BAB 2

TELAAH LITERATUR

A. *Business intelligence*

Inovasi *data processing* yang terus berkembang menyebabkan makin banyak informasi yang disimpan dalam format yang semakin detail. Hasilnya yaitu, terdapat kebutuhan untuk mengecilkan dan menstrukturkan data sehingga dapat lebih mudah untuk dianalisis. Analisis data ini diperlukan untuk menghasilkan *business intelligence* dari data-data yang telah dikumpulkan.

Definisi BI menurut laporan Gartner Group (1996) dalam SAP (2006): "*Business intelligence* merupakan kategori luas dari aplikasi dan teknologi untuk mengumpulkan, menyimpan, menganalisis, dan menyediakan akses ke data untuk membantu perusahaan mengambil keputusan bisnis yang lebih baik. Dalam aplikasi BI terdapat aktivitas yang terdiri dari *decision support system, query and reporting, online analytical processing* (OLAP), *statistical analysis, forecasting,* dan *Data Mining*."

Business intelligence (BI) merupakan sistem dan aplikasi yang berfungsi untuk mengubah data-data dalam suatu perusahaan atau organisasi ke dalam bentuk informasi. Aplikasi ini melakukan analisis data-data menjadi informasi dan kemudian menggunakan informasi tersebut untuk mendukung keputusan dan perencanaan organisasi.

BI dapat membantu suatu perusahaan mendapatkan pengetahuan yang jelas mengenai faktor-faktor yang mempengaruhi kinerja perusahaan sehingga

dapat membantu organisasi dalam pengambilan keputusan serta sekaligus meningkatkan keunggulannya (*competitive advantage*). BI juga dapat membantu suatu perusahaan dalam menganalisis perubahan tren yang terjadi sehingga akan membantu organisasi menentukan strategi yang diperlukan dalam mengantisipasi perubahan tren tersebut.

B. *Data Warehouse*

1. Definisi *data warehouse*

Inmon (2005, p29) mendefinisikan *data warehouse* sebagai koleksi data yang bersifat *subject-oriented, integrated, nonvolatile,* dan *time-variant* digunakan untuk mendukung pengambilan keputusan manajemen. Data dikumpulkan dari berbagai aplikasi yang ada. Data yang telah dikumpulkan tersebut kemudian divalidasi dan direstrukturisasi lagi, untuk selanjutnya disimpan dalam *data warehouse*. Pengumpulan data ini memungkinkan para pengambil keputusan untuk pergi hanya ke satu tempat untuk mengakses data yang ada di organisasinya.

Sebagai tambahan, *data warehouse* memuat proses *extraction, transformation, and loading* (ETL) *solution,* sistem *online analytical processing* (OLAP), *client analysis tools,* dan aplikasi lainnya yang mengatur proses pengumpulan dan pengiriman data ke user.

Data warehouse sering menjadi bagian inti dari infrastruktur *business intelligence* (BI) organisasi. *Data warehouse* adalah kumpulan dari basis data yang terintegrasi dan *subject oriented* yang didesain untuk mendukung *Decision*

Support Systems (DSS). Menurut Inmon dalam Connolly (2005, p1151) karakteristik utama dari *data warehouse* antara lain:

a. *Subject oriented*, data disusun dan diorganisasikan berdasarkan bagaimana *user* menggunakannya.
b. *Integrated*, semua sifat ketidakkonsistenan yang disebabkan oleh kesepakatan penamaan dan representasi nama serta perbedaan format dihilangkan .
c. *Time- variant*, data tidak bersifat *current* tapi lebih bersifat *time series*.
d. *Non-volatile*, data disimpan dalam dalam format *read-only* dan tidak dapat diubah.

2. Perbedaan data operasional dan *data warehouse*

Data operasional dan *data warehouse* memiliki beberapa perbedaan. Ponniah (2001, p11) memaparkan perbedaan tersebut di antaranya:

a. Dalam sistem operasional, data yang disimpan menunjukkan data yang sekarang (*Current Values*), sedangkan apa yang tersimpan dalam *data warehouse* adalah data *archived* atau data sebagai hasil penurunan- penurunan atau *summarized* dari suatu data besar.
b. Dilihat dari struktur keduanya juga berbeda, data operasional didesain dengan sedemikian sehingga optimum untuk melakukan transaksi, sedangkan dalam *data warehouse* dioptimalkan untuk menangani *query* yang rumit.
c. Penggunaan data operasional bersifat perulangan, sedangkan *data warehouse* bersifat *ad-hoc*, acak dan *heuristic*.
d. Ditinjau dari frekuensi mengaksesnya, data operasional jauh lebih sering diakses dibandingkan *data warehouse*.

e. Jika dalam data operasional akses terhadap datanya dapat berupa *read*, *update* atau bahkan *delete*, maka dalam *data warehouse* data hanya dapat di-*read*.

f. Jumlah pengguna juga berbeda, jika operasional digunakan oleh orang banyak, sedangkan *data warehouse* oleh beberapa orang saja untuk mendukung analisis data.

Perbedaan tersebut adalah karena memang adanya perbedaan tujuan dari dibuatnya sistem. Data Operasional digunakan untuk menjalankan operasional bisnis dengan cara memasukan data-data kedalamnya. Tujuan dari *data warehouse* adalah mendapatkan informasi yang bisa mendukung pengambilan keputusan. Pengguna bisa mengetahui produk apa yang menjadi favorit, daerah mana yang banyak mengalami masalah penjualan, mengapa bisa terjadi masalah, yang pada prinsipnya untuk menangkap peluang dan mengurangi resiko dalam menjalankan bisnis.

3. Keuntungan *data warehouse*

Menurut Connolly (2005, p1152) implementasi *data warehouse* yang tepat dapat memberikan keuntungan- keuntungan antara lain:

a. Meningkatkan produktifitas dari pengambil keputusan perusahaan

Data warehouse meningkatkan produktifitas dari pengambil keputusan perusahaan dengan membuat integrasi *database* yang konsisten, berorientasi data *subject* dan *historical*. *Data warehouse* mengintegrasikan data dari banyak sistem yang tidak kompatibel menjadi suatu bentuk yang menyediakan satu tampilan yang konsisten mengenai perusahaan. Proses transformasi data menjadi informasi penting yang dilakukan oleh *data warehouse* mengijinkan

pengambil keputusan untuk melakukan analisis sesuai dengan fakta secara akurat dan konsisten.

b. Potensi *Return On Investment* (ROI) yang besar

Suatu perusahaan akan mengeluarkan sumber daya yang cukup besar untuk mengimplemtasikan *data warehouse* dan pengeluaran yang berbeda-beda sesuai dengan variasi solusi teknikal yang akan diterapkan pada perusahaan. Suatu studi oleh *International Data Corporation* (IDC) pada tahun 1996 melaporkan bahwa rata-rata tiga tahun *return of investment* (ROI) dalam *data warehouse* mencapai 401% dengan lebih dari 90% dari perusahaan yang disurvei mencapai lebih dari 40% ROI, setengah dari perusahaan mencapai lebih dari 160% ROI, dan seperempat lebih mendapat lebih dari 600% ROI.

c. *Competitive Advantage*

Return on investment yang besar dari perusahaan yang berhasil mengimplementasikan suatu *data warehouse* adalah bukti dari sangat besarnya *competitive advantage* yang dapat diperoleh dengan menggunakan teknologi ini. *Competitive advantage* diperoleh dengan mengijinkan si pengambil keputusan untuk mengakses data tersembunyi yang sebelumnya tidak tersedia, tidak diketahui, dan tidak dimanfaatkan seperti data mengenai pelanggan, tren, dan permintaan.

Berikut adalah contoh-contoh peluang yang ada karena ketersediaan informasi strategis menurut Ponniah (2001, p6):

a. Ketersediaan informasi strategis di salah satu bank terbesar di United States dengan asset $250 miliar memberikan kesempatan pada *user* untuk membuat keputusan yang cepat untuk mempertahankan nilai mereka pada pelanggan.
b. Pada kasus organisasi pelayanan kesehatan yang besar, terjadi peningkatan yang signifikan program-program pelayanan kesehatan yang terealisasi, dengan hasil 22% penurunan kunjungan *emergency room*, 29% penurunan terhadap pasien anak asma, diabetes dan peningkatan tingkat vaksinasi dan lebih 100.000 *performance report* dibuat untuk pasien dan apoteker.
c. Komunitas apoteker yang bersaing dalam skala nasional dengan lebih dari 800 *franchise* apotik mengerti betul apa yang dibutuhkan oleh pelanggan, hasilnya penurunan level *inventory*, meningkatkan efektifitas promosi dan marketing, meningkatkan keuntungan bagi perusahaan.

4. Kategori data pada *data warehouse*

Untuk memahami *data warehouse* lebih dalam, ada dua aspek penting yang harus di pahami yaitu pertama adalah memahami tipe spesifik (*classification*) dari data yang akan disimpan di *data warehouse* dan kedua mengenai tahapan proses dalam pembuatan *data warehouse*. Mengenai kategori pada *data warehouse*, kategori ini diakomodasikan berdasarkan *time-dependent data sources*.

Adapun klasifikasinya adalah sebagai berikut ini:

a. *Old detail data* merupakan data yang secara frekuensi, sudah jarang digunakan sehingga data tidak disimpan pada penyimpanan utama.
b. *Current detail data* merupakan data yang merefleksikan kondisi terkini dari suatu organisasi maupun perusahaan karena itu *current detail data* memiliki volume yang besar dan disimpan pada penyimpanan utama.
c. *Lightly summarized data* merupakan *summaries* dari data dengan level *granularity* detail.
d. *Highly summarized data* merupakan *summaries* data yang lebih *compact* dan berdasarkan pada *lightly summarized data*.
e. *Metadata* merupakan direktori data atau informasi tentang data dan disimpan pada penyimpanan yang terpisah dari tipe data lainnya.

5. Bentuk *Data warehouse*

Menurut Inmon (2005, pp193-194), *data warehouse* memiliki dua bentuk yaitu *data warehouse* terpusat dan *data warehouse* terdistribusi.

a. *Data warehouse* terpusat

   Data warehouse jenis ini adalah *data warehouse* yang dibangun dengan mengumpulkan semua data dari berbagai sumber dan disimpan dalam sebuah tempat yang terpusat, kemudian data tersebut disebarkan ke dalam fungsi atau bagian terkait pada perusahaan tersebut sesuai dengan kebutuhan.

b. *Data warehouse* terdistribusi

Jenis *data warehouse* ini menggunakan *gateway* yang berfungsi sebagai jembatan penghubung antara *data warehouse* dengan *workstation* yang menggunakan sistem beraneka ragam. Jadi perusahaan dapat mengakses sumber data yang berada di luar lokasi perusahaan.

6. Arsitektur *data warehouse*

Arsitektur dalam *data warehouse* mencakup komponen-komponen penyusun *data warehouse*, baik itu *software* ataupun *hardware*. Untuk mempermudah proses pembangunan suatu *data warehouse* diperlukan pemilihan arsitektur yang tepat dan pemahaman yang baik terhadap arsitektur *data warehouse*. Berikut adalah komponen-komponen penyusun *data warehouse* (Gambar 2.1.).

a. *Operational Data Source*, dapat bersumber dari *mainframe* data operasional, data departemen pada *file system*, *private data* pada *workstation* dan *private server*, *external system* seperti internet, dan *database* yang terhubung dengan *supplier* atau *customer*.
b. *Operational Data Store*, merupakan penyimpanan dari data operasional yang aktual dan terintegrasi, digunakan untuk melakukan analisis data.
c. *Load Manager*, melakukan semua operasi yang terkait dengan *extraction* dan *loading* data ke dalam *warehouse*.
d. *Warehouse Manager*, melakukan semua operasi yang terkait dengan manajemen data di dalam *warehouse*.
e. *Query Manager*, melakukan semua operasi yang terkait dengan manajemen *user queries*.

f. *Detailed Data*, menyimpan data yang bersifat detail dalam *database schema*.

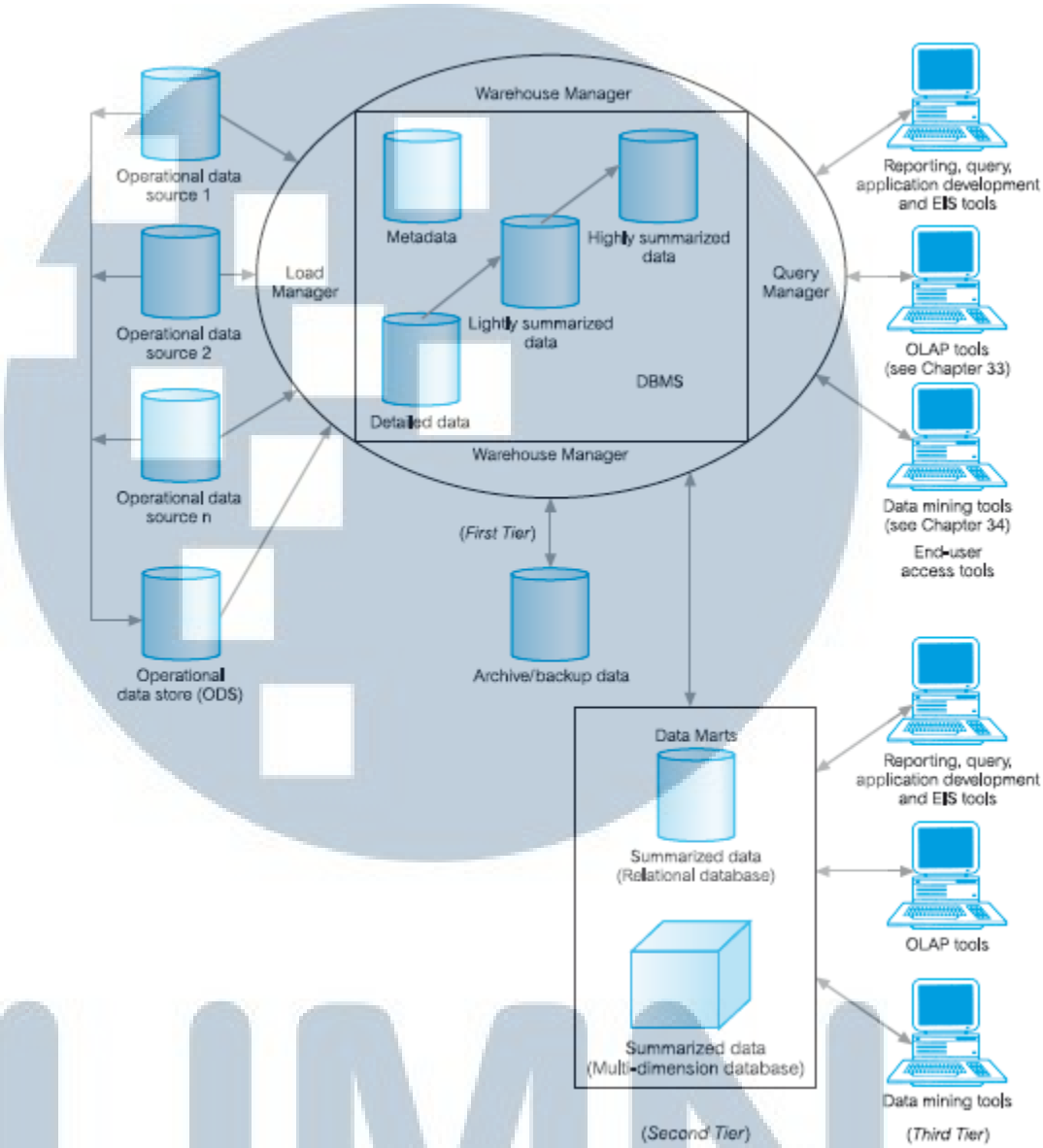
g. *Lightly and Highly Summarized Data*, menyimpan data yang sudah di-*summarized* atau diagregasi dalam *warehouse manager*.

h. *Archive/Backup Data*, menyimpan data yang digunakan untuk proses *archiving* dan *backup*.

i. *Metadata*, data mengenai data dalam *data warehouse* menyimpan detail dari proses yang terjadi di dalam *warehouse* seperti, proses *extraction* dan *loading*, proses manajemen *warehouse*, dan proses manajemen *query*.

j. *Data Mart*, merupakan bagian dari *data warehouse* yang mendukung kebutuhan informasi dari departemen atau fungsi bisnis tertentu.

k. *End –User Access Tools*, dapat dikategorikan menjadi: *data reporting and query tools*, *application development tools*, *executive information system tools*, *Online Analytical Processing* (OLAP) *tools*, dan *data mining tools*.



Gambar 2. 1 Arsitektur *data warehouse* (Connoly, 2005, p1172)

7. Proses ETL

Proses ETL (*extract, transform, load*) adalah proses yang digunakan dalam memproses data sebelum dimasukkan ke dalam suatu *data warehouse* yang akan dilakukan oleh *load manager*. Proses ini dilakukan untuk menstandarisasikan data yang akan digunakan pada *data warehouse*. ETL adalah langkah kritis dalam pembuatan *data warehouse*. Proses ETL dilakukan secara periodik dan otomatis.

Penjelasan dari masing-masing adalah sebagai berikut:

a. *Extract*

Proses *extract* adalah proses mengekstrak (*extracting*) dan pengambilan data dari sumber pada sistem untuk diload ke *data warehouse*. Sumber data dapat diperoleh secara alternatif melalui ODS (*Operational Data Storage*). Data harus dikonstruksi ulang sebelum dimasukkan ke dalam *data warehouse*.

Ekstraksi dilakukan dari sumber data yang digunakan melalui proses pemilihan data yang kemudian disimpan pada *temporary database*. Penempatan *temporary database* diletakkan pada penyimpanan *database*, mesin dan platform yang sama dengan *data warehouse* agar proses ETL dapat dilakukan lebih cepat dan tidak mengganggu proses operasional.

Proses *extract* ini melibatkan proses:

1) *Cleansing data*, yaitu proses pembersihan data kotor. Data kotor dalam hal ini adalah data berkualitas rendah seperti ketidakkonsistenan penulisan nama, kode id, duplikasi data, data tidak lengkap dan lain-lain.

2) Restrukturisasi data pada *data warehouse* untuk memahami kebutuhan yang ada. Contoh: penambahan atau pengurangan *fields* dan *denormalisasi data*
3) Memastikan sumber data konsisten dengan data yang sudah ada di dalam *data warehouse*

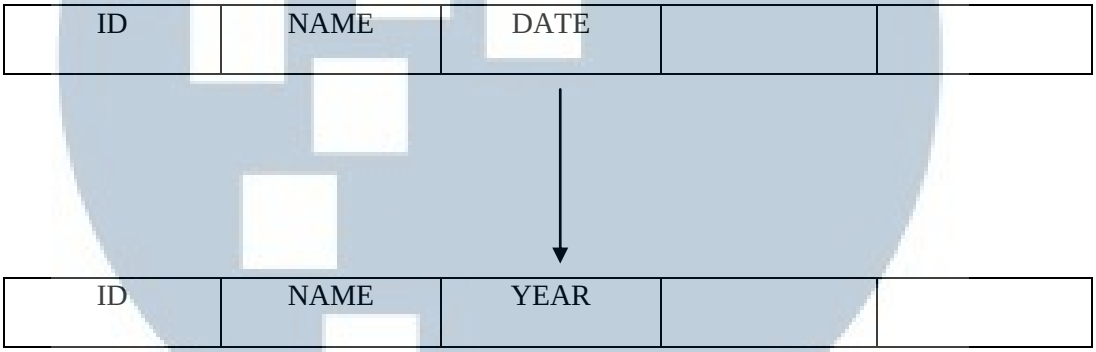
b. *Transform*

Proses *transform* adalah proses pengubahan data, dimana data yang diperoleh dari proses extract (dalam format operasional) menjadi data dalam bentuk *data warehouse*. Proses transformasi yang dilakukan dibagi berdasarkan level, yaitu fungsi *recordlevel*, dan fungsi *field-level*.

Fungsi record-level melibatkan proses *Summarizing the data*, dengan cara pemilihan (*selection*), proyeksi (*projecting*), penggabungan (*joining*), normalisasi (*normalization*), agregasi (*aggregation*) dan *grouping* data relasional menjadi *views* yang lebih nyaman dan berguna bagi pengguna.

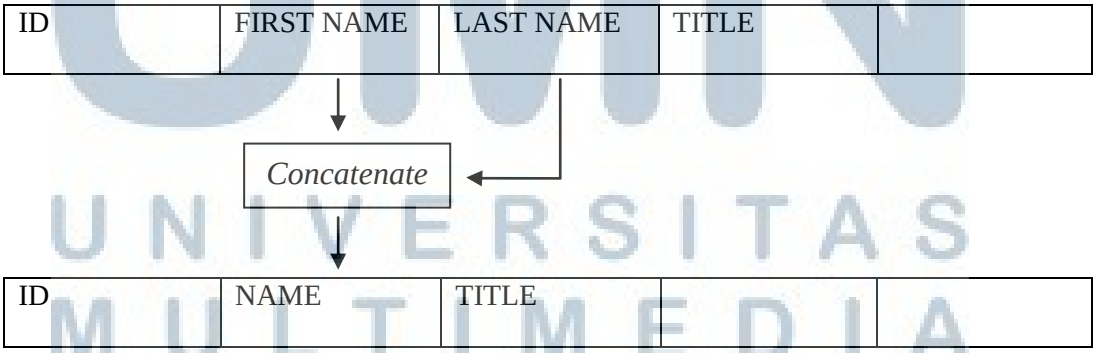
Fungsi *field-level* terdiri dari *single-field* dan *multi-field.* Transformasi *single-field* mengubah satu *field* menjadi *field* yang lain sedangkan *multi-field* mengubah satu *field* atau lebih menjadi *field* lain dan dapat berlaku sebaliknya.

Pada Gambar 2.2 di bawah dapat dilihat contoh proses transformasi *single field* dari *field* DATE menjadi *field* YEAR dengan transformasi *time conversion*, yaitu mengubah data pada field DATE dari format dd.mm.yyyy menjadi yyyy dengan nama field YEAR.



Gambar 2. 2 Contoh Transformasi *Single Field*

Pada Gambar 2.3 di bawah dapat dilihat contoh proses transformasi *multi field* dengan menggabungkan *field* FIRST NAME dan LAST NAME menjadi *field* NAME melalui proses *concatenate*, yaitu menggabungkan data dari dua *field* menjadi satu.



Gambar 2. 3 Contoh Transformasi *Multi Field*

c. *Load*

Proses *load* adalah proses tahapan terakhir dari proses ETL. Pada proses ini akan dilakukan proses pemuatan data dari proses *transform* ke dalam suatu *data warehouse*. Pada proses ini dilakukan juga proses *indexing* untuk memberikan indeks ke masing-masing data untuk mempercepat proses *query*. Terdapat dua mode *loading* ke dalam *data warehouse* yaitu *refresh* dan update. Mode *refresh* yaitu proses menuliskan kembali keseluruhan data di dalam *data warehouse* pada suatu interval waktu. Sedangkan untuk mode *update* yaitu suatu proses untuk meng-*update* (tidak menghapus atau menimpa data lain) data yang berubah ke tempat tujuan pada *data warehouse*. Mode *refresh* digunakan pertama kali ketika *data warehouse* berjalan dan hendak dimuat, sedangkan mode *update* umumnya digunakan ketika pemeliharaan data atau ketika *data warehouse* sedang *running*.

8. Desain *data warehouse*

Komponen *database* dari *data warehouse* dapat didefinisikan menggunakan teknik yang disebut *dimensional modeling*. Menurut Connoly (2005, p1183) pengertian dari *dimensional modeling* adalah suatu teknik desain secara *logical* yang memiliki sasaran untuk mempresentasikan data dengan bentuk standar serta intuitif sehingga dapat diakses secara cepat. Setiap *dimensional model* memiliki komposisi yang terdiri dari satu tabel dengan *composite key* yang dinamakan *fact table* dan sekumpulan set tabel yang lebih kecil yang dinamakan *dimension table.*

Dimensional modelling memiliki beberapa struktur skema, salah satunya yaitu skema bintang (*Star Schema*) yang akan digunakan dalam penelitian ini. Dalam Connoly (2005, p1183) dijelaskan bahwa pada skema bintang, struktur *logical* memiliki *fact table yang* mengandung data fakta, dikelilingi oleh *dimension tables* yang mengandung referensi data (yang bisa di-*denormalisasi*). Contoh skema bintang dapat dilihat pada Gambar 2.4.

Skema bintang dapat meningkatkan kecepatan, karena denormalisasi informasi ke dalam sebuah tabel dimensi. Denormalisasi sangat disarankan bila terdapat sejumlah entitas yang berhubungan dengan tabel dimensi yang sering diakses, untuk menghindari perlunya menghubungkan tabel lain untuk mengakses atribut tersebut. Namun denormalisasi tidak disarankan jika data tambahan tidak sering diakses, karena dalam kasus itu menghubungkan tabel lain tidak akan terlalu memperlambat performa *query*.

Gambar 2. 4 *Simple Star Schema* (SAP AG, 2006)

Keuntungan dari penggunaan skema bintang menurut Ponniah (2001, pp220-222):

a. Mudah dimengerti

   Skema bintang merefleksikan bagaimana pengguna berpikir dan mencari data untuk *query* dan analisis. Skema bintang mendefinisikan hubungan antara tabel sama seperti bagaimana pengguna memvisualisasikan hubungannya pada umumnya sehingga mudah dimengerti oleh pengguna.

b. Mengoptimalkan navigasi

   Hubungan-hubungan antara tabel dalam skema bintang sederhana, sehingga navigasinya dapat menjadi optimal. Walaupun query yang digunakan terlihat kompleks, tetapi navigasi di dalamnya tetap sederhana.

c. Cocok untuk pemrosesan *query*

   Struktur skema bintang adalah adalah struktur yang berpusat pada *query*, sehingga menjadikan skema bintang paling cocok untuk pemrosesan *query*. Setiap *query* dieksekusi dengan terlebih dahulu memilih baris dari tabel dimensi menggunakan filter dari parameter *query*, lalu menemukan baris yang terhubung dengannya dalam tabel fakta. Hal ini memungkinkan karena hubungan yang sederhana dan tidak adanya tabel penghubung lainnya dari tabel dimensi ke tabel fakta.

Penjelasan lebih detail mengenai tabel dalam skema bintang adalah sebagai berikut:

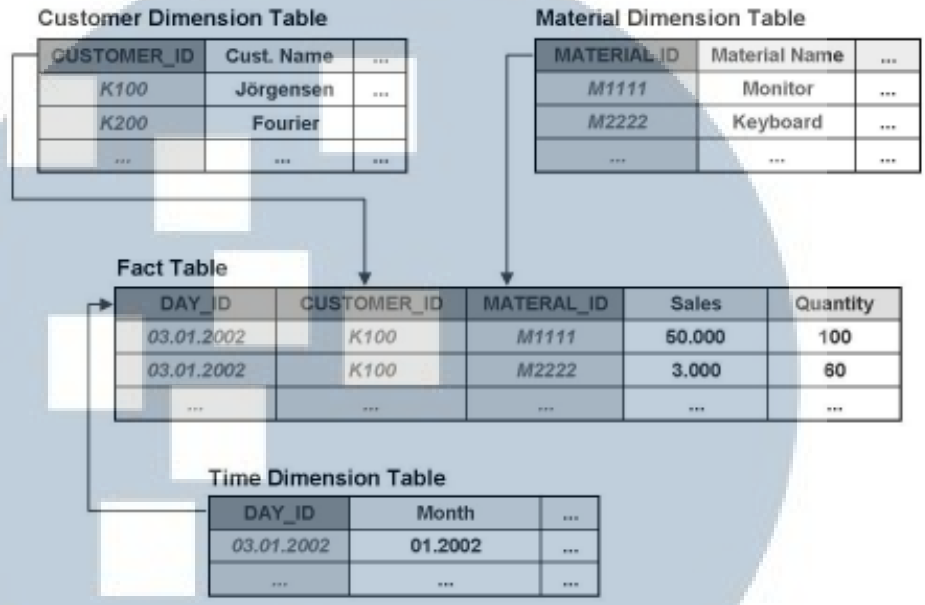
a. Tabel Dimensi (*Dimension Table*)

Tabel yang berisikan penjelasan deskriptif dari sebuah bisnis. Umumnya tabel dimensi adalah berupa teks dan dapat menjelaskan tabel dimensi itu sendiri. Walaupun sebuah data berupa angka, data tersebut dapat dimasukkan ke tabel dimensi asalkan data tersebut merupakan deskripsi dan bukan pengukuran. Setiap tabel dimensi diidentifikasi dengan sebuah *primary key*.

b. Tabel Fakta (*Fact Table*)

Tabel fakta merupakan tabel utama yang mengandung angka dari hasil perhitungan bisnis. Tabel fakta memiliki dua atau lebih *foreign key* sebagai penghubung ke *primary key* dari tabel-tabel dimensi. *Primary key* pada tabel fakta itu sendiri merupakan gabungan dari *foreign key* dan dikenal sebagai *composite key*.

Setiap tabel fakta dalam konsep pemodelan dimensional memiliki sebuah *composite key*, dan sebaliknya, setiap tabel yang memiliki *composite key* merupakan tabel fakta.



Gambar 2. 5 *Relationship Table Star Schema* (SAP AG, 2006)

Pada Gambar 2.5 menunjukkan hubungan antara satu tabel fakta dan tiga tabel dimensi. Pada tabel utama (*fact table*) terdapat sebuah *composite key* yang terdiri dari tiga *primary key*, yaitu *CUSTOMER*_ID, DAY_ID, dan MATERIAL_ID dimana masing-masing *primary key* tersebut merupakan *foreign key* dari tabel-tabel dimensi yang berhubungan dengan tabel fakta.

Struktur skema lainnya, yaitu skema *snowflake* dan skema *starflake*. Menurut Connolly (2005, p1184), skema *snowflake* adalah variasi dari skema bintang dimana tabel dimensi tidak berisikan data hasil denormalisasi. Sehingga tabel dimensi dapat memiliki dimensi lagi.

Skema *starflake* merupakan struktur campuran antara skema bintang dan skema *snowflake* (Connolly, 2005, p1185). Dalam skema *starflake*, beberapa tabel dimensi dapat disajikan dalam bentuk skema bintang maupun skema *snowflake* untuk memenuhi kebutuhan *query* yang berbeda-beda.

9. Metodologi data warehouse

Dalam teori *data warehouse* terdapat dua metodologi yang umum dalam proses perancangan *data warehouse*, yaitu metodologi *top-down design* oleh Inmon dan metodologi *bottom-up design* oleh Kimball.

Pada metodologi *top-down design*, perancangan *data warehouse* dimulai dengan mendefinisikan arsitektur *data warehouse* perusahaan secara menyeluruh baru dilanjutkan dengan perancangan *data mart* untuk departemen spesifik. Berikut merupakan kelebihan dari metodologi *top-down design* menurut Ponniah (2001, p26):

Kelebihan	Kekurangan
Data dapat dilihat secara menyeluruh dari *view* perusahaan	Menghabiskan waktu yang lebih lama meskipun dengan cara perulangan
Arsitektur *data warehouse* bersifat inheren, bukan gabungan dari *data mart* yang berbeda	Resiko kegagalan yang tinggi
Penyimpanan data di satu tempat yang terpusat	Membutuhkan *cross-functional level* yang tinggi
Kontrol dan aturan yang terpusat	Pengeluaran yang besar tanpa bukti dari konsep
Hasil dapat diperoleh dengan cepat apabila implementasi dilakukan secara berulang.	

Tabel 2. 1 Kelebihan dan kekurangan metodologi *top-down design*

Pada metodologi *bottom-up design*, perancangan *data warehouse* dimulai dengan mendefinisikan *data mart* satu per satu dimulai dari departemen atau proses bisnis yang dianggap paling penting kemudian setiap *data mart* yang sudah dirancang akan diintegrasikan untuk membentuk satu *data warehouse*. Berikut merupakan kelebihan dari metodologi *bottom-up design* menurut Ponniah (2001, p27):

Kelebihan	Kekurangan
Implementasi yang lebih cepat dan mudah	Setiap *data mart* memiliki *view* yang sempit dari data
Return on investment dan bukti konsep yang baik	Dapat menyebabkan redundansi data tersebar di setiap *data mart*
Resiko kegagalan yang lebih kecil	Dapat menyebabkan ketidakkonsistenan data secara terus menerus
Data marts dapat dibuat mulai dari bagian yang paling penting	Pertumbuhan *data mart* dapat menyebabkan tampilan data menjadi sulit dikelola
Memungkinkan tim proyek untuk berkembang	

Tabel 2. 2 Kelebihan dan kekurangan metodologi *bottom-up design*

Pada penelitian ini akan metodologi yang digunakan untuk perancangan *data warehouse* adalah metodologi *bottom-up design* oleh Kimball dengan pertimbangan sebagai berikut:

a. Peneliti tidak merancang *data warehouse* untuk seluruh departemen perusahaan, tetapi spesifik untuk satu departemen.
b. Metodologi *bottom-up design* merupakan solusi yang paling cocok dalam perancangan *data warehouse* dengan waktu penelitian yang terbatas.

C. *Dashboard*

Sebuah *dashboard* adalah tampilan visual dari informasi terpenting yang diperlukan untuk mencapai satu atau lebih objektif, dikonsolidasikan dan diatur dalam sebuah layar sehingga informasi dapat dimonitor dalam sekilas. Seperti *dashboard* mobil yang menyediakan semua informasi penting yang diperlukan untuk menjalankan mesin secara sekilas, sebuah *Business intelligence dashboard* melayani dengan tujuan yang sama; apakah digunakan untuk mengambil keputusan strategis untuk perusahaan besar atau menjalankan operasi harian tim, atau untuk mengerjakan tugas yang hanya melibatkan satu orang. Tujuan *dashboard* adalah agar seseorang dapat secara efisien terhubung dengan informasi yang diperlukan untuk melakukan pekerjaannya.

Selain dari hal diatas, terdapat beberapa atribut untuk menunjang *dashboard* agar dapat bekerja secara efektif. Atribut-atribut tersebut adalah:

1. *Summaries* tingkat tinggi. Informasi yang ditampilkan dalam sebuah *dashboard* harus mengandung informasi *summaries* tingkat tinggi agar dapat dikomunikasikan secara cepat. Informasi tersebut kemudian dapat dianalisis lebih lanjut melalui proses *drill down*.
2. Mekanisme yang ringkas, jelas dan intuitif. Mekanisme dalam penampilan data dan informasi pada *dashboard* dilakukan secara ringkas, jelas dan intuitif melalui presentasi grafik yang dapat menampilkan informasi yang dibutuhkan oleh pengguna.
3. Kustomisasi. Informasi pada *dashboard* dirancang untuk suatu kebutuhan spesifik dari seseorang atau suatu grup atau suatu fungsi.

Gambar 2. 6 Contoh *Dahboard* dari SAP Business Object Xcelcius

D. *Loyalty Program* dan *Frequent Flyer Program*

Loyalitas didefinisikan Oliver dalam Khokhar (2011, p10176) sebagai komitmen yang tinggi untuk membeli kembali suatu produk atau jasa yang disukai di masa mendatang, di samping pengaruh situasi dan usaha pemasar dalam mengubah perilaku. Dengan kata lain konsumen akan setia untuk melakukan pembelian ulang secara terus-menerus.

Menurut Shoemaker dan Lewis (1998) dalam Wijaya (2005, p25), *loyalty program* adalah program yang ditawarkan pada konsumen untuk membangun ikatan terhadap merek/*brand* tertentu. Lebih lanjut Wijaya (2005, p26) menyatakan bahwa sebagian besar konsumen melakukan pembelian ulang dalam rangka menambah keuntungan yang ditawarkan kemudian *redeem* dengan menggunakan *reward* yang telah dikumpulkan. Konsumen loyal terhadap

program, bukan kepada perusahaannya. Tidak ada hubungan langsung antara program dengan ikatan emosional konsumen terhadap perusahaan. Hubungan emosional dapat terbentuk salah satunya dengan memberikan pelayanan yang baik.

Lebih dalam lagi Gramer dan Brown (Utomo, 2006, p27) memberikan definisi mengenai *loyalitas* (loyalitas jasa), yaitu derajat sejauh mana seorang konsumen menunjukkan perilaku pembelian berulang dari suatu penyedia jasa, memiliki suatu kecenderungan sikap positif terhadap penyedia jasa, dan hanya mempertimbangkan untuk menggunakan penyedia jasa ini pada saat muncul kebutuhan untuk memakai jasa tersebut. Dari definisi yang disampaikan Gramer dan Brown, konsumen yang *loyal* tidak hanya seorang pembeli yang melakukan pembelian berulang, tetapi juga mempertahankan sikap positif terhadap penyedia jasa. *Frequent Flyer Program* (FFP) adalah *loyalty program* yang ditujukan bagi para pengguna jasa penerbangan. *Airline* pengelola FFP memberikan *reward point* pada konsumen (disebut *miles*) yang bisa digunakan untuk membeli tiket (Emch, 2007, p646). Ada 3 alasan bagaimana FFPs dapat mengurangi ongkos produksi dalam dunia penerbangan (Adrian Emch , 2007, p663):

1. Mempertahankan konsumen yang sudah jadi pelanggan lebih murah dibandingkan dengan mencari pelanggan baru.
2. Pelanggan yang setia lebih menguntungkan dari pada pelangggan baru
3. FFP dapat digunakan untuk meningkatkan pelayanan melalui personalisasi dari setiap *service* yang diberikan untuk penumpang.

Pada prinsipnya setiap tiket gratis yang dibeli dengan menggunakan *mileage* yang dimiliki oleh anggota FFP adalah tiket untuk tempat duduk yang kosong sehingga tidak ada biaya yang harus dikeluarkan untuk tiket yang dikeluarkan. Di GFF sendiri, setiap tiket yang dibeli oleh penumpang ada persentase bagian *award*-nya.

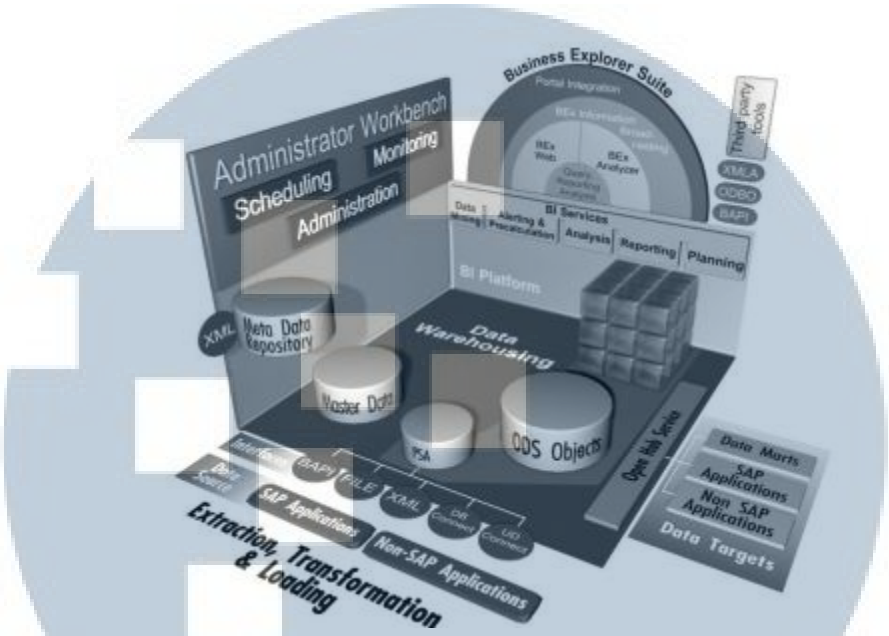
E. *Tools*

1. *SAP NetWeaver Business intelligence*

SAP NetWeaver menyediakan *open integration and application platform* sehingga memungkinkan untuk integrasi dengan *Enterprise Service Architecture*. *SAP NetWeaver* dapat menyatukan proses bisnis pada berbagai teknologi, mengintegrasikan aplikasi yang dibutuhkan karyawan serta mengakses dan mengedit informasi dengan mudah secara terstrukrur. *SAP NetWeaver* merupakan basis untuk semua solusi SAP pada *hardware* tertentu.

2. *SAP Business Information Warehouse* (SAP BW)

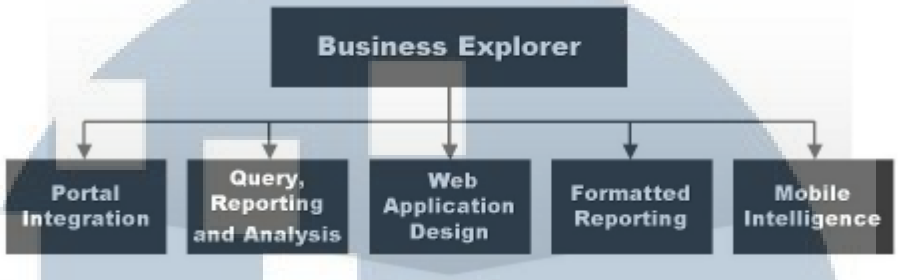
Sebagai komponen inti dari *SAP NetWeaver, SAP Business Information Warehouse* menyediakan fungsionalitas *data warehousing*, BI *Platform*, dan BI *Tools* yang membantu perusahaan mencapai tujuannya. Informasi bisnis yang relevan dari aplikasi SAP dan sumber data eksternal dapat diintegrasikan dan ditransformasikan dalam SAP BW dengan *toolset* yang disediakan. SAP BW menyediakan pelaporan yang fleksibel dan alat analisis untuk mendukung dalam proses evaluasi dan penafsiran data, serta menfasilitasi pendistribusian data tersebut. Perusahaan juga dapat membuat keputusan dengan baik dan menentukan aktivitas yang berorientasikan target berdasarkan hasil analisis tersebut.



Gambar 2. 7 Arsitektur SAP BW (SAP AG, 2006)

3. *SAP BW Business Explorer* (BEx)

Business Explorer (BEx) adalah komponen SAP BW yang menyediakan *flexible reporting* dan *tools* analisis yang dapat digunakan untuk analisis strategis dan pendukung proses pengambilan keputusan dalam organisasi. *Tools* ini meliputi *Query*, *Reporting*, dan fungsi OLAP. BEx ini memungkinkan berbagai pengguna untuk mengakses informasi SAP BW, menggunakan *Enterprise Portal, Intranet* / *Internet* (*Web Application Design*) atau menggunakan perangkat *mobile* (WAP *mobile phone* dan PDA).



Gambar 2. 8 Fungsi SAP Business Explorer (SAP AG, 2006)

4. *BEx Query Designer*

BEx Query Designer merupakan *tools* untuk mendesain *query*. *Query* tersebut kemudian digunakan untuk menganalisis *info provider* pada SAP BW.

5. *SAP Business Objects Xcelcius*

SAP Business Objects Xcelcius merupakan aplikasi *desktop* untuk merancang *dashboard* yang dapat digunakan sebagai aplikasi *reporting* dengan menampilkan data dari SAP BW secara visual dalam bentuk tabel, grafik, dll.