BAB 2

LANDASAN TEORI

2.1 Teknologi Informasi

Menurut Alter (1999, p42) teknologi informasi adalah perangkat keras dan piranti lunak yang digunakan dalam sistem informasi.

1) Perangkat keras

   Mengarah pada peralatan dan obyek fisik lain yang terlibat dalam proses informasi seperti komputer, *workstation, physical network*, *data storage, dan transmission device*.

2) Piranti lunak

   Mengarah pada program komputer untuk proses penerjemahan input *user* dan menyampaikan pada perangkat keras mengenai apa yang harus dilakukan. Yang termasuk dalam piranti lunak yaitu sistem operasi, *end user software* seperti *word processor*, dan *application software* yang berhubungan dengan tugas bisnis yang khusus seperti merekam transaksi kartu kredit atau merancang mobil.

2.2 Konsep Dasar Sistem Informasi

Menurut Alter (1999, p42) sistem informasi adalah sistem kerja yang menggunakan teknologi informasi untuk mengambil, mengirimkan, menyimpan, mendapatkan kembali, memanipulasi atau menampilkan informasi.

Menurut O'Brien (2003, p4) sistem informasi adalah kombinasi terorganisir dari manusia, perangkat keras, perangkat lunak, jaringan komunikasi dan sumber data yang mengumpulkan, mengolah, dan menyebarkan informasi di dalam suatu organisasi.

2.3 Prototyping

Menurut McLeod (2004, p163) *prototype* adalah sebuah versi dari suatu sistem potensial yang menyediakan pengembang dan *user* dengan suatu gambaran tentang bagaimana sistem dalam bentuk sempurnanya akan berfungsi. Proses untuk menghasilkan sebuah *prototype* disebut *prototyping*. Ada 2 jenis *prototype*, antara lain:

- *Evolutionary prototype*

  Evolutionary prototype akan terus-menerus disempurnakan hingga mempunyai semua fungsi yang dibutuhkan *user* dari sistem yang baru. Kemudian barulah pembuatannya dilanjutkan. Jadi, *evolutionary prototype* nantinya akan menjadi sistem yang sesungguhnya.

- *Requirements prototype*

  Requirements prototype dikembangkan sebagai suatu cara untuk menentukan kebutuhan fungsional dari sistem yang baru ketika *user* tidak bisa menyampaikan secara eksplisit apa yang mereka inginkan. Dengan meninjau kembali *requirements prototype* selagi fitur-fitur ditambahkan, *user* bisa menetapkan proses yang dibutuhkan untuk sistem baru tersebut. Ketika semua kebutuhannya terpenuhi, *requirements prototype* sudah

mencapai tujuannya dan proyek lain akan dibangun untuk mengembangkan sistem baru tersebut. Jadi, *requirements prototype* tidak akan menjadi sistem yang sesungguhnya.

Keuntungan menggunakan pendekatan *prototyping*, antara lain:

- Memperlancar komunikasi antara pengembang dengan *user*.
- Pengembang menjadi lebih jeli dalam menentukan kebutuhan *user*.
- *User* memegang peran yang lebih aktif dalam pengembangan sistem.
- Pengembang dan *user* menghabiskan lebih sedikit waktu dan usaha dalam mengembangkan sistem.
- Implementasi menjadi lebih mudah karena *user* telah mengetahui apa yang diharapkan.

Sedangkan kelemahan menggunakan pendekatan *prototyping* adalah:

- Sikap terlalu tergesa-gesa dalam membuat *prototype* bisa menyebabkan diambilnya jalan pintas dalam pendefinisian masalah, evaluasi alternatif, dan dokumentasi. Hal ini menghasilkan usaha yang cepat namun tidak akurat lagi.
- *User* bisa menjadi terlalu bersemangat dengan *prototype*, yang mengarah ke ekspektasi yang tidak realistik terhadap pengembangan sistem.
- *Computer-human interface* yang disediakan alat *prototype* tertentu tidak mencerminkan teknik perancangan yang baik.

2.4 Konsep Database

Menurut McLeod (2004, p130) *database* adalah koleksi semua data yang berbasis komputer dalam suatu perusahaan.

Menurut Connoly (2002, p14) *database* adalah koleksi bersama dari data yang terkait secara logis, dan suatu deskripsi dari data, yang dirancang untuk memenuhi kebutuhan informasi dalam suatu organisasi.

Menurut Inmon (2002, p388) *database* adalah sebuah kumpulan dari data yang saling berhubungan yang disimpan (biasanya dengan redundansi yang terkontrol dan terbatas) berdasarkan suatu skema.

Menurut Date (2000, p10) *database* adalah koleksi dari *persistent data* yang digunakan oleh sistem aplikasi dari suatu perusahaan.

2.5 Definisi Data Warehouse

Menurut Inmon (2002, p31) *data warehouse* adalah koleksi data yang mempunyai sifat *subject-oriented, integrated, nonvolatile*, dan *time-variant* untuk mendukung proses pengambilan keputusan dalam manajemen.

Menurut Mannino (2001, p455) *data warehouse* adalah tempat penyimpanan data terpusat di mana data dari *database* operasional dan sumber lainnya diintegrasikan, dibersihkan, dan diarsipkan untuk mendukung pengambilan keputusan.

Menurut McLeod (2004, p406) *data warehouse* adalah sebuah sistem penyimpanan data yang berkapasitas besar, di mana data dikumpulkan dengan menambahkan *record* baru daripada meng-*update record* yang sudah ada

dengan informasi baru. Data jenis ini digunakan hanya untuk proses pengambilan keputusan dan bukan untuk kegiatan operasional perusahaan sehari-hari.

2.6 Karakteristik Data Warehouse

Beberapa karakteristik *data warehouse* menurut Mannino (2001, p455) adalah sebagai berikut:

2.6.1 Subject-Oriented

Sebuah *data warehouse* diorganisasikan melingkupi subyek atau entitas bisnis utama seperti pelanggan, pesanan atau produk. Orientasi subyek ini berbeda dengan pengolahan transaksi yang lebih berorientasi proses.

Tabel 2.1 Tabel Perbandingan Data Warehouse dengan OLTP (Connoly, 2002, p1049)

Sistem OLTP	Sistem Data Warehouse
Menyimpan data sekarang	Menyimpan data historis
Menyimpan *detailed data*	Menyimpan *detailed, lightly, highly summarized data*
Data bersifat dinamis	Data bersifat statis
Proses yang dilakukan secara berulang	*Ad hoc*, tidak terstruktur, *heuristic processing*
High level dari *transaction throughput*	*Medium* ke *low level* dari *transaction throughput*
Pemakaian dari pola yang dapat diprediksi	Pemakaian dari pola yang tidak dapat diprediksi
Mengarah pada transaksi	Mengarah pada analisis
Berorientasi pada aplikasi	Berorientasi pada subyek
Mendukung keputusan sehari-hari	Mendukung keputusan strategi
Operational user dalam jumlah yang besar	*Managerial user* dalam tingkat yang relatif rendah

2.6.2 Integrated

Data operasional dari beberapa *database* dan sumber data eksternal diintegrasikan dalam sebuah *data warehouse* untuk memperoleh sebuah *database* tunggal yang mendukung sebuah keputusan. Penggabungan data membutuhkan konvensi penamaan yang konsisten, format data yang seragam, serta skala pengukuran lintas *database* dan sumber data eksternal yang dapat diperbandingkan.

2.6.3 Time-Variant

Data warehouse menggunakan *time stamp* untuk merepresentasikan data historis. Dimensi waktu sangat kritis untuk mengidentifikasikan *trend*, memprediksi operasi-operasi mendatang, dan mengatur sasaran-sasaran yang beroperasi. *Data warehouse* terdiri dari serangkaian *snapshot*, masing-masing merepresentasikan data operasional yang diambil pada suatu waktu tertentu.

2.6.4 Nonvolatile

Data baru di dalam sebuah *data warehouse* adalah ditambahkan, dan bukan digantikan, sehingga data historis tetap terjaga. Tindakan menambahkan data baru juga dikenal dengan pembaharuan *data warehouse*. Kurangnya operasi *update* dan *delete* memastikan sebuah *data warehouse* bebas dari anomali *update* maupun *delete*. Transaksi

data dipindahkan ke dalam *data warehouse* hanya ketika aktivitas perubahan terbaru telah diselesaikan.

2.7 Struktur Data Warehouse

2.7.1 Current Detail Data

Current detail data merupakan level terendah dalam struktur *data warehouse*. *Current detail data* menggambarkan data detil yang aktif pada saat ini dan keadaan yang sedang berjalan. Data jenis ini memerlukan media penyimpanan yang besar dan merupakan data yang sering diakses. *Current detail data* ini cepat diakses, tetapi mahal dan kompleks dalam pemeliharaannya.

2.7.2 Older Detail Data

Older detail data merupakan data *back-up* (cadangan) yang jarang diakses. Data *back-up* seperti ini biasanya disimpan pada media penyimpanan yang berbeda. Penyusunan direktori dilakukan berdasarkan urutan umur data, sehingga data dapat tersusun rapi dan mempermudah dalam melakukan akses selanjutnya.

2.7.3 Lightly Summarized Data

Lightly summarized data merupakan data ringkasan dari *current detail data*. Di dalam tahap ini, data belum dapat digunakan untuk pengambilan keputusan karena data masih belum bersifat *total summary*, yang artinya

data masih bersifat detil. Akses terhadap data jenis ini biasanya digunakan untuk memantau kondisi yang sedang dan sudah berjalan.

2.7.4 Highly Summarized Data

Highly summarized data merupakan data yang bersifat *total summary*. Pada level ini, data sangat mudah diakses terutama untuk melakukan analisis perbandingan data berdasarkan urutan waktu dan analisis yang menggunakan data multidimensi. Data multidimensi adalah suatu teknologi *software* komputer yang dirancang untuk meningkatkan efisiensi dalam *query* data sehingga menjadi media penyimpanan yang lebih baik, serta memudahkan pengambilan data dalam volume besar.

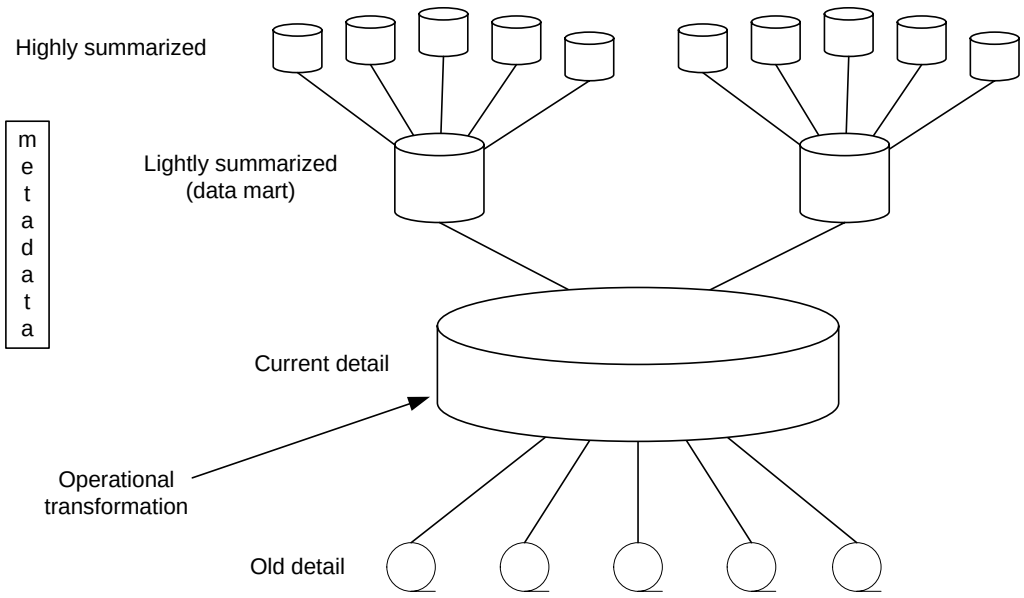
2.7.5 Metadata

Menurut Inmon (2002, p393) *metadata* merupakan data tentang data. *Metadata* merupakan gambaran tentang struktur, isi, kunci, indeks dari data.

Menurut Poe (1996, p31) *metadata* adalah “data tentang data” dan menyediakan informasi tentang struktur data dan *relationships* di antara masing-masing struktur data ataupun antar-*database*.

Menurut Berson (2000, p60) *metadata* adalah data tentang data yang menggambarkan *data warehouse*. *Metadata* dapat dikelompokkan ke dalam:

- *Technical metadata*, yang berisi informasi mengenai *data warehouse* yang digunakan oleh administrator dan perancang *data warehouse* ketika melakukan pengembangan *data warehouse* dan tugas-tugas manajemen.
- *Business metadata*, yang berisi informasi yang memberikan *user* suatu perspektif yang mudah dimengerti dari informasi yang tersimpan dalam *data warehouse*.
- *Data warehouse operational information*, seperti *data history* (*snapshots*, *versions*), *ownership*, menelusuri jejak audit, penggunaan data.



Gambar 2.1 Struktur Data Warehouse (Inmon, 2002, p36)

2.8 Anatomi Data Warehouse

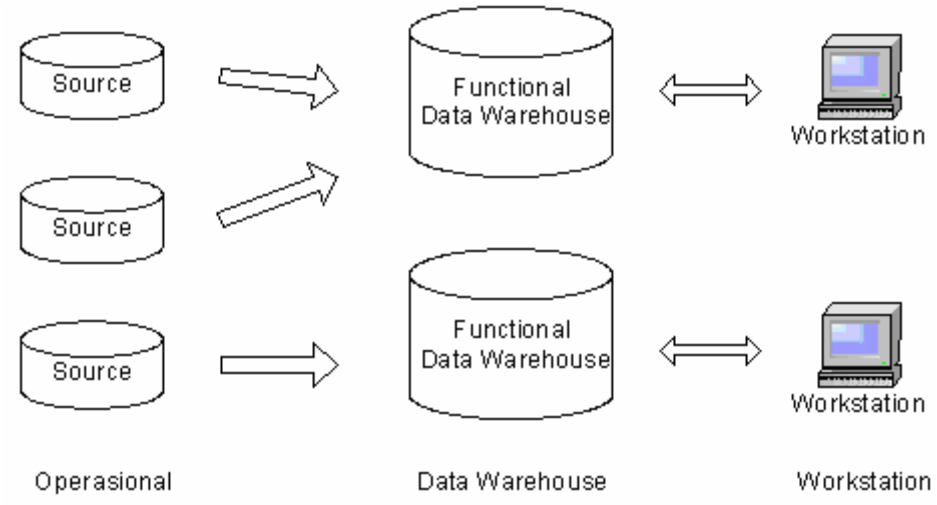
Berikut ini adalah tiga jenis dasar sistem anatomi *data warehouse* menurut Prabowo (1996a):

2.8.1 Functional Data Warehouse

Data warehouse fungsional menggunakan pendekatan kebutuhan dari tiap bagian fungsi bisnis, misalnya departemen, divisi, dan sebagainya, untuk mendefinisikan jenis data yang ditampung oleh sistem. Setiap unit fungsi dapat memiliki gambaran data masing-masing.

Pendekatan ini banyak digunakan karena sistem memberikan solusi yang mudah untuk dibangun dengan biaya investasi yang relatif rendah dan dapat memberikan kemampuan sistem pengumpulan data yang terbatas kepada kelompok pemakai.

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



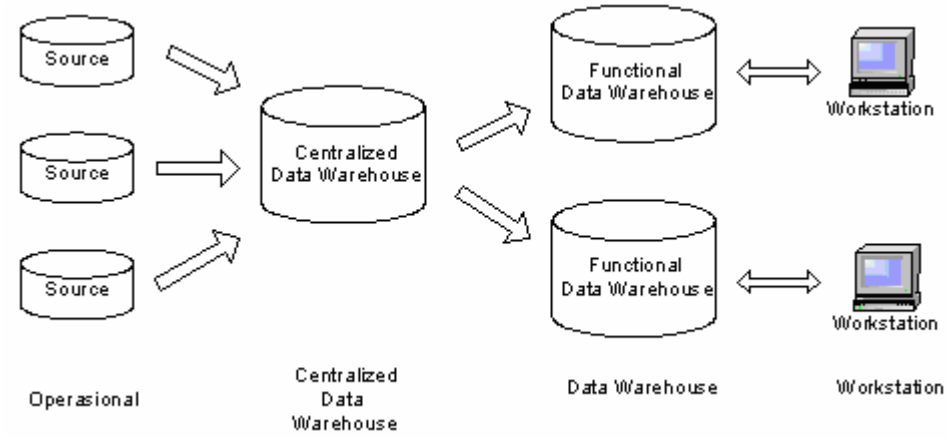
Gambar 2.2 Functional Data Warehouse

2.8.2 Centralized Data Warehouse

Data warehouse terpusat adalah pendekatan yang paling baik digunakan, disebabkan oleh keterbiasaan pemakai dengan lingkungan *mainframe* terpusat. Data diambil dari seluruh sistem operasional dan disimpan di dalam pusat penyimpanan data. *User* kemudian menggunakan data yang telah terkumpul tersebut untuk membangun *data warehouse* fungsional masing-masing sesuai dengan kebutuhannya.

Keuntungan sistem ini dibanding *data warehouse* fungsional adalah bahwa data benar-benar terpadu. Sistem ini mengharuskan data dikirim tepat pada waktunya agar tetap konsisten dengan pemasok data lainnya. Di samping itu, *user* hanya dapat mengambil data dari pusat pengumpulan saja dan tidak dapat berhubungan secara langsung dengan pemasok datanya sendiri.

Penerapan sistem ini membutuhkan biaya pemeliharaan yang tinggi atas sistem pengumpulan data yang besar. Selain itu, diperlukan waktu yang lama untuk membangun sistem tersebut.



Gambar 2.3 Centralized Data Warehouse

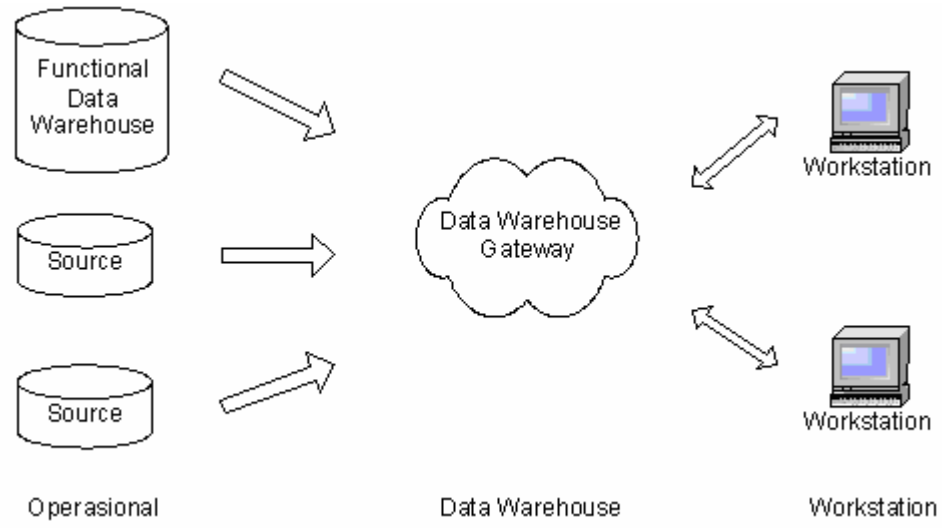
2.8.3 Distributed Data Warehouse

Data warehouse terdistribusi dikembangkan berdasarkan konsep *gateway data warehouse*, sehingga memungkinkan *user* dapat langsung berhubungan dengan sumber data atau pemasok data maupun dengan pusat pengumpul data lainnya. Gambaran *user* atas data adalah gambaran logika karena data mungkin diambil dari berbagai sumber yang berbeda.

Pendekatan ini menggunakan teknologi *client/server* untuk mengambil data dari berbagai sumber, sehingga memungkinkan tiap departemen atau divisi untuk membangun sistem operasionalnya sendiri serta dapat membangun pengumpul data fungsionalnya masing-masing dan menggabungkan bagian-bagian tersebut dengan teknologi *client-*

server. Pendekatan ini akan menjadi sangat efektif bila data tersedia dalam bentuk yang konsisten dan *user* dapat menambahkan data tersebut dengan informasi baru apabila ingin membangun gambaran baru atas informasi.

Penerapan *data warehouse* terdistribusi ini memerlukan biaya yang sangat besar karena setiap pengumpul data fungsional dan sistem operasinya dikelola secara terpisah. Selain itu, supaya berguna bagi perusahaan, data yang ada harus disinkronisasikan untuk memelihara keterpaduan data.



Gambar 2.4 Distributed Data Warehouse

2.9 Kegunaan Data Warehouse

Menurut Williams (1998, p533) *data warehouse* mempunyai kegunaan sebagai berikut:

2.9.1 Pembuatan Laporan

Pembuatan laporan adalah salah satu kegunaan *data warehouse* yang paling umum. Dengan menggunakan *query-query* sederhana dalam *data warehouse*, dapat dihasilkan informasi pertahun, perkuartal, perbulan, atau bahkan perhari. *Query-query* tersebut digunakan dengan tujuan memperoleh jawaban atas pertanyaan-pertanyaan khusus, seperti kapan, siapa, di mana, dan sebagainya.

2.9.2 On-Line Analytical Processing (OLAP)

Data warehouse digunakan dalam melakukan analisis bisnis untuk menyelidiki kecenderungan pasar dan faktor-faktor penyebabnya karena dengan adanya *data warehouse*, semua informasi baik rincian maupun ringkasan yang dibutuhkan dalam proses analisis mudah didapat. Dalam hal ini, *data warehouse* merupakan *tool* yang handal untuk analisis data yang kompleks.

2.9.3 Data Mining

Penggunaaan *data warehouse* dalam pencarian pola dan hubungan data, dengan tujuan membuat keputusan bisnis bagi para pihak manajemen. Dalam hal ini, *software* dirancang untuk pola statistik dalam data untuk mengetahui kecenderungan yang ada, misalnya kecenderungan pasar akan suatu produk tertentu.

2.9.4 Proses Informasi Eksekutif

Data warehouse digunakan untuk mencari ringkasan informasi yang penting dengan tujuan membuat keputusan bisinis tanpa harus menjelajahi keseluruhan data. Dengan menggunakan *data warehouse*, segala laporan telah diringkas dan dapat pula diketahui rinciannya secara lengkap, sehingga mempermudah proses pengambilan keputusan. Informasi dan data pada laporan *data warehouse* menjadi sangat informatif bagi *user*, dalam hal ini adalah pihak eksekutif.

2.10 Perancangan Data Warehouse dengan Skema Bintang

Menurut Poe (1996, p33) skema bintang adalah suatu jenis spesifik dari perancangan *database* yang digunakan untuk mendukung proses analitis serta memiliki secara spesifik satuan tabel normalisasi. Skema bintang memiliki dua macam tabel, yaitu:

- Tabel fakta (*fact table*)

  Disebut juga tabel utama (*major table*), merupakan inti dari skema bintang dan berisi data aktual yang akan dianalisis (data kuantitatif dan transaksi). *Field-field* tabel fakta sering disebut *measure* dan biasanya dalam bentuk numerik. Selalu berisi *foreign key* dari masing-masing tabel dimensi. Tabel ini dapat terdiri dari banyak kolom dari ribuan baris data.

- Tabel dimensi (*dimension table*)

  Disebut juga tabel kecil (*minor table*), biasanya lebih kecil dan memegang data deskriptif yang mencerminkan dimensi suatu bisnis. Tabel dimensi

berisi data yang merupakan deskripsi lebih lanjut dari data yang ada pada tabel fakta.

2.10.1 Keuntungan Menggunakan Skema Bintang

Menggunakan skema bintang memberikan beberapa keuntungan yang tidak terdapat pada struktur relasional yang biasa. Skema bintang merupakan standar rancangan *data warehouse* karena (Poe, 1996, p121):

- Membangun rancangan *database* yang memberikan *response time* yang cepat.
- Memberikan rancangan yang dapat dimodifikasi dengan mudah atau ditambahkan sesuai dengan perkembangan dan pertumbuhan *data warehouse.*
- Paralel dalam rancangan *database*, bagaimana *user* biasanya memandang dan menggunakan data.
- Mempermudah pemahaman dan navigasi *metadata* baik untuk perancang maupun pemakai.
- Memperluas pilihan *front end data access tools*, karena beberapa produk yang memerlukan rancangan skema bintang.

2.10.2 Perancangan Skema Bintang

Menurut Poe (1996, p120) tujuan dari *database* pendukung keputusan dapat dicapai dengan perancangan *database* yang disebut sebagai skema bintang. Skema bintang merupakan suatu struktur

sederhana yang secara relatif terdiri dari beberapa tabel dan alur gabungan yang dirumuskan dengan baik. Perancangan *database* ini berlawanan dengan struktur normalisasi yang digunakan untuk *database* proses transaksi. *Database* ini menyediakan *response time query*, skema bintang yang dapat dibaca dan dipahami oleh analis, *end user*, bahkan bagi mereka yang belum terbiasa dengan struktur *database*.

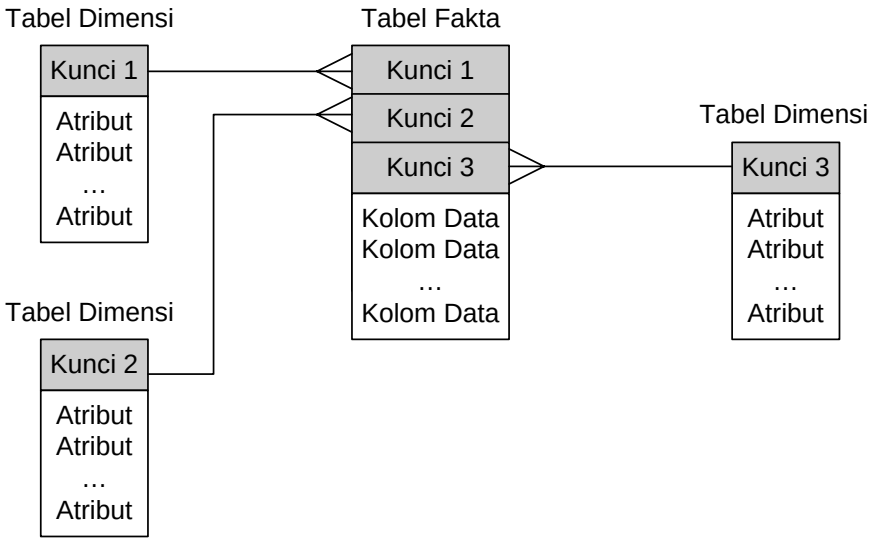
2.10.3 Skema Bintang Sederhana

Dalam skema bintang sederhana, setiap tabel harus mempunyai *primary key* yang bisa terdiri dari satu kolom atau lebih. *Primary key* tersebut membuat masing-masing baris menjadi unik. Di sini *primary key* dari tabel fakta terdiri dari satu atau lebih *foreign key*. *Foreign key* adalah kolom pada suatu tabel yang nilainya didefinisikan oleh *primary key* pada tabel yang lain.

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

Alasan memilih skema bintang sederhana antara lain:

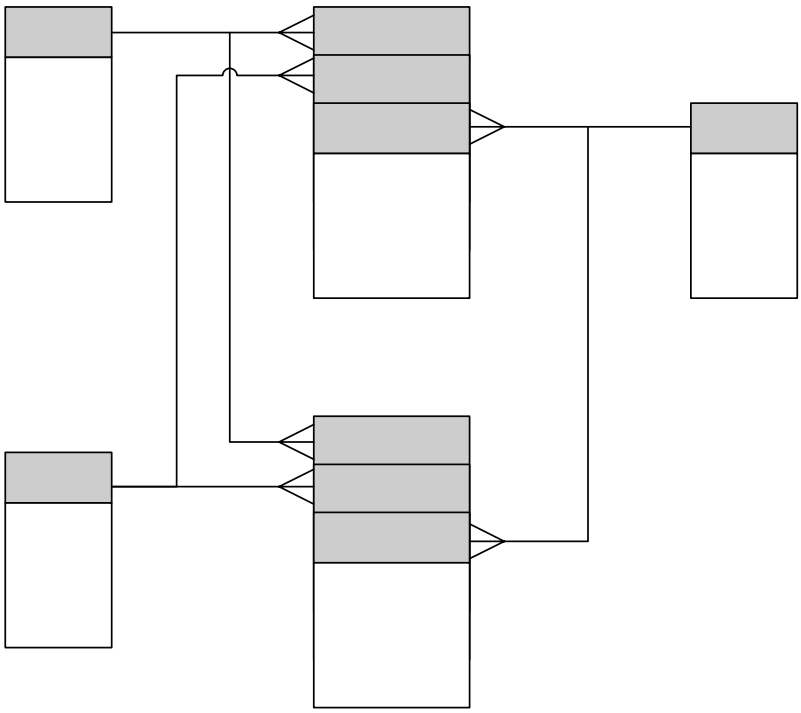
1. Menyediakan *response time* yang lebih baik
2. Struktur rancangan yang sederhana mudah dimengerti pemakai



Gambar 2.5 Skema Bintang Sederhana

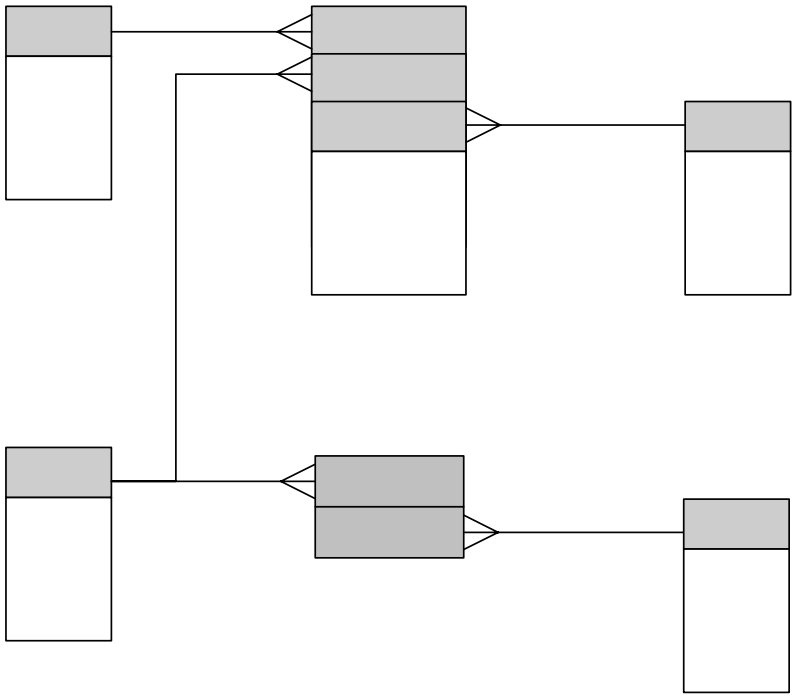
2.10.4 Skema Bintang dengan Banyak Tabel Fakta

Skema bintang juga dapat terdiri dari lebih dari satu tabel fakta. Hal ini terjadi karena mereka berisi kenyataan yang tidak saling berhubungan atau karena perbedaan waktu pemuatan data, di samping itu juga dapat meningkatkan penampilan. Tabel fakta yang banyak sering digunakan untuk menampung berbagai tingkat dari data yang bermacam-macam, terutama jika data tersebut dalam jumlah besar.



Gambar 2.6 Skema Bintang dengan Banyak Tabel Fakta

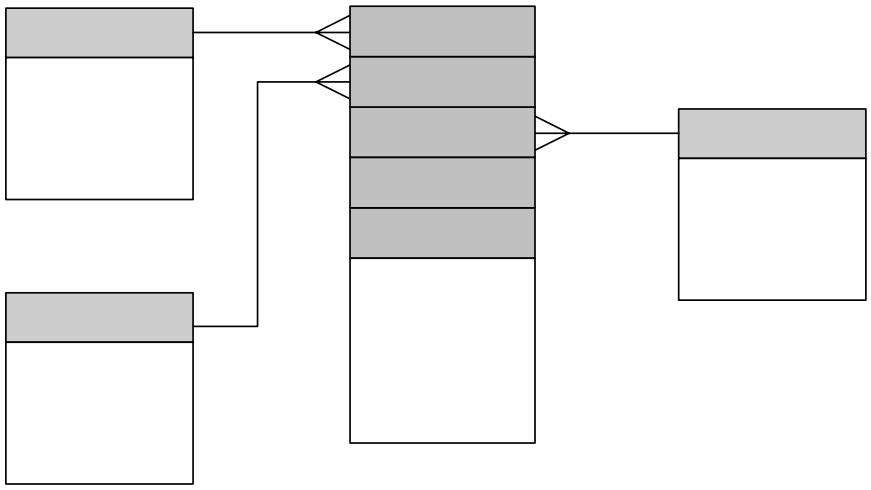
Kegunaan lain dari tabel fakta adalah untuk mendefinisikan hubungan *many to many* dari suatu tabel dimensi tertentu. Bentuk tabel seperti ini biasanya disebut tabel asosiasi. Tabel ini dibuat untuk menyelaraskan hubungan *many to many* di antara dimensi yang berbeda. Skema tersebut dapat dilihat pada gambar di bawah ini.



Gambar 2.7 Tabel Asosiasi

2.10.5 Skema Bintang Majemuk

Tabel fakta dalam skema majemuk memiliki dua kumpulan *foreign key*, yang pertama mengandung suatu referensi dengan tabel dimensi sedangkan sisanya adalah *primary key* yang merupakan gabungan dari satu atau lebih kolom yang menghasilkan suatu identifikasi unik untuk setiap barisnya. Bedanya skema bintang majemuk dengan skema bintang sederhana adalah saling tidak identiknya *primary key* dengan *foreign key* dalam skema bintang majemuk.



Gambar 2.8 Skema Bintang Majemuk

2.10.6 Skema Snowflake

Menurut Connoly (2002, p1080) skema *snowflake* merupakan skema yang berbeda dengan skema bintang karena tabel dimensi tidak berisi data denormalisasi.

Snowflake merupakan variasi lain dari skema bintang di mana tabel dimensi dari skema bintang diorganisasi menjadi suatu hierarki dengan melakukan normalisasi.

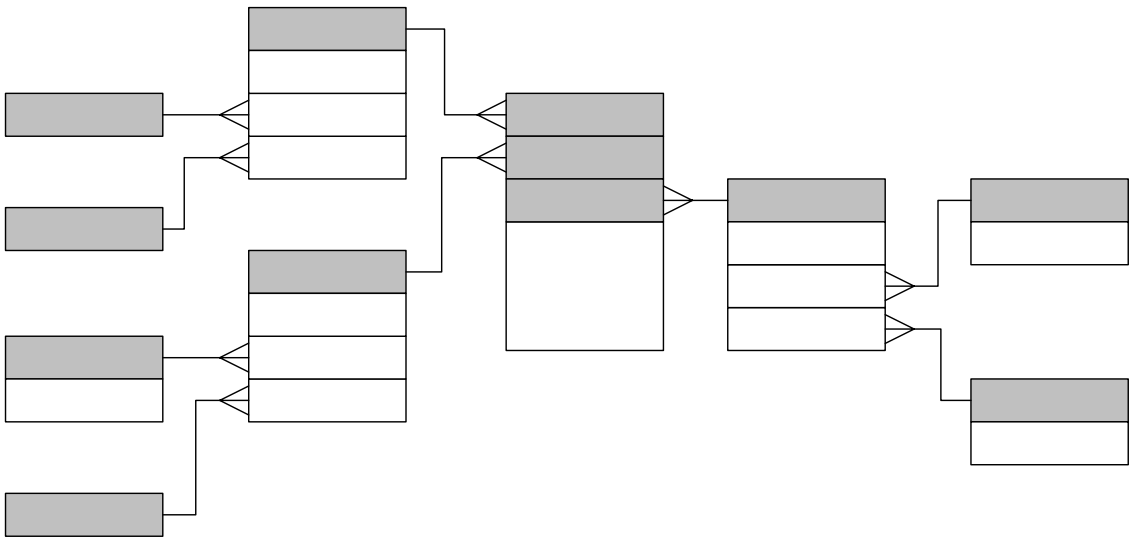
Prinsip dasar dari skema ini tidak jauh berbeda dari skema bintang. Penggunaan tabel dimensi sangatlah menonjol, karena itulah perbedaan mendasar dari skema bintang dan skema *snowflake*. Skema snowflake menggunakan beberapa tabel fakta dan tabel dimensi yang sudah mengalami normalisasi, sedangkan skema bintang menggunakan tabel dimensi yang masih denormalisasi. Skema *snowflake* dibuat

berdasarkan OLTP sehingga semua data akan termuat detail dalam setiap tabel fakta dan tabel dimensi.

Keuntungan dari skema *snowflake* adalah:

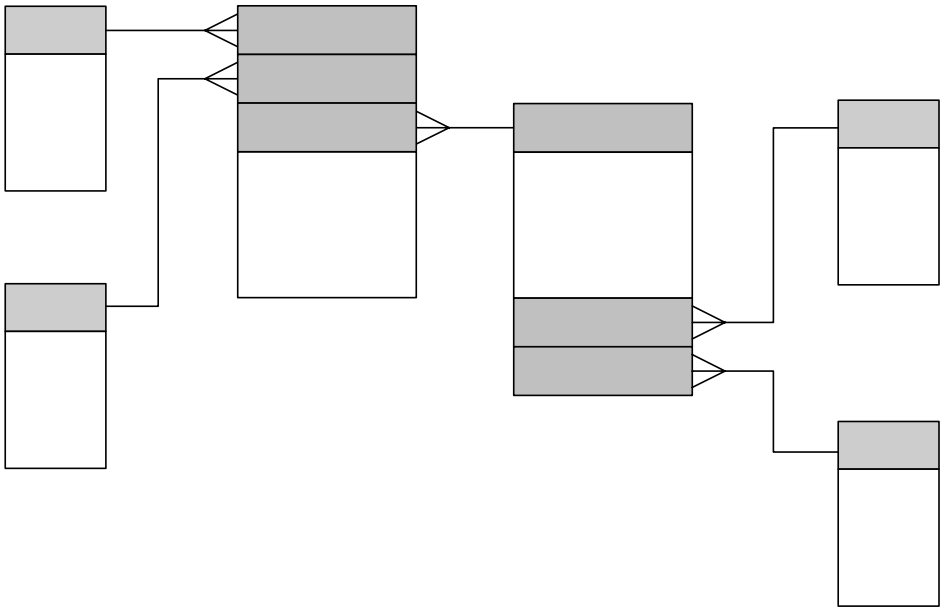
- Kecepatan memindahkan data dari data OLTP ke dalam metadata.
- Sebagai kebutuhan dari alat pengambil keputusan tingkat tinggi di mana dengan tipe yang seperti ini seluruh struktur dapat digunakan sepenuhnya.
- Banyak yang beranggapan lebih nyaman merancang dalam bentuk normal ketiga.

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 2.9 Skema Snowflake

Tabel dimensi mungkin mengandung *foreign key* yang mereferensikan *primary key* di tabel dimensi yang lain. Tabel dimensi yang direferensikan ini yang dinamakan *outboard table* atau *secondary dimension table*. Pada gambar terdapat skema bintang dengan *outboard table* atau *secondary dimension table*. Tabel dimensi 3 mempunyai dua *outboard table* yaitu tabel dimensi 4 dan tabel dimensi 5.



Gambar 2.10 Skema Bintang dengan Tabel Outboard

2.10.7 Agregasi

Menurut Poe (1996, p136) agregasi merupakan proses penghitungan data fakta selama pendefinisian atribut. Definisi dari agregasi biasanya mengandung makna penghitungan, perumusan informasi yang mendasari hubungan antar data yang terdapat dalam

sebuah tabel. Selain itu, agregasi juga dapat dibuat selama proses transformasi dan pemuatan data ke dalam *data warehouse*. Beberapa faktor yang mendorong pembuatan agregasi adalah untuk:

- Mempercepat waktu respon saat melakukan pencarian.
- Mengurangi jumlah dari penggunaan siklus CPU.

2.10.8 Denormalisasi

Menurut Poe (1996, p137) denormalisasi adalah suatu proses penggabungan tabel yang dilakukan dengan cermat dan hati-hati yang bertujuan untuk meningkatkan kinerja. Sebenarnya ini merupakan proses yang melanggar aturan dari *third normal form* (*3NF*).

Menurut Mannino (2001, p553) denormalisasi menggabungkan tabel-tabel sehingga lebih mudah untuk di-*query*. Denormalisasi merupakan kebalikan dari normalisasi. Denormalisasi berguna untuk meningkatkan kinerja *query* atau mengabaikan adanya *dependency* yang tidak menimbulkan anomali *storage* yang signifikan.

Keuntungan melakukan proses denormalisasi adalah (Poe, 1996, p139):

1. Mengurangi jumlah relasi yang terjadi antar tabel yang harus diproses pada saat pencarian sehingga akan meningkatkan kecepatan proses *query* data.
2. Memetakan struktur fisik *database* agar mudah dimengerti menurut model dimensi dari pemakai (*Dimensional Business Model*). Struktur

tabel yang dibuat sesuai keinginan pemakai memungkinkan terjadinya akses langsung yang sekali lagi akan meningkatkan kinerja.

Sedangkan kelemahan melakukan proses denormalisasi adalah:

1. Proses denormalisasi secara tidak langsung akan membuat redundansi data.
2. Proses denormalisasi memerlukan alokasi *memory* dan *storage* (tempat penyimpanan) yang besar.

2.11 Penjualan

Menurut Mulyadi (2001, p202), kegiatan penjualan terdiri dari transaksi penjualan barang dan jasa, baik secara kredit maupun secara tunai. Dalam transaksi penjualan kredit, jika order dari pelanggan telah dipenuhi dengan pengiriman barang atau penyerahan jasa, untuk jangka waktu tertentu, perusahaan memiliki piutang kepada pelanggannya. Dalam sistem penjualan secara tunai, barang atau jasa baru diserahkan oleh perusahaan kepada pembeli jika perusahaan telah menerima kas dari pembeli.

Menurut Mulyadi (2001, pp204-205), fungsi yang terkait pada penjualan kredit adalah:

1. Fungsi kredit

   Dalam transaksi kredit dengan kartu kredit, fungsi ini bertanggung jawab atas pemberian kartu kredit kepada pelanggan terpilih. Dalam sistem penjualan secara kredit dengan kartu kredit, fungsi kredit tidak diperlukan

lagi otorisasinya, karena otorisasi pemberian kredit sudah tercermin dari kartu kredit yang ditunjukkan oleh pelanggan pada saat melakukan pembelian.

2. Fungsi penjualan

   Dalam sistem penjualan dengan kartu kredit ini, fungsi penjualan bertanggung jawab melayani kebutuhan barang pelanggan. Fungsi penjualan mengisi faktur penjualan kredit untuk memungkinkan fungsi gudang dan fungsi pengiriman melaksanakan penyerahan barang kepada pelanggan.

3. Fungsi gudang

   Dalam sistem penjualan ini, fungsi gudang menyediakan barang yang diperlukan oleh pelanggan sesuai dengan yang tercantum dalam tembusan faktur penjualan kredit yang diterima dari fungsi penjualan.

4. Fungsi pengiriman

   Fungsi ini bertanggung jawab untuk menyerahkan barang yang kuantitas, mutu, dan spesifikasinya sesuai dengan yang tercantum dalam tembusan faktur penjualan kartu kredit yang diterima dari fungsi penjualan.

5. Fungsi akuntansi

   Fungsi ini bertanggung jawab untuk transaksi bertambahnya piutang kepada pelanggan ke dalam kartu piutang berdasarkan faktur penjualan kartu kredit yang diterima dari fungsi pengiriman.

6. Fungsi penagihan

   Fungsi ini bertanggung jawab untuk membuat surat tagihan secara periodik kepada pemegang kartu kredit.