dan persediaan suku cadang.

Persediaan merupakan proses yang penting untuk menghitung dan mengontrol produk-produk yang tersimpan. Transaksi persediaan yang terjadi di gudang, biasanya mencakup :

- Penerimaan produk
- Penempatan produk di bagian pemeriksaan
- Pengambilan produk dari bagian pemeriksaan
- Pengembalian produk kepada *vendor* jika produk tidak sesuai dengan yang diharapkan
- Penempatan produk pada tempat penyimpanannya
- Pengesahan produk untuk dijual
- Pengambilan produk dari tempat penyimpanannya
- Pengepakan produk untuk pengiriman
- Pengiriman produk ke pelanggan
- Penerimaan produk dari pelanggan

- Pengembalian produk ke bagian persediaan dari pengembalian pelanggan
- Produk yang dikembalikan pelanggan dipisahkan dari bagian persediaan

2.6.2 Teori Penjualan

Proses penjualan merupakan rangkaian kegiatan operasi yang melayani pelanggan, membantu pelanggan memilih produk dan jasa, mengirim produk dan jasa yang diminta pelanggan dan menagih pembayaran untuk produk dan jasa, secara bersama-sama. Dan juga proses tersebut harus dapat :

- Meminimalkan waktu antara penjualan yang terjadi dengan pembayaran yang diterima
- Meminimalkan piutang tak tertagih
- Menjaga keseimbangan antara kualitas produk dengan harga

Proses penjualan seringkali terjadi hanya pada saat pelanggan berinteraksi langsung dengan organisasi. Seefisien apapun proses organisasi yang ada, jika proses penjualan tidak berfungsi dengan baik, tidak mungkin suatu organisasi dapat menghasilkan pendapatan yang cukup untuk bertahan Menghasilkan pendapatan merupakan kunci untuk berkembang dan memperoleh keuntungan. Suatu organisasi dapat menghasilkan produk dalam jumlah yang banyak dan memberikan berbagai jenis jasa, akan tetapi yang terpenting apakah semua itu bisa menutup biaya yang sudah dikeluarkan, dan mengembalikan modal yang sudah dikeluarkan. Jika suatu organisasi menjual produk dan jasa dengan kualitas yang rendah maka kemampuan perusahaan untuk menarik pelanggannya sangat diragukan.