BAB III

DATA WAREHOUSE

3.1 *Data Warehouse*

Para pengambil keputusan membutuhkan akses ke seluruh data perusahaan, kemanapun data-data itu ditempatkan. Untuk menyediakan analisis yang meliputi banyak hal dari suatu perusahaan, baik mengenai proses bisnis, kebutuhan, dan tren yang sedang terjadi, perusahaan membutuhkan akses tidak hanya ke dalam nilai *current* pada *database* tetapi juga data historikal. Data tersebut semestinya juga dipersiapkan, diorganisasikan dan ditampilkan kepada pengguna sehingga data tersebut dapat digunakan secara optimal oleh pengguna. Kemudahan yang ditawarkan untuk menangani tipe analisis seperti ini adalah terciptanya *Data Warehouse* untuk menyimpan data yang diperoleh dari beberapa sumber data yang berbeda dan dipelihara dengan unit operasi yang berbeda pula, mencakup transformasi data historikal dan data ringkasan. Data warehouse didasarkan pada teknologi *database* yang secara luas menyediakan pengelolaan terhadap penyimpanan data.

Teknologi *Data Warehouse* terdiri dari seperangkat konsep baru dan program bantu (*tool*) yang mendukung pekerja pengetahuan (eksekutif, manajer, analis) dalam hal pengaturan, pemahaman, penggunaan, penganalisaan data, dan penyediaan informasi secara cepat untuk membantu mendapatkan keputusan yang strategis dalam suatu aplikasi bisnis atau organisasi. Bab ini akan membahas detail *Data Warehouse* secara lebih mendalam.

3.1.1 Definisi *Data Warehouse*

Bill Inmon yang disebut sebagai "Bapak *Data Warehouse*" mendefinisikan *data warehouse* sebagai sekumpulan data yang berorientasi subjek, terintegrasi, *time-variant*, dan *non-volatile* dalam rangka mendukung proses pengambilan keputusan manajemen. Beberapa definisi lain dari *data warehouse* adalah sebagai berikut:

a. Barry Devlin dari IBM menyatakan *Data Warehouse* sebagai pusat koleksi yg lengkap dan konsisten data perusahaan yang dikumpulkan dari berbagai sumber untuk memberikan informasi yang dapat dipahami dan dipergunakan dalam konteks bisnis.
b. Vivek Gupta dari System Services Corp mendefinisikan *Data Warehouse* sebagai sistem terstruktur berskala besar untuk menganalisa data statis yang telah ditransformasikan dari berbagai aplikasi asalnya agar sesuai dgn struktur bisnis, terkumpul dlm waktu yang lama, direpresentasikan dalam terminologi bisnis, dan terangkum untuk memudahkan analisa.

Ada berbagai definisi dari *data warehouse*, dimana semua definisi awal tersebut terfokus pada karakteristik data yang terdapat dalam *data warehouse*. Definisi lain yang lingkupnya lebih luas meliputi pengolahan yang berhubungan dengan akses data dari sumber asli untuk pengiriman data kepada para pengambil keputusan (Anahory & Murray, 1997). Apapun definisinya, tujuan utama dari *data warehouse* adalah untuk

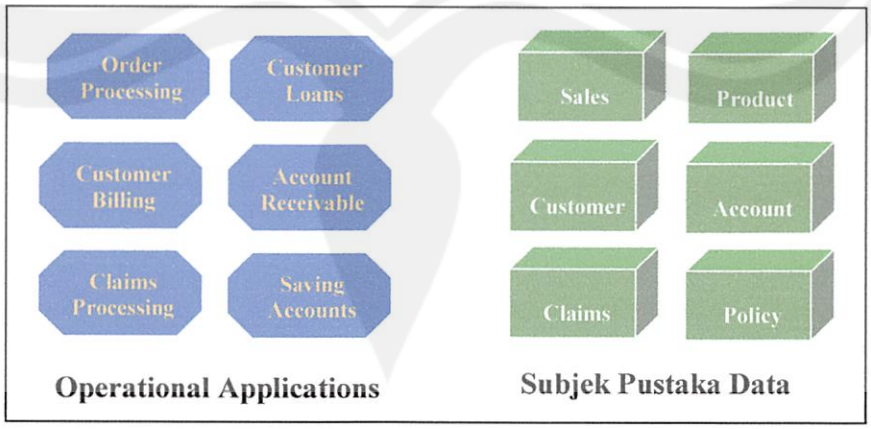
mengintegrasikan bermacam-macam data perusahaan ke dalam satu tempat penyimpanan yang akan memudahkan pengguna menjalankan query, membuat laporan, dan melakukan analisis. Pada dasarnya, *data warehouse* dapat dikatakan sebagai sebuah teknologi manajemen data dan analisis data (Connoly & Begg, 2005).

3.2 Karakteristik Data pada *Data Warehouse*

Empat karakteristik utama yang ada pada *data warehouse*, yaitu:

1. *Subject Oriented*

   Data warehouse diorganisasikan berdasarkan pada area subjek utama perusahaan (misal: pelanggan, produk, dan penjualan), bukan berdasarkan pada area aplikasi operasional. *Data warehouse* dipusatkan pada pemodelan dan analisis data untuk pembuatan keputusan, bukan untuk operasi harian atau pemrosesan transaksi. Maka, *data warehouse* secara khusus menyediakan *view* yang ringkas dan sederhana di sekitar subjek pembicaraan dengan memisahkan data-data yang tidak berkaitan dengan proses penunjang keputusan.

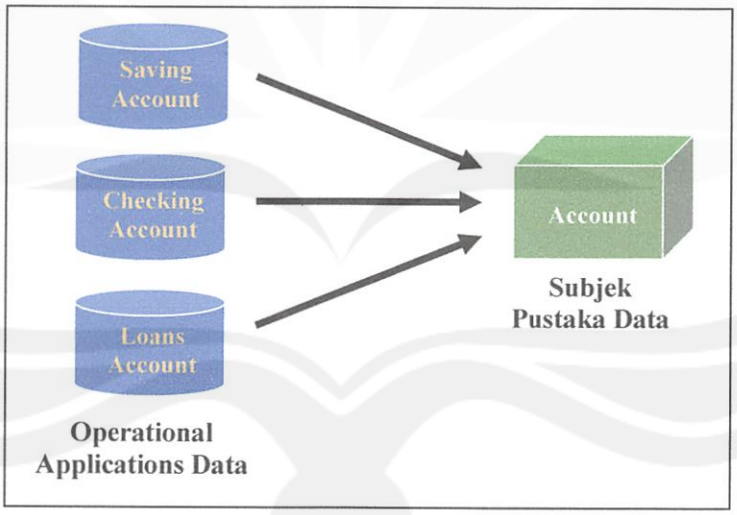


Gambar 3.1 Masalah *Subject Oriented*

2. *Integrated*

Data pada *data warehouse* merupakan kumpulan bermacam-macam sumber data dari berbagai sistem aplikasi perusahaan yang berbeda. Data sumber sering tidak konsisten, sebagai contoh satu atribut dapat disimpan dalam format yang berbeda. Sumber data yang terintegrasi harus dibuat secara konsisten untuk memberikan suatu gambaran data yang terpadu kepada pengguna.

Teknik pembersihan dan integrasi data digunakan untuk memastikan konsistensi dalam penamaan, struktur penyandian, ukuran atribut dan sebagainya, diantara sumber-sumber yang berbeda. Data atas suatu subjek tertentu didefinisikan dan disimpan sekali.



Gambar 3.2 Masalah *Integrated*

3. *Time Variant*

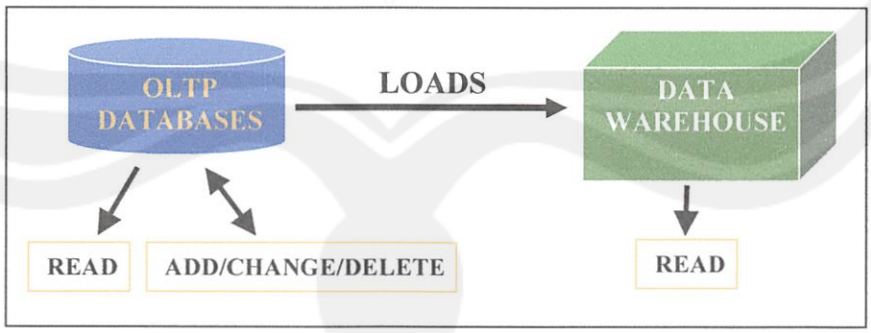
Horison waktu untuk *data warehouse* secara signifikan lebih lama dari pada sistem operasional. Data dari *data warehouse* menyediakan informasi dari suatu tinjauan historis (misalkan,

5-10 tahun terakhir). Setiap struktur *key* di dalam data dari *data warehouse* memuat suatu elemen waktu, baik itu secara eksplisit dinyatakan maupun implisit.

Tetapi setiap struktur *key* dari data operasional bisa memuat atau bisa juga tidak memuat elemen waktu. Data disimpan dalam sederetan *snapshot*, yang masing-masing menggambarkan suatu periode waktu. Fokus dari *data warehouse* atas perubahan dari waktu ke waktu adalah apa yang dimaksud dengan istilah *time variant* (perbedaan waktu).

4. *Non Volatile*

Data tidak di-*update* dalam waktu riil tetapi di-*refresh* dari sistem operasional secara teratur. Data baru selalu ditambahkan sebagai tambahan pada *database* dan bukan sebagai data pengganti. Secara terus menerus, *database* menampung data baru tersebut dan mengintegrasikannya dengan data sebelumnya.



Gambar 3.3 Masalah *Non Volatile*

5. Granularitas

Granularitas menunjuk pada level perincian atau peringkasan yang ada pada unit-unit data dalam *data warehouse*. Semakin banyak detail yang

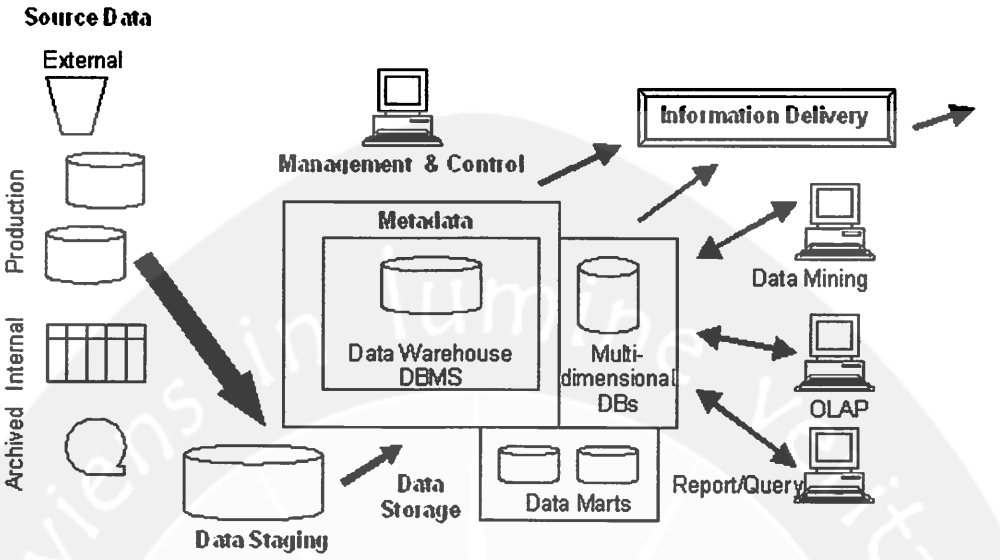
ada, maka semakin rendah level granularitas. Semakin sedikit detail yang ada, maka semakin tinggi level granularitas.

Granularitas data selalu menjadi masalah perancangan. Ketika *detailed data* di-*update*, data akan disimpan pada level granularitas terendah. Alasan mengapa granularitas merupakan masalah utama perancangan pada *data warehouse* adalah bahwa granularitas sangat mempengaruhi volume data yang terletak pada *data warehouse* dan pada waktu yang sama berpengaruh pada tipe *query* yang dapat dijawab.

Semakin tinggi level granularitas maka *query* yang dapat ditangani oleh *data warehouse* semakin terbatas. Semakin rendah level granularitas maka *query* yang dapat ditangani oleh *data warehouse* semakin banyak dan jawaban *query* yang diperolehpun semakin detail.

3.3 Komponen *Data Warehouse*

Komponen dasar dari *Data Warehouse* adalah *source data*, *data staging*, dan *data storage* yang akan mengatur data dari *Data Warehouse*. Komponen ini tidak hanya menyimpan dan mengatur data namun juga menjaga bagian data yang disebut *metadata repository*. Komponen *Information Delivery* terdiri dari seluruh cara yang berbeda untuk menyediakan informasi dari *Data Warehouse* yang tersedia bagi *user*. Bagaimanapun sebuah *Data Warehouse* dibangun akan memiliki *building block* yang sama. Berikut ini adalah gambar mengenai komponen-komponen *Data Warehouse*:



Gambar 3.4 Building Blocks atau Komponen Data Warehouse (Ponniah, 2001)

3.3.1 Komponen *Source Data* (Sumber Data)

Source data yang digunakan dalam *data warehouse* dapat dikelompokkan menjadi empat kategori, yaitu:

a. *Production Data*

Kategori data ini berasal dari berbagai macam sistem operasional dalam perusahaan. Berdasarkan kebutuhan informasi di dalam *data warehouse*, segmen-segmen data dipilih dari sistem operasional yang berbeda. Dalam proses ini, data-data yang ditangani kemungkinan besar berada dalam format yang bermacam-macam, kemungkinan juga berasal dari platform yang berbeda-beda. Istilah seperti sebuah *account* dapat memiliki arti yang berbeda dalam sistem yang berbeda. Lebih lanjut lagi, data-data tersebut didukung oleh sistem basis data dan sistem operasi yang berbeda-beda.

Karakteristik *production data* yang paling signifikan dan mengganggu adalah perbedaan (*disparity*). Tantangan terbesar adalah untuk membuat standarisasi dan mentransformasi data-data yang berlainan dari berbagai sistem operasional yang berbeda, mengkonversi data, dan mengintegrasikannya menjadi beberapa bagian yang kemudian akan disimpan dalam *data warehouse*.

b. *Internal Data*

Dalam setiap organisasi, masing-masing user menjaga dokumen-dokumen, *spreadsheets*, profil-profil pelanggan mereka, bahkan basis data lengkap berdasarkan departemennya. Semua hal tersebut merupakan data internal, bagian yang dapat menjadi berguna dalam sebuah *data warehouse*.

Jika suatu organisasi menjalankan bisnis dengan pelanggannya dalam basis *one-to-one* dan kontribusi dari tiap-tiap pelanggan di garis bawah adalah signifikan, maka detail profil dari pelanggan dengan demografis yang luas merupakan hal penting dalam *data warehouse*. Profil dari pelanggan menjadi sangat penting sebagai pertimbangan ketika departemen pemasaran ingin melakukan penawaran khusus pada pelanggan. Meskipun mungkin banyak dari data ini didapatkan dari sistem produksi, tetapi sebagian besar data tersebut dipegang oleh individu dan departemen dalam *file-file* pribadi mereka. Data-data internal yang dipegang dalam *file* pribadi di dalam organisasi seperti ini tidak dapat diabaikan.

Data internal menambah kompleksitas dalam proses transformasi dan integrasi data sebelum dapat

disimpan ke dalam *data warehouse*. Strategi untuk mengambil data kerja (*spreadsheets*), cara untuk mengambil data dari dokumen-dokumen tekstual dan menghubungkannya ke dalam basis data-basis data deparmental harus ditentukan untuk mengumpulkan data yang saling berhubungan dari sumber-sumber tersebut.

c. *Archived Data*

Sistem operasional terutama ditujukan untuk menjalankan suatu bisnis. Dalam setiap sistem operasional, proses yang dilakukan secara berkala adalah mengambil data lama dan menyimpannya dalam *file* arsip. Biasanya suatu data akan menjadi arsip setelah satu tahun, namun terkadang data disimpan pada data operasional sepanjang lima tahun.

Sebuah *data warehouse* menyimpan data-data historikal. Pada dasarnya, data-data historikal ini sangat dibutuhkan untuk melakukan analisis, dan untuk melihat data-data tersebut, cukup dengan melihatnya melalui *data set-data set* yang telah diarsipkan.

d. *External Data*

Banyak para eksekutif menggunakan data yang berasal dari sumber-sumber eksternal untuk meningkatkan persentase informasi yang mereka gunakan. Mereka menggunakan statistik-statistik yang berhubungan dengan industri mereka, yang dihasilkan oleh agensi-agensi eksternal. Mereka menggunakan data-data *market share* dari para kompetitor, serta nilai-nilai standar dari indikator-indikator keuangan terhadap bisnis mereka untuk mengetahui *performance* perusahaan.

Data eksternal dapat memberikan gambaran mengenai apa yang akan dilakukan atau apa yang telah dilakukan oleh suatu perusahaan. Data eksternal ini sangat dibutuhkan apabila suatu perusahaan ingin membandingkan perusahaannya dengan organisasi lain. Hal seperti inilah yang dapat meningkatkan kinerja perusahaan.

3.3.2 Komponen *Data Staging* (Penyiapan Data)

Setelah melakukan pengekstrakan data dari sistem operasional dan sumber eksternal, maka selanjutnya harus dilakukan pengolahan data untuk menyimpannya dalam *data warehouse*. Ada tiga fungsi utama yang harus dilakukan untuk menyiapkan data. Fungsi tersebut adalah pengekstrakan data, transformasi data, dan me-*load* data ke dalam penyimpanan *data warehouse*. Tahap pembersihan ini dikenal juga dengan istilah ETL (*Extraction, Transformation and Loading*). Tiga fungsi utama ini berlangsung di sebuah *staging area*.

Data staging menyiapkan suatu tempat dengan satu set fungsi untuk membersihkan, mengubah, menggabungkan, mengkonversi, menyalin, dan menyiapkan data sumber untuk disimpan dan digunakan dalam *data warehouse*. *Data staging* sangat diperlukan dalam *data warehouse* karena data-data yang berasal dari berbagai sumber yang berbeda tidak dapat langsung dipindahkan ke dalam *data warehouse*. Hal ini disebabkan karena data dalam *data warehouse* adalah *subject oriented* dan merupakan perpaduan dari berbagai aplikasi operasional. Di bawah ini akan dijelaskan lebih detail lagi mengenai tiga fungsi utama di *staging area*:

a. *Data Extraction*

Fungsi ini berhubungan dengan sumber data yang banyak. Karena itu diperlukan penerapan teknik yang tepat untuk tiap-tiap data sumber. Data sumber mungkin berasal dari mesin-mesin yang berbeda dan dalam format-format yang berlainan. Sebagian besar sumber data mungkin berada dalam sistem-sistem basis data relasional. Beberapa data mungkin berada dalam *legacy network* dan model-model data hirarkial. Banyak data-data mungkin masih berada dalam bentuk flat files. Bisa juga data yang diinginkan berasal dari *spreadsheets* dan data set-data set departemen lokal. Jadi, ekstraksi data bisa menjadi benar-benar kompleks.

Biasanya dilakukan pengekstrakan data ke dalam lingkungan fisik yang terpisah, yang membuat pemindahan data tersebut ke dalam *data warehouse* menjadi lebih mudah. Pada lingkungan yang terpisah, sumber data dapat diekstraksi ke dalam sebuah grup *flat file*, atau sebuah basis data relasional *data staging*, atau kombinasi dari keduanya.

b. *Data Transformation*

Setelah melalui tahap ekstraksi, data tersebut masih merupakan data mentah dan tidak dapat diaplikasikan ke *data warehouse*. Pertama-tama semua data hasil ekstraksi tersebut harus dibuat berguna dalam *data warehouse*. Mengolah informasi agar dapat digunakan untuk membuat keputusan strategis adalah prinsip utama dari

data warehouse. Karena data-data operasional didapatkan dari banyak sistem lama, kualitas data pada sistem-sistem tersebut menjadi kurang baik untuk *data warehouse*. Kualitas data harus diperkaya dan dikembangkan sebelum dapat digunakan dalam *data warehouse*.

Sebelum memindahkan data yang telah diekstrak dari *source systems* ke dalam *data warehouse*, diperlukan beberapa macam transformasi data. Data harus ditransformasikan menurut standar-standar yang telah ditetapkan, karena data-data tersebut berasal dari *source systems* yang tidak sama. Harus dipastikan bahwa setelah data-data disatukan, kombinasi data tidak akan melanggar aturan-aturan bisnis (*business rule*).

Usaha yang besar dalam transformasi data adalah peningkatan kualitas data. Secara sederhana, mencakup antara lain pengisian nilai-nilai yang hilang dari atribut-atribut dalam data yang telah diekstrak. Kualitas data adalah hal yang paling penting dalam *data warehouse*, karena akibat yang ditimbulkan dari keputusan-keputusan strategis yang diambil berdasarkan pada informasi yang salah dapat sangat merugikan.

Tugas-tugas dasar transformasi antara lain adalah:

Tabel 3.1 Tugas-Tugas Dasar Transformasi Data

TUGAS	DESKRIPSI
Selection	Terletak di awal proses transformasi data. Yaitu memilih seluruh atau sebagian dari beberapa *record* dari *source systems*.
Splitting/joining	Meliputi manipulasi data yang diperlukan untuk bagian-bagian *record* hasil operasi *selection*. Kadang-kadang data akan dipisahkan (*split*) atau akan digabungkan (*join*) dalam proses transformasi data.
Conversion	Meliputi konversi dari sebuah *field*, untuk dua alasan utama. Pertama untuk standarisasi data-data hasil ekstraksi dari *source systems* yang berbeda-beda, dan yang kedua untuk membuat *field-field* berarti dan dapat dimengerti user.
Summarization	Terkadang *user* tidak memerlukan data pada granularitas terendah untuk analisis atau *query*, karena itu diperlukan ringkasan (*summary*) untuk disimpan ke dalam *data warehouse*.
Enrichment	Mengatur ulang dan menyederhanakan *field* individual untuk membuatnya lebih berguna untuk lingkungan *data warehouse*.

c. *Data Loading*

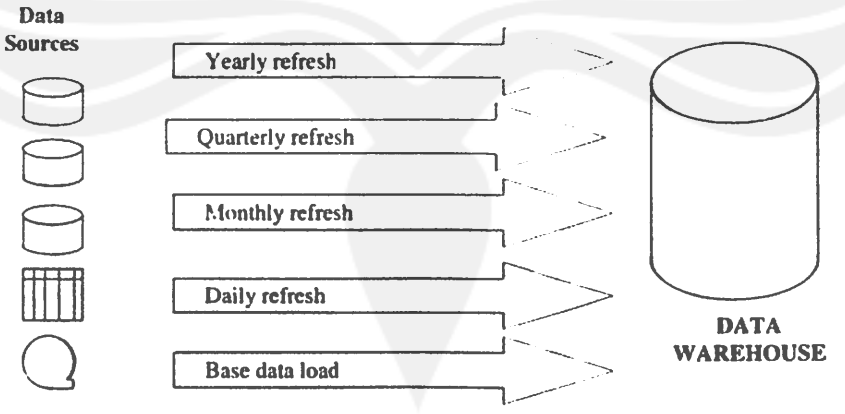
Setelah data ditransformasikan, langkah berikutnya adalah data *loading*. Sebagian besar data *loading* mencakup pengambilan data yang telah siap (bersih), mengaplikasikannya ke *data*

warehouse, dan menyimpannya ke dalam *database* yang ada disana.

3.3.3 Komponen *Data Storage* (Penyimpanan Data)

Penyimpanan data untuk *data warehouse* diletakkan pada tempat penyimpanan (*repository*) yang berbeda. Pada umumnya suatu sistem operasional dalam perusahaan, dapat mendukung operasi transaksional (*day-to-day*). Oleh karena itu, tempat penyimpanan untuk sistem operasional hanya mengandung *current data* saja.

Penyimpanan data untuk suatu *data warehouse* dibutuhkan untuk menyimpan data historikal yang bervolume besar yang digunakan untuk melakukan suatu analisis. Tempat penyimpanan data dalam *data warehouse* disimpan terpisah dari tempat penyimpanan data untuk sistem operasional. Hal ini disebabkan karena data dalam *data warehouse* hanyalah bersifat *read only*, sedangkan penyimpanan data untuk sistem operasional biasanya digunakan dalam suatu proses transaksional sehingga data-data yang terdapat didalamnya dapat terus di-*update*.



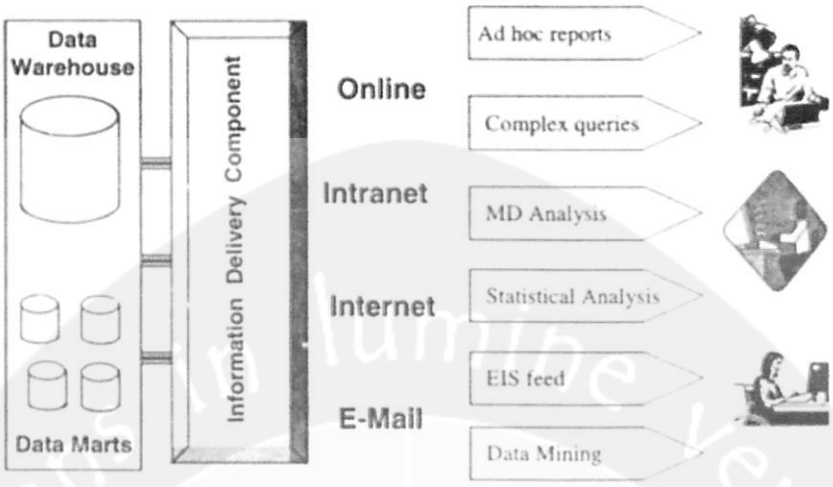
Gambar 3.5 Perpindahan Data ke dalam *Data Warehouse*

(Ponniah, 2001)

3.3.4 Komponen *Information Delivery* (Penyampaian Data)

Informasi dari *data warehouse* diperlukan oleh berbagai macam tipe user. U*ser-user* baru yang menggunakan *data warehouse* tanpa pelatihan, memerlukan *template-template report* dan *query-query* yang telah diset sebelumnya. *User-user* tidak tetap yang memerlukan informasi hanya sesekali dan tidak secara teratur. *User* tipe ini juga memerlukan paket informasi yang dipersiapkan. *User-user* analis bisnis yang memerlukan fasilitas untuk melakukan analisis kompleks menggunakan informasi dalam *data warehouse*. Terakhir *user-user* yang berkemampuan yang ingin dapat mengetahui seluruh *data warehouse*, mengambil data-data yang menarik, membentuk *query-query* sendiri, melakukan *drill* pada lapisan-lapisan data, dan membuat *report-report* tertentu dan *query-query ad hoc*.

Dalam usaha menyediakan informasi untuk komunitas *user-user data warehouse*, komponen ini meliputi beberapa metode untuk melakukan penyampaian informasi. Dalam *data warehouse* mungkin diperlukan untuk memasukkan lebih dari satu mekanisme *information delivery*. Yang paling umum yaitu penyediaan *query-query* dan *report-report* secara *online*. Para *user* akan menyampaikan permintaan secara *online* dan menerima hasilnya secara *online*. Dapat juga ditetapkan penyampaian *report-report* terjadwal melalui *e-mail*, atau dapat digunakan intranet milik organisasi yang memadai untuk penyampaian informasi. Mekanisme lainnya adalah dengan menggunakan OLAP dan *data mining*.



Gambar 3.6 Komponen Penyampaian Data (Ponniah, 2001)

3.3.5 Komponen *Metadata*

Metadata dalam suatu *data warehouse* mirip dengan kamus data atau katalog data dalam sebuah *database management systems*. Dalam kamus data, informasi yang disimpan yaitu mengenai struktur data yang bersifat logikal, informasi mengenai *file* dan alamat, informasi mengenai indeks dan lain sebagainya. Kamus data berisi data mengenai data dalam *database*. Sama halnya dengan kamus data, komponen metadata adalah data mengenai data dalam *data warehouse*. Metadata dalam *data warehouse* dibagi menjadi 3 kategori utama:

a) *Operational Metadata*

Seperti yang telah diketahui sebelumnya, data untuk suatu *data warehouse* berasal dari berbagai macam sistem operasional dalam perusahaan. Dalam memilih data dari *source systems* untuk dimasukkan ke dalam *data warehouse*, dilakukan pemisahan *record*, penggabungan bagian-bagian *record* dari berbagai sumber *file-file* yang

berbeda. Penyeleksian data dari *source systems* berhubungan dengan bermacam-macam pola pengkodean dan panjangnya *field*. *Operational metadata* berisi semua informasi mengenai sumber data operasional.

2) *Extraction and Transformasi Metadata*

Ekstraksi dan transformasi metadata mengandung data mengenai ekstraksi data dari *source systems*, yang disebut juga frekuensi ekstraksi, metode ekstraksi, dan aturan bisnis untuk ekstraksi data. Selain itu, kategori metadata berisi ini juga mengandung informasi mengenai semua transformasi data yang berlangsung di staging area.

3) *End-User Metadata*

End-user metadata adalah peta navigasi dari *data* warehouse yang memungkinkan *end-user* untuk menemukan informasi dari *data warehouse*. *End-user metadata* memampukan *end-user* untuk menggunakan istilah bisnis mereka sendiri dan mencari informasi dengan cara seperti biasanya dipikirkan dalam bisnis.

3.3.6 Komponen *Management and Control*

Komponen ini adalah arsitektur *data warehouse* yang berada di atas komponen-komponen lainnya. Komponen ini mengendalikan transformasi data dan transfer data ke dalam *data warehouse*, serta mengatur penyampaian informasi ke user. Komponen manajemen dan kontrol berinteraksi dengan komponen metadata untuk melakukan fungsi manajemen dan kontrol. Sebagai suatu komponen

metadata, komponen ini mengandung informasi mengenai data warehouse itu sendiri, metadata adalah sumber informasi untuk modul manajemen.

3.4 Pemodelan Data Multi-Dimensional

Multidimensional data adalah teknik untuk memodelkan bisnis secara konseptual sebagai sekumpulan ukuran yang dijabarkan oleh segi bisnis secara umum. Multidimensional *database* merupakan suatu cara yang digunakan untuk melakukan analisa data guna mendukung keputusan. Teknologinya didukung dengan menggunakan metoda OLAP yang dapat dirancang dengan cara khusus.

Multidimensional data mempunyai konsep dimensi, hirarki, level, dan anggota yang merupakan suatu *cube* atau kubus yang mempunyai hubungan struktur diantaranya. Kubus tersebut merupakan pusat metadata pada multidimensional. Konsep ini cukup baik dipergunakan pada data yang dapat dibuat suatu agregat yang menghasilkan bentuk keluaran berupa kalkulasi untuk sebuah aplikasi bisnis. Berikut konsep-konsep multidimensional data, yaitu:

1. Dimensi: merupakan sebuah kategori yang independent dari multidimensional *database*. Tipe dari dimensi ini mengandung item yang digunakan sebagai kriteria *query* untuk ukuran *database*. Sebagai contoh adalah pendistribusian obat di suatu daerah:
    a. Dimensi Daerah = {Jawa Timur, DKI Jakarta, Surabaya, Jakarta}

b. Dimensi Waktu = { Tahun 1999, Tahun 2000, Bulan April, Bulan Maret, Bulan Juni, Tanggal 1, Tanggal 2, Tanggal 10}

c. Dimensi Obat = { Anti Biotik, Vitamin, Ampicilin, Amoxcicilin}

2. Hirarki: merupakan bentuk kesatuan dari sebuah dimensi. Sebuah dimensi bisa terbentuk dari multilevel, yang mempunyai *parent-child relationship*. Hirarki didefinisikan bagaimana hubungan antar level. Contoh hirarki pada Dimensi Daerah:

Gambar 3.7 Hirarki pada Dimensi Daerah

3. Level: merupakan sebuah kumpulan dalam hirarki. Sebuah dimensi mempunyai *multiple layer* informasi, setiap *layer* adalah level. Contoh level pada Dimensi Daerah:

a. Propinsi = {DKI Jakarta, Jawa barat,Jawa Timur, Sumatra Selatan}

b. Kab./Kodya = {Jakarta Pusat, Bandung, Surabaya, Palembang}

c. Kecamatan = {Dago, Caringin, Senen, Matraman}

4. *Member* adalah sebuah item data dalam dimensi, kita bisa mendefinisikan ukuran *database* menggunakan *member* yang merupakan *parent-child relationship*.

Contoh *member* dalam sebuah tabel Pendistribusian Obat:

Tabel 3.2 Tabel Pendistribusian Obat

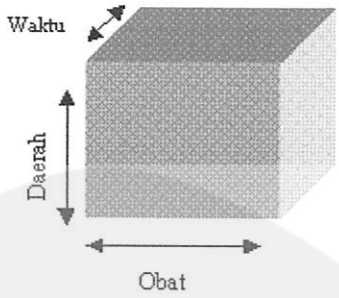
Obat	Daerah	Waktu	Jumlah
Ampicilin	Dago	1990	2359
Ampicilin	Dago	1991	5489
Ampicilin	Senen	1992	2546
Redoxon	Matraman	1990	1254
Redoxon	Senen	1991	623
Redoxon	Dago	1992	2452
Redoxon	Matraman	1992	1254
Enervon-C	Senen	1990	1254
Enervon-C	Dago	1991	1258
Vit.C	Senen	1992	671
Amoxicilin	Dago	1990	7983
Enervon-C	Caringin	1992	568

Data diatas bisa digambarkan dalam bentuk 2 dimensi.

Tabel 3.3 Bentuk 2 dimensi Tabel Pendistribusian Obat

Penjualan Obat			Dimensi 1			
			th	1990	1991	1992
Dimensi 2	Obat	Daerah				
	Ampicilin	Dago		2359	5489	
		Senen				2546
		.....				
	Enervon-C	Dago			1258	
		Senen		1254		
		.....				

Dari data diatas dapat di bentuk suatu analisa data. Data tersebut dapat diambil suatu analisa untuk total penjualan obat Ampicilin atau total penjualan obat selama tahun 1990 atau total penjualan untuk daerah Matraman. Data tersebut dapat di gambarkan dalam bentuk kubus dimensi, seperti berikut:



Gambar 3.8 Bentuk 3 dimensi tabel Pendistribusian Obat (www.informatika.lipi.go.id/olap-dan-terminologi-multi-dimensional-database)

Model penyimpanan multidimensi terdiri dari 2 tipe tabel yaitu tabel dimensi dan tabel fakta. Karakteristik dari tabel fakta yaitu:

a. Dapat tumbuh secara cepat.
b. Dapat mengenai volume data yang cukup besar.
c. Terdiri dari ukuran-ukuran numerik bisnis.
d. Terhubung dengan tabel-tabel dimensi melalui *foreign key* yang akan mereferensi *primary key* pada tabel-tabel dimensi.

Sedangkan karakteristik dari tabel dimensi yaitu:

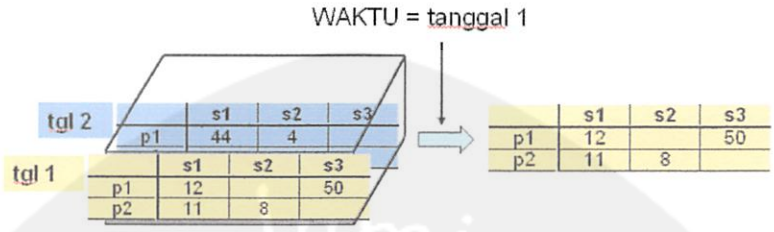
a. Terdiri dari informasi tekstual yang merepresentasikan atribut-atribut bisnis.
b. Terdiri dari metadata sederhana.
c. Terdiri dari data *static*.
d. Terhubung dengan tabel fakta melalui referensi dari *foreign key*.

Operasi pada Multidimensional Data:

1. *Slicing dan Dicing*

   Operasi ini digunakan untuk mengambil potongan kubus berdasarkan nilai tertentu pada

satu atau beberapa dimensinya. Operasi ini dapat dilakukan dengan *query* biasa.



Gambar 3.9 Contoh operasi *slicing-dicing*

2. *Pivoting*

Operasi ini digunakan untuk menampilkan nilai-nilai ukuran dalam tata letak tabel yang berbeda.

Operasi ini juga dapat dilakukan pada model data 2 dimensi.

	Penjualan (juta $)			
	Produk:	Waktu		
		Tgl-1	Tgl-2	
Toko t1	Electronics	$5.2		
	Toys	$1.9		
	Clothing	$2.3		
	Cosmetics	$1.1		
Toko t2	Electronics	$8.9		
	Toys	$0.75		
	Clothing	$4.6		
	Cosmetics	$1.5		

Slice & Pivot

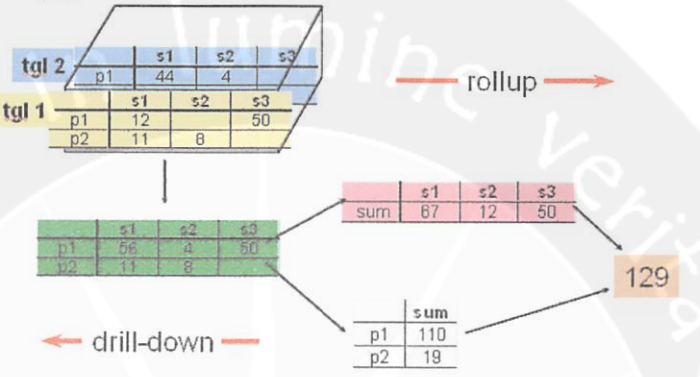
Penjualan (juta $)			
Produk:	Tgl-1		
	Toko t1	Toko t2	
Electronics	$5.2	$8.9	
Toys	$1.9	$0.75	
Clothing	$2.3	$4.6	
Cosmetics	$1.1	$1.5	

Gambar 3.10 Contoh operasi *pivoting*

3. *Roll-up* dan *Drill-down*

Operasi *Roll-up* digunakan untuk melakukan generalisasi atau naik ke tingkat atasnya dalam hirarki dimensi sebanyak satu atau beberapa

dimensi dengan cara merangkum nilai-nilai ukurannya. Sedangkan operasi *Drill-down* digunakan untuk memilih dan menampilkan data rincian dalam satu atau beberapa dimensi. Operasi *Drill-down* merupakan kebalikan dari operasi *Roll-up*.



Gambar 3.11 Contoh operasi *roll-up*

Skema Multidimensinal Data:

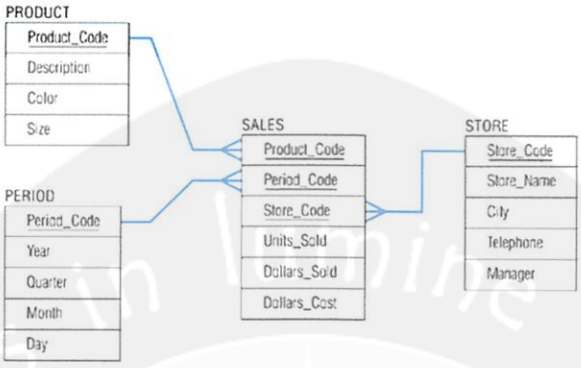
1. *Star Schema*

Terdiri dari tabel fakta terpusat dan tabel dimensi yang terdapat disekelilingnya yang terhubung oleh *Primary Key* dan *foreign key*. Pada skema ini, tabel fakta tunggal berisi data rinci dan agregat. Satu kolom kunci untuk tiap dimensi sebagai kunci primer (*primary key*) tabel fakta. Nilai-nilai pada kolom kunci telah terdefinisi. Setiap dimensi akan direpresentasikan dalam satu tabel yang umumnya sangat terdenormalisasi. Keuntungan *Star Schema*:

a. Mudah dipahami.
b. Mudah untuk merepresentasikan hirarki dimensi.
c. Metadata tidak rumit.

d. *Low maintenance*.

e. Jumlah operasi *join* minimal.



Gambar 3.12 Contoh *Star Schema*

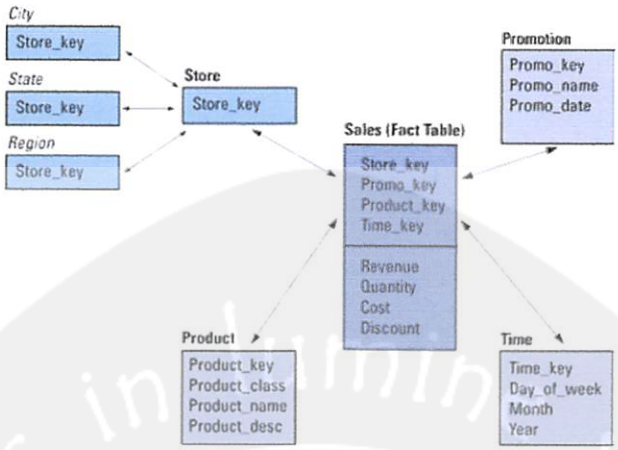
2. *Snowflake Schema*

Skema ini lebih dekat dengan *Entity Relationship Diagram* (*ERD*) karena lebih dinormalisasi. Pada skema ini, atribut level tidak diperlukan lagi. Setiap tabel dimensi memiliki satu kolom kunci untuk setiap level dalam hirarki dimensi. Tabel dimensi pada level terendah menggabungkan atribut-atribut tabel dimensi lainnya dan masih berupa tabel fakta yang terdenormalisasi.

Kelebihan *Snowflake Schema* dibandingkan *Star Schema* adalah pemrosesan *query* tinggi untuk *query-query* yang melibatkan agregasi (hitungan total). Sedangkan kekurangan *Snowflake Schema*:

a. Rumit dalam pemeliharaan dan metadatanya.

b. Jumlah tabel dalam database membengkak.



Gambar 3.13 Contoh *Snowflake Schema*

3.5 *Data Mart*

Selain istilah *data warehouse*, dikenal juga istilah *data mart*. *Data mart* adalah versi yang lebih kecil atau *subset* dari *Data Warehouse* yang mendukung kebutuhan-kebutuhan departemental atau fungsi bisnis tertentu dalam perusahaan. *Data mart* mengandung *subset* data pada *data warehouse* yang biasanya merupakan ringkasan data yang berhubungan dengan departemen khusus atau fungsi bisnis tertentu.

Data mart tidak hanya berukuran lebih kecil namun *data mart* merupakan *Data Warehouse* yang lebih fokus. Pada banyak kasus, perusahaan menemukan kegunaan menciptakan *data mart* untuk unit bisnis yang spesifik yang memiliki kesamaan kebutuhan analisis data yang lebih spesifik. *Data mart* dapat berguna sebagai langkah pertama untuk membangun sebuah *Data Warehouse* yang skala besar. *Data mart* dapat juga diektraksi dari sebuah *Data Warehouse* dan ditempatkan pada sebuah lokal server untuk digunakan oleh anggota dari sebuah *work*

group ketika isi *data mart* mencukupi untuk kebutuhan yang terbatas.

Data mart menyediakan suatu struktur sistem informasi yang memperkenankan organisasi memiliki akses data fleksibel untuk melakukan *slice* dan *dice* pada data dengan berbagai cara dan menyelidiki hubungan antara *summary* dan detail data secara dinamis. *Data mart* adalah repositori kumpulan data dari data operasional dan sumber-sumber lain yang dirancang untuk menyajikan kumpulan bagian informasi dari para pekerja pengetahuan. Dalam cakupannya, data mungkin diproses dari *database enterprise* yang luas atau *data warehouse* atau mungkin lebih khusus.

Beberapa alasan yang mendasari pembuatan *data mart* antara lain:

1) Memberikan hak kepada *user* untuk mengakses data yang sering dibutuhkan untuk dianalisis.
2) Menyediakan data dalam bentuk yang sesuai dengan keseluruhan *view* data untuk sekelompok *user* dalam suatu departemen.
3) Meningkatkan waktu respon bagi *end-user* karena pengurangan volume data yang diakses.
4) Normalnya *data mart* menggunakan data yang lebih sedikit sehingga tugas-tugas seperti pembersihan data, *load* data, transformasi, dan integrasi menjadi jauh lebih mudah, dan karena itu pengimplementasian dan pembentukan *data mart* lebih sederhana daripada pembentukan *data warehouse*.

5) Biaya implementasi *data mart* normalnya lebih sedikit bila dibandingkan dengan kebutuhan dalam pembentukan *data warehouse*.

Ada beberapa pendekatan dalam pembangunan *data mart*. Pendekatan pertama yaitu membangun beberapa *data mart* dengan menggunakan data yang terintegrasi ke *data warehouse*. Pendekatan lainnya adalah dengan membuat *data mart* sejalan dengan pembuatan *data warehouse*. Hal ini dikarenakan untuk pemenuhan kebutuhan bisnis yang mendesak, sebagai contoh perusahaan ingin segera mengetahui total penjualan produk tertentu pada saat tertentu (Connolly & Begg, 2005).

3.5.1 Tipe-Tipe *Data Mart*

Data mart dapat dikategorikan ke dalam dua tipe, yaitu: *Independent Data Mart* dan *Dependent Data Mart*. Pengkategorian ini didasarkan pada sumber data (*data sources)* dari *data mart* tersebut. Berikut ini akan dijelaskan secara lebih rinci tentang kedua tipe ini.

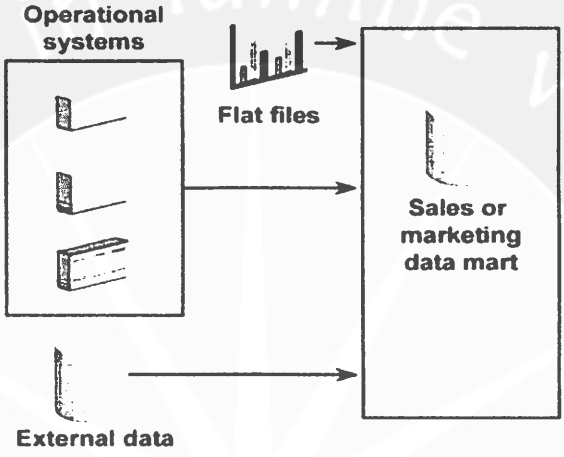
3.5.1.1 *Independent Data Mart*

Independent data mart merupakan sebuah sistem *stand-alone* yang dibangun per bagian, yang mengambil data langsung dari sumber-sumber data operasional atau eksternal. *Independent data mart* mempunyai karakteristik sebagai berikut:

a. Sumbernya adalah sistem-sistem operasional dan eksternal.
b. Proses ETL-nya sulit. Karena *independent data mart* mengambil data dari sumber-sumber data yang belum bersih atau belum konsisten,

sehingga usaha yang dilakukan dititikberatkan kepada *error processing* dan integritas data.

c. *Data mart* dibangun untuk memenuhi kebutuhan-kebutuhan analitik. Penciptaan *independent data mart* seringkali diakibatkan karena kebutuhan solusi yang cepat untuk permintaan-permintaan analisis.



Gambar 3.14 *Independent Data Mart*

3.5.1.2 *Dependent Data Mart*

Dependent data mart mempercayakan data-datanya pada *enterprisewide warehouse*, yang menyediakan beberapa keuntungan seperti berikut:

a. Konsistensi data pada *data mart* terjamin.

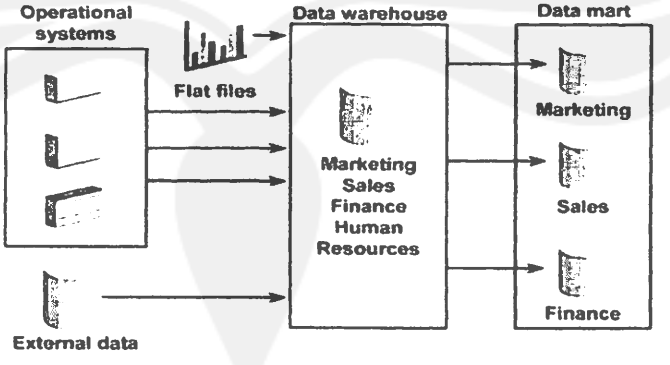
b. Proses pengembangan untuk pemindahan, transformasi, dan penyimpanan data pada *data warehouse* hanya membutuhkan modifikasi dan penambahan apabila kita menciptakan *data mart* tambahan.

c. Manajemen *database* lebih mudah, karen aturan-aturan untuk *backup*, *recovery*, dan pengarsipan tersedia untuk semua *data mart* yang ada.

d. Data yang lebih detail dapat disimpan dalam *enterprise warehouse*, dan *data mart* dapat berbagi informasi, yang memungkinkan

Sedangkan karakteristik dari *dependent data mart* antara lain:

1. Sumber datanya adalah *data warehouse*.
2. Proses ETL-nya mudah. *Dependent data mart* mengambil data dari *data warehouse* pusat yang telah terlebih dahulu dibangun. Karena itu upaya utama dari membangun sebuah *data mart*, yaitu membersihkan dan mengekstraksi, dapat dilewatkan. *Data mart* tipe ini hanya membutuhkan data untuk dipindahkan dari satu *database* ke *database* lainnya.
3. *Data mart* ini adalah bagian dari perencanaan perusahaan. *Dependent data mart* biasanya dibangun untuk mencapai ketersediaan data dan performa yang lebih baik, kontrol yang lebih baik, dan biaya telekomunikasi yang lebih rendah sebagai hasil dari akses lokal ke data yang relevan terhadap departemen tertentu.



Gambar 3.15 *Dependent Data Mart* (Green, 2002)

3.5.2 *Data Warehouse* dan *Data Mart*

Seperti halnya *data warehouse*, *data mart* terdiri atas sebuah gambaran dari data operasional yang membantu pelaku bisnis untuk mengatur strategi berdasarkan analisis dari kecenderungan yang terjadi pada masa lalu dan pengalaman-pengalaman. Kunci yang membedakannya adalah pembuatan sebuah data mart berdasarkan pada sebuah data khusus, kebutuhan yang telah didefinisikan sebelumnya untuk sebuah pengelompokan tertentu dan konfigurasi dari data terpilih. Sejak konfigurasi data mart menekankan pada kemudahan akses pada informasi yang saling berhubungan, *star schema* atau model dimensional adalah pilihan desain yang sangat popular, karena menyediakan sebuah *relational database* untuk melebihi fungsi analisis dari sebuah *multidimensional database*.

Karakteristik yang membedakan antara *data mart* dan *data warehouse* adalah sebagai berikut (Connolly & Begg, 2005):

1. *Data mart* memfokuskan hanya pada kebutuhan-kebutuhan pemakai yang terkait dalam sebuah departemen atau fungsi bisnis.
2. *Data mart* biasanya tidak mengandung data operasional yang rinci, seperti halnya pada pada *data warehouse*.
3. Karena *data mart* mengandung lebih sedikit data bila dibandingkan dengan *data warehouse*, maka *data mart* lebih mudah untuk dipahami dan dinavigasikan.

Perbedaan-perbedaan antara *data mart* dan *data warehouse* dapat dilihat pada tabel di bawah ini:

Tabel 3.4 Perbandingan Data Warehouse dan Data Mart

Data Warehouse	Data Mart
▪ Lingkup perusahaan/*Enterprise*	▪ Lingkup departemen
▪ Perpaduan dari semua *data mart*	▪ Subjek tunggal, *line of bussiness* (LOB)
▪ Menerima data dari *staging area*	▪ *Star-join* (*facts* dan *dimesions*)
▪ Struktur data sesuai cara pandang perusahaan	▪ Struktur data sesuai dengan cara pandang tiap departemen
▪ Teknologi yang optimal untuk menyimpan dan mengatur data dalam jumlah yang sangat besar	▪ Teknologi optimal untuk akses dan analisa data
▪ Diorganisir pada E-R model	

3.6 Oracle Warehouse Builder

Oracle Warehouse Builder (OWB) lebih dari sekedar alat bantu (*tool*) ETL. OWB juga merupakan *tool* untuk perancangan dan pemeliharaan *data warehouse, data mart,* dan *e-business intelligence*. Aspek penting OWB berasal dari perspektif pelanggan yang memungkinkan integrasi dari lingkungan *data warehouse* tradisional dengan lingkungan *e-business* yang baru (Oracle Corporation, 2000). Sumber data OWB tidak hanya terbatas pada *database* Oracle saja. OWB juga mendukung *database* lain, mencakup DB2, SQL Server, Sybase, Informix, dan Teradata. OWB juga bekerja dengan *file* dan aplikasi sumber data pada perusahaan, sehingga OWB mendukung koleksi data yang lengkap untuk perusahaan.

OWB menyediakan kemampuan *profiling* dan *auditing* melalui *user interface*-nya, menyediakan manajemen

kualitas data yang lengkap ketika dikombinasikan dengan keikutsertaan *user* tersebut dan peningkatan proses. Dari sejarah yang ada, pembersihan dan *profiling* data memerlukan pengetahuan yang baik mengenai SQL dan PL/SQL. Akan tetapi OWB terintegrasi dengan *profiler* dan *Data Correction Wizard* yang menjadikan tugas tersebut relatif lebih mudah untuk dilaksanakan. Warehouse Builder terdiri dari komponen fungsional utama, seperti berikut:

1. *Repository*

   Terdiri dari satu set tabel pada *database* Oracle yang diakses melalui Java-*based access layer*. Semua pekerjaan user disimpan dalam *repository*, sebagai contoh: definisi sumber, definisi target, dan *mapping* sumber ke target.

2. *Graphical User Interface* (GUI)

   Memungkinkan akses ke *repository*. GUI menonjolkan editor grafis dan penggunaan *wizard* secara luas. GUI ditulis dalam Java, yang memampukan untuk dapat berjalan pada *platform* Windows dan UNIX.

3. *Code Generator*

   Ditulis dalam Java dan didasarkan pada definisi yang ada pada *repository*, serta men-*generate code* untuk *implementasi* pada *warehouse*.

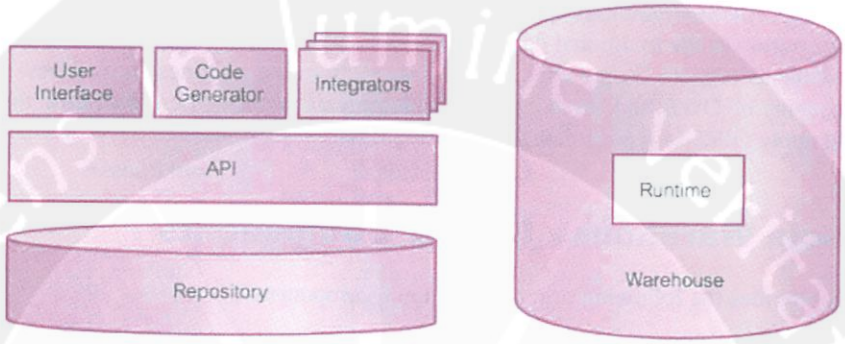
4. *Integrators*

   Merupakan komponen yang digunakan untuk mengekstrak data yang berasal dari sumber data tertentu.

5. *Run Time*

   Merupakan satu set tabel, *sequence*, *package*, dan *trigger* yang di*install* pada skema target. Objek

database ini adalah dasar dari kemampuan *auditing* dan deteksi kesalahan dari OWB. Sebagai contoh, *load* data dapat di-*restart* berdasarkan informasi yang tersimpan di tabel *runtime*. OWB mencakup sebuah *runtime audit viewer* untuk *browsing* tabel *runtime* dan *report runtime*.



Gambar 3.16 Arsitektur Oracle Warehouse Builder
(Connolly and Begg, 2005)

OWB membantu *user* dalam melakukan beberapa tugas pada *data warehouse*, seperti berikut ini:

1. Pendefinisian struktur data sumber

   Segera setelah kebutuhan ditetapkan dan sumber data telah dikenali, *tool* seperti OWB dapat digunakan untuk membangun *data warehouse*. OWB mampu menangani bermacam-macam sumber data dengan menggunakan *integrator*. OWB juga mempunyai konsep sebuah modul yang merupakan pengelompokkan yang logis dari objek yang saling terkait.

   Ada dua tipe modul, yaitu sumber data dan *warehouse*. Sebagai contoh, modul sumber data mungkin berisi semua definisi tabel dalam *database* OLTP yang merupakan sumber untuk data warehouse. Dan modul yang bertipe *warehouse* mungkin berisi

definisi dari tabel fakta, dimensi, dan *staging* yang menyusun *data warehouse*.

Hal ini penting untuk dicatat bahwa modul selalu berisi definisi, yaitu metadata dari salah satu sumber atau *warehouse*. *User* mengidentifikasikan integrator yang sesuai dengan sumber data, dan setiap integrator mengakses sebuah sumber dan meng-*import* metadata yang menggambarkannya.

2. Perancangan target *warehouse*

Langkah selanjutnya adalah merancang target *warehouse* berdasarkan kebutuhan *user*. Salah satu desain yang paling populer adalah *star schema* dan variasinya. Selain itu, *tool* intelegensi bisnis juga banyak yang dioptimalkan untuk jenis desain tersebut. OWB mendukung semua variasi dari desain skema. OWB menonjolkan *wizard* dan editor grafis untuk tabel fakta dan dimensi. Contohnya, dalam *Dimension Editor* didefinisikan atribut, level, dan hirarki dari dimensi.

3. Pemetaan (*mapping*) sumber ke target

Bilamana sumber dan target telah terdefinisi, tugas selanjutnya adalah memetakan keduanya secara bersama. Modul sumber dapat dipakai kembali dalam pemetaan yang berbeda. Modul *warehouse* sendiri dapat juga digunakan sebagai modul sumber. Contohnya, pada arsitektur dimana *database* OLTP memberi masukan data ke *data warehouse* pusat, yang kemudian memberi masukan data ke *data mart*, maka di sini *data warehouse* berperan sebagai target

dilihat dari perspektif *database* OLTP dan sebagai sumber dilihat dari perspektif *data mart*.

Pemetaan dari OWB didefinisikan menjadi dua level. *High-level mapping* menandai adanya modul sumber dan target. Satu level dibawahnya adalah *detail mapping* yang memperbolehkan *user* untuk memetakan kolom sumber ke dalam kolom target dan menetapkan transformasi. Fitur *transformation library* dari OWB memungkinkan user untuk dapat memilih transformasi yang sudah ditetapkan sebelumnya.

4. *Generate code*

   Code Generator adalah komponen OWB yang membaca definisi target dan pemetaan sumber ke target dan men-*generate code* untuk implementasi *warehouse* selanjutnya. Tipe code yang dihasilkan dapat bervariasi tergantung dari tipe objek yang *user* inginkan untuk implementasi.

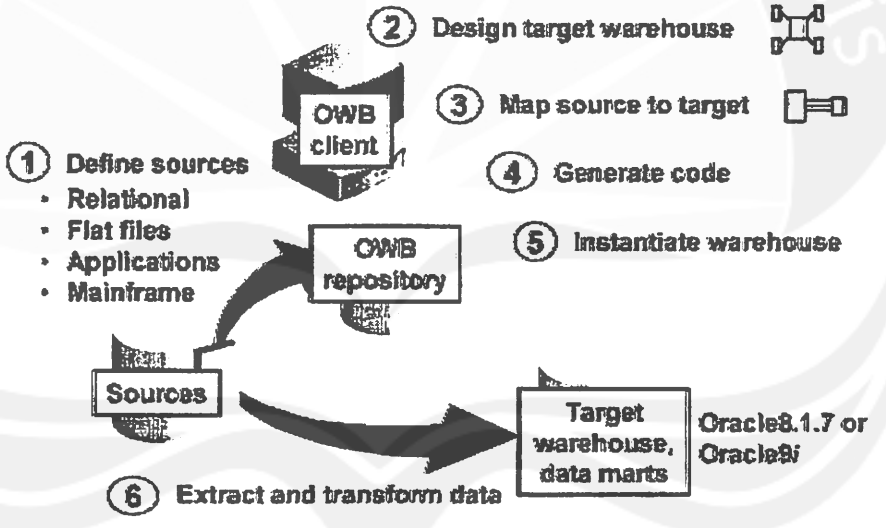
5. Pembentukan *warehouse* dan ekstrak data

   Sebelum data dipindahkan dari sumber ke *database* target, *developer* harus membentuk *warehouse*. Dengan kata lain, menjalankan *script* DDL yang telah di*generate* untuk menghasilkan skema target. OWB memilih langkah ini sebagai *deployment*. Segera setelah skema target terbentuk, program PL/SQL dapat memindahkan data dari sumber ke dalam target.

6. Pemeliharaan *data warehouse*

   Setelah *data warehouse* terbentuk dan proses *initial load* selesai dijalankan, maka *data warehouse* harus dipelihara. Sebagai contoh, tabel

fakta harus di*refresh* secara teratur, sehingga *query* yang dijalankan mengembalikan hasil yang *up-to-date*. Tabel dimensi harus diperluas dan diperbaharui, walaupun frekuensinya sangat sedikit dibandingkan dengan tabel fakta. Salah satu contoh dari *slowly changing dimension* adalah tabel *customer*, dimana alamat, status, atau nama mungkin saja berubah sepanjang waktu. Selain INSERT, OWB menyediakan cara lain untuk memanipulasi data warehouse antara lain UPDATE, DELETE, INSERT/UPDATE (*insert* baris, jika sudah ada, akan diubah), dan UPDATE/INSERT (*update* baris, jika tidak ada, akan ditambahkan).



Gambar 3.17 Alur Proses Pembangunan *Data Warehouse* dalam OWB