BAB II

LANDASAN TEORI

Sebelum membahas lebih jauh mengenai metode penelitian dan analisis yang dilakukan di penelitian ini, beberapa terminologi perlu dipahami untuk mempermudah pemahaman penelitian ini seperti data, *metadata*, *database*, dan *data warehouse*.

2.1 Data, *Metadata*, dan *Database*

Data menurut O'brien (2001) adalah fakta atau observasi mentah, umumnya mengenai fenomena fisik atau transaksi bisnis. Secara lebih spesifik, data merupakan objek pengukuran atribut-atribut dari suatu entitas. Ladjamudin (2005) menambahkan bahwa data juga merupakan deskripsi dari suatu kejadian yang kita hadapi. Data dapat berupa catatan-catatan dalam kertas, buku, atau tersimpan sebagai *file* dalam *database*. Data yang akan dipergunakan dalam penelitian ini berupa fakta-fakta mengenai transaksi bisnis yang merupakan objek pengukuran atribut dari suatu entitas yang tersimpan sebagai *file* dalam *database*.

Curry et al. (2010) menambahkan bahwa data yang berkualitas memiliki karakteristik sebagai berikut:

1) *Discoverablity & accescibility*

Pengguna harus dapat menemukan dan mengakses data semudah mungkin. Sangatlah penting untuk memfasilitasi pengguna dalam pencarian informasi.

2) *Completeness*

Apakah seluruh informasi yang dibutuhkan ada? Pada beberapa kasus, kekurangan data bisa tidak berpengaruh. Akan tetapi, ketika informasi tersebut sangat penting bagi proses bisnis, kelengkapan data menjadi masalah. Kelengkapan data juga digunakan untuk menghubungkan sebuah data dengan data lainnya.

3) *Interpretation*

Masing-masing orang bisa memiliki asumsi berbeda yang mempengaruhi pemahamaan mereka atas sebuah data. Batasan-batasan yang ada mengenai data tersebut sebaiknya dijelaskan secara eksplisit untuk meminimalisasi kesalahpahaman.

4) *Accuracy*

Data haruslah merepresentasikan nilai sebenarnya dari infomasi yang diwakilinya.

5) *Consistency*

Dalam organisasi, data bisa berasal dari berbagai sistem dan aplikasi. Untuk itu, data harus menggunakan definisi, terminologi, dan identifikasi yang sama atau distandarisasi.

6) *Provenance* & *reputation*

Sumber data sangat mempengaruhi kualitas data. Dengan mengetahui dari mana sebuah data berasal dan bagaimana reputasi dari sumber data tersebut, kualitas data dapat ditentukan.

7) *Timeliness*

Kualitas data juga dapat ditentukan dari seberapa up-to-date-nya data tersebut, dengan memperhatikan juga untuk apa data tersebut digunakan.

Menurut Inmon (2005), *metadata* adalah data tentang sebuah data yang mendeskripsikan struktur, isi, kunci, *index*, dan informasi lain mengenai data tersebut. Connolly dan Begg (2005) menambahkan bahwa kegunaan *metadata* dalam *data warehouse* antara lain:

1) dalam proses *exract* dan *load*, *metadata* digunakan untuk memetakan sumber data ke *common view* dari data yang berbeda di *data warehouse*;
2) dalam proses manajemen *warehouse*, *metadata* digunakan untuk mengotomasi pembuatan tabel *summary*;
3) sebagai bagian dari proses manajemen *query*, *metadata* digunakan untuk mengarahkan *query* ke sumber data yang paling sesuai.

Date (2000) mengarisbawahi bahwa *database* merupakan *record* yang terkomputerisasi yang bertujuan untuk menyediakan informasi ketika dibutuhkan. Connolly dan Begg menambahkannya dengan mengatakan bahwa *database* merupakan kumpulan data yang saling berhubungan secara logis dan didesain untuk memenuhi kebutuhan informasi suatu organisasi.

2.2 *Data warehouse*

McLeod (2001) menjelaskan bahwa *data warehouse* merupakan perkembangan dari konsep *database* yang menyediakan sumber daya data yang lebih baik bagi *user*, memungkinkan *user* untuk memanipulasi dan menggunakan data tersebut secara intuitif. Inmon mengembangkannya dengan menjelaskan bahwa *data warehouse* adalah kumpulan data yang mempunyai karakteristik berorientasi subjek, terintegrasi, *time-variant*, dan *non-volatile* untuk mendukung proses pengambilan keputusan.

2.2.1 *Karakteristik Data warehouse*

Menurut Inmon karakteristik data dalam *data warehouse* adalah sebagai berikut:

1) *Subject oriented* (berorientasi subjek)

   Data warehouse menyimpan data yang disesuaikan dengan fungsi bisnis dari perusahaan untuk mendukung pengambilan keputusan. Data berisi data-data *summary* yang akan digunakan dalam proses analisis, bukan tentang rincian atau detail data.

2) *Integrated* (terintegrasi)

   Data harus terintegrasi karena sumber data mungkin berasal dari sistem aplikasi yang berbeda di sebuah perusahaan. Sumber data tersebut sering kali menggunakan format yang berbeda satu sama lain. Tujuan dari integrasi data tersebut agar data tidak terbagi-bagi. Syarat integrasi sumber data dapat dipenuhi dengan berbagai cara seperti konsisten dalam penamaan variabel, ukuran variabel, struktur pengkodean, dan atribut fisik dari data.

3) *Time-variant* (rentang waktu)

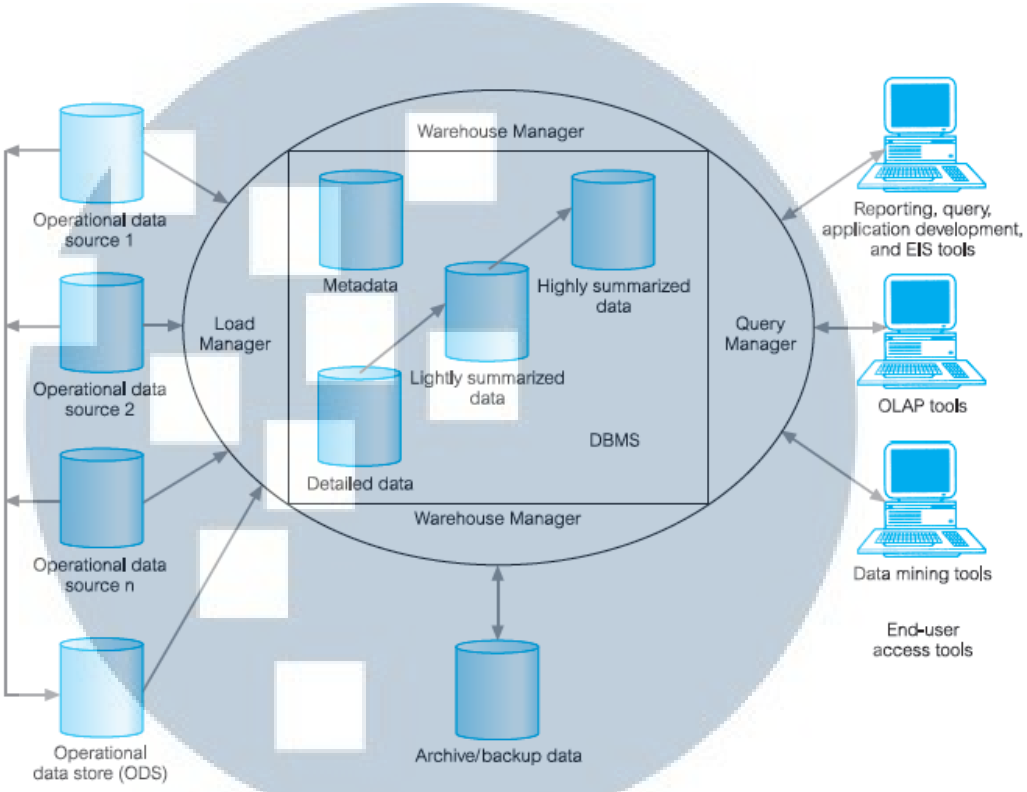
Data dalam *data warehouse* hanya akurat dan valid pada satu waktu atau dalam interval waktu tertentu, misalnya *data warehouse* menyajikan informasi dari lima tahun terakhir. Karakteristik *time-variant* ini juga bisa dilihat secara eksplisit, dari adanya data yang menandakan waktu di *data warehouse*. Selain itu, *data warehouse* juga menyimpan karakteristik waktu secara implisit dari kunci-kunci yang menandakan urutan penyimpanan data.

4) Non-volatile

Hal ini berarti data dalam *data warehouse* cenderung tidak berubah. Penambahan data baru tidak mengantikan data yang ada melainkan menambahkan data yang sudah ada. Jadi, data yang lama dan yang baru sama-sama tersimpan di *data warehouse*.

2.2.2 *Arsitektur Data warehouse*

Connolly dan Begg menjelaskan garis besar arsitektur dan komponen-komponen penting dari sebuah *data warehouse* terlihat seperti pada gambar 2.1 di bawah ini.



Gambar 2.1 Arsitektur *Data warehouse* (Connolly, 2005)

Arsitektur *data warehouse* tersebut memiliki komponen-komponen sebagai berikut:

1) *Operational Data*

Operational data digunakan sebagi sumber data dalam proses perancangan *data warehouse*.

2) *Operational Data Store (ODS)*

ODS merupakan tempat penyimpanan data operasional terkini dan terintegrasi yang digunakan untuk analisis. Umumnya, ODS dibuat dan diisi dengan data

yang sama seperti *data warehouse*, atau hanya berfungsi sebagai penyimpanan sementara sebelum data dipindahkan ke *data warehouse*.

ODS umumnya dibuat ketika sistem operasional tidak mampu menyediakan kebutuhan *reporting*. Pembuatan ODS bisa membantu pembuatan *data warehouse* karena ODS bisa menyediakan data yang sudah di-*extract* dari sumber dan bersih.

3) *Load Manager*

Load manager menjalankan semua operasi yang berkaitan dengan proses ekstraksi dan *loading* data ke dalam *data warehouse*.

4) *Warehouse Manager*

Warehouse manager menjalankan semua operasi yang berkaitan dengan manajemen data di dalam *data warehouse*. Operasi yang dikerjakan oleh *warehouse manager* mencakup:

a) analisis data untuk menjamin konsistensi;
b) transformasi dan penggabungan sumber data dari penyimpanan sementara ke dalam tabel *data warehouse* ;
c) pembuatan *index* dan *view* ;
d) pembuatan denormalisasi (jika dibutuhkan);
e) pembuatan agregasi (jika dibutuhkan);
f) back-up dan archiving data.

5) *Query Manager*

Query manager menjalankan semua operasi yang berkaitan dengan manajemen *query* yang dilakukan oleh *user*.

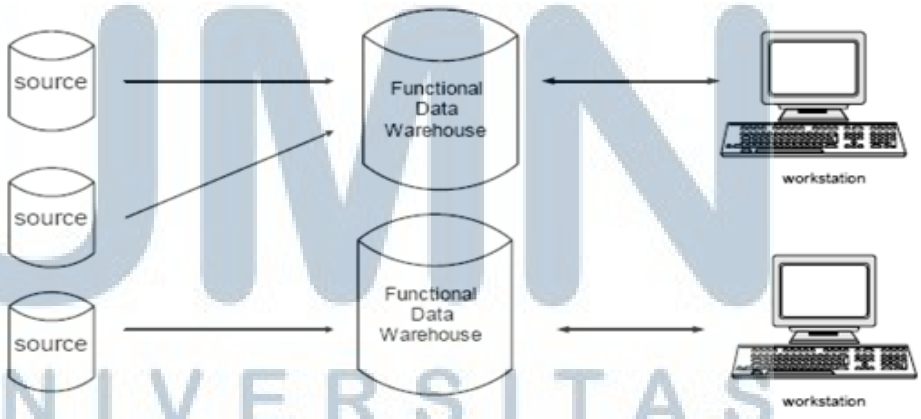
6) *End-User Access Tools*

Secara umum, tools ini terdiri dari reporting and query tools, application development tools, executive information system tools, online analytical processing tools, serta data mining tools.

2.2.3 Bentuk *Data warehouse*

1) *Data warehouse* fungsional

Data warehouse fungsional dibuat berdasarkan fungsi bisnis yang ada dalam perusahaan. Keuntungan dari bentuk ini adalah sistem akan mudah dibangun dengan biaya yang relatif rendah. Kerugiannya, ada resiko kehilangan konsistensi data dan terbatasnya kemampuan pengguna dalam hal pengumpulan data.

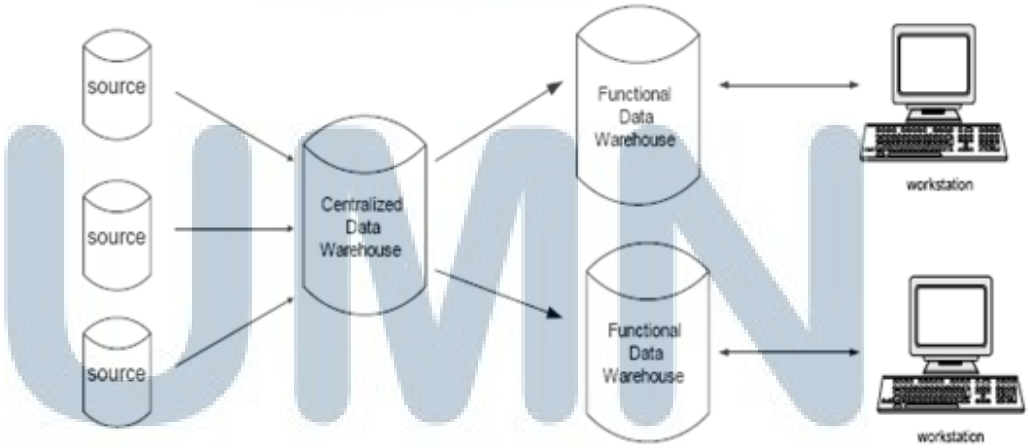


Gambar 2.2 *Data warehouse* Fungsional

2) *Data warehouse* terpusat

Data warehouse terpusat merupakan *data warehouse* fisikal tunggal yang memuat semua data untuk area fungsional yang khusus. *Data warehouse* ini digunakan ketika terdapat kebutuhan data yang informatif dan sudah terdapat banyak *end-user* yang terhubung ke komputer pusat atau jaringan. Pada *data warehouse* terpusat, data dari sumber data lebih dahulu dikumpulkan pada suatu tempat terpusat, baru kemudian data tersebut digunakan berdasarkan fungsi-fungsi bisnis yang ada.

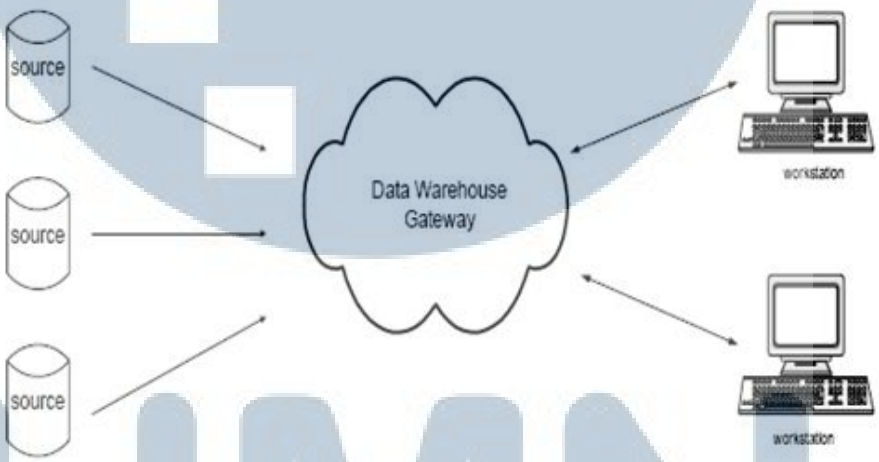
Keuntungan dari arsitektur ini adalah datanya terpadu sehingga memberikan konsistensi yang tinggi. Kerugiannya, pembangunannya membutuhkan waktu yang cukup lama.



Gambar 2.3 *Data warehouse* Terpusat

3) *Data warehouse* terdistribusi

Pada *data warehouse* terdistribusi, digunakan *gateway* yang berfungsi sebagai jembatan penghubung antara *data warehouse* dengan *workstation* yang menggunakan sistem beraneka ragam. Sistem terdistribusi ini memungkinkan perusahaan mengakses sumber data yang berada di luar lokasi perusahaan. Bentuk ini memberikan data yang konsisten karena sebelum digunakan, data terlebih dahulu disesuaikan. Kerugiannya, sistem menjadi lebih kompleks karena sistem operasi dikelola secara terpisah.



Gambar 2.4 *Data warehouse* Terdistribusi

2.2.4 *Keuntungan Data warehouse*

Menurut Connolly dan Begg, implementasi dari *data warehouse* akan memberikan keuntungan sebagai berikut:

1) Pengembalian investasi yang tinggi

Sebuah perusaahan perlu mengeluarkan sumber daya yang besar untuk memastikan implementasi *data warehouse* berhasil. Jumlah yang perlu dikeluarkan tersebut bervariasi sesuai dengan solusi yang ingin didapat. Penelitian yang dilakukan oleh *International Data Corporation* (IDC) pada tahun 1996 (*Business Wire*, 9 April 1996) memperlihatkan bahwa rata-rata *return of investment* (ROI) dalam *data warehousing* mencapai 401%, dengan lebih dari 90% perusahaan yang disurvey mencapai lebih dari 40% ROI, setengahnya mencapai lebih dari 160% ROI, dan seperempatnya lebih dari 600% ROI.

2) Keunggulan Kompetitif.

ROI yang besar tersebut membuktikan bahwa perusahaan juga mendapatkan keunggulan kompetitif yang besar dengan penerapan *data warehouse* ini. Keunggulan kompetitif didapatkan dengan memungkinkan pengambil keputusan mengakses dta yang sebelumnya tidak tersedia dan tidak diketahui, seperti pelanggan, tren, dan permintaan.

3) Peningkatan produktifitas dari pengambil keputusan

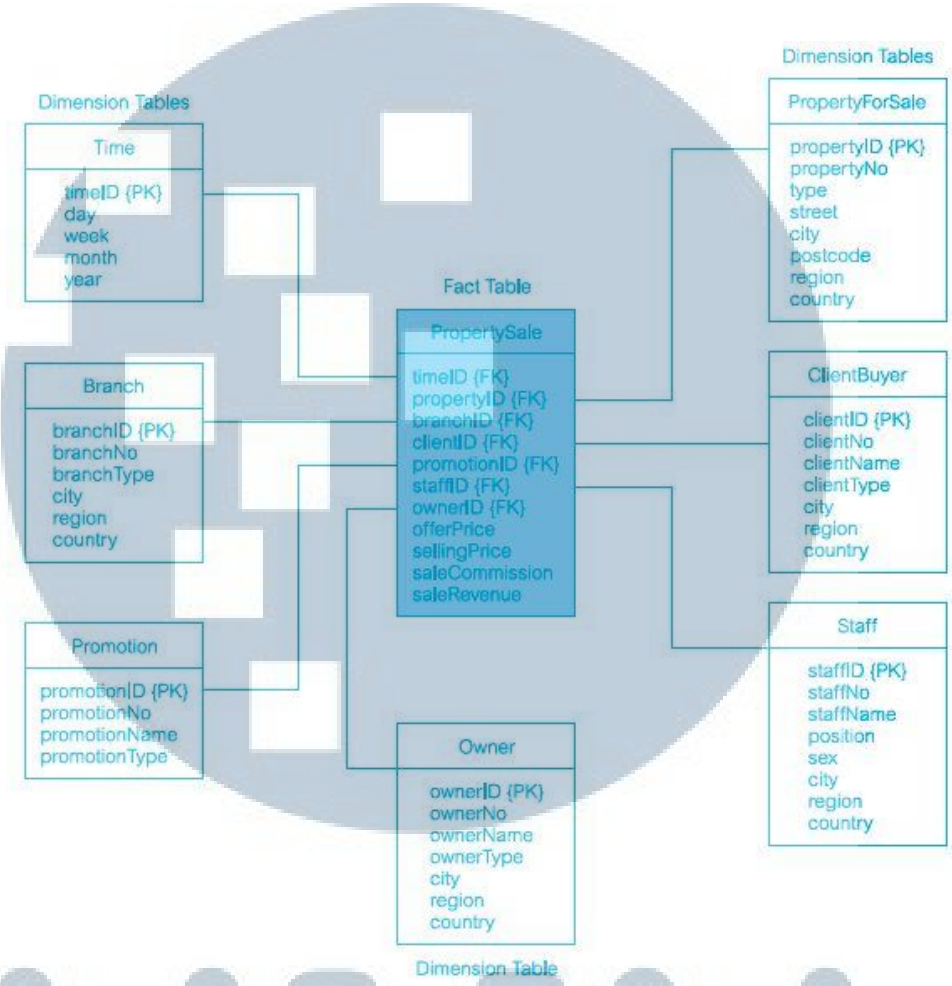
Data warehousing meningkatkan produktifitas pengambil keputusan dengan menyediakan *database* yang terintegrasi, konsisten, berorientasi subjek, dan historikal. Data tersebut dibentuk menjadi informasi yang bernilai sehingga pengambil keputusan dapat melakukan analisis yang lebih substansif, akurat, dan konsisten.

2.2.5 *Pemodelan Data warehouse*

Connolly dan Begg menjelaskan bahwa komponen *database* dari *data warehouse* dapat dideskripsikan dengan menggunakan teknik yang disebut *dimensionality modeling*. *Dimensionality modeling* menggunakan konsep *entity-relationship modeling* dengan beberapa batasan yang penting. Setiap *dimensional model* tersusun atas sebuah tabel yang memiliki *composite primary key*, disebut tabel fakta, dan kumpulan tabel-tabel kecil yang disebut tabel dimensi.

1) *Star Schema*

Setiap tabel dimensi memiliki sebuah *primary key* (*non-composite*) yang berhubungan dengan tepat satu komponen dari *primary key* yang ada di tabel fakta. Dengan kata lain, *primary key* di tabel fakta tersusun atas dua atau lebih *foreign key*. Karakteristik yang menyerupai bintang tersebut disebut skema *star*. Dengan kata lain, skema *star* merupakan model data dimensional yang memiliki sebuah tabel fakta di tengah, dikelilingi oleh tabel-tabel dimensi yang masih bisa disederhanakan lagi (belum dinormalisasi).



Gambar 2.5 Contoh *Star Schema* (Connolly, 2005)

Keunggulan dari *star schema* menurut Ponniah (2001) adalah :

a) Mudah dimengerti

Star schema menggambarkan dengan jelas bagaimana *user* berpikir dan memerlukan data untuk *query* dan analisa. *Star schema* menggambarkan hubungan antar tabel sama seperti *user* melihat hubungan tersebut secara normal;

b) Mengoptimalkan navigasi

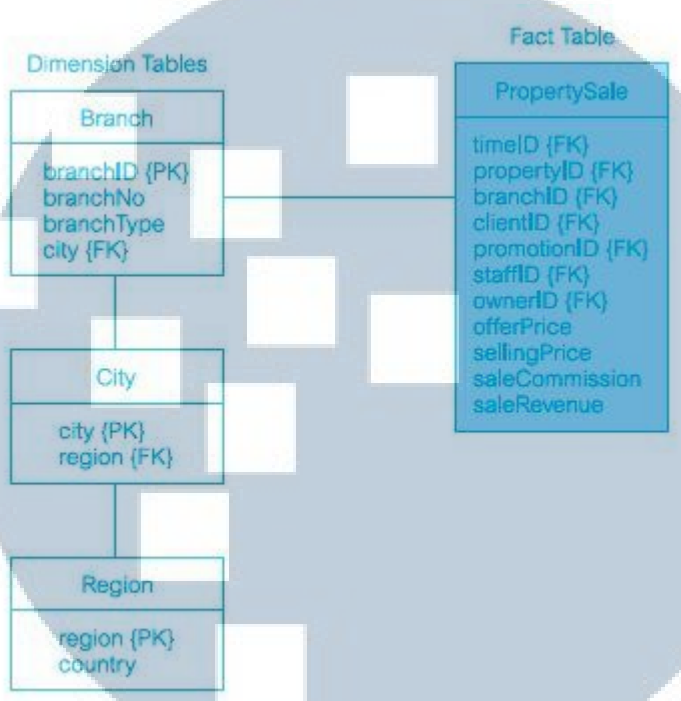
Hubungan antar tabel yang sederhana memungkinkan *user* untuk mengoptimalkan navigasi. Walaupun hasil *query* terlihat kompleks, tapi *user* masih dapat melakukan navigasi dengan mudah;

c) Cocok dengan pemrosesan *query*

Star schema terpusat pada *query* sehingga setiap proses *query* dapat lebih mudah dijalankan. Tanpa bergantung pada banyak dimensi dan kerumitan *query*, yang pertama kali dilakukan adalah memilih baris dari tabel dimensi dan kemudian menemukan baris yang sama di tabel fakta.

2) *Snowflake Schema*

Salah satu variasi dari skema *star* di atas adalah skema *snowflake*. Dalam skema *snowflake*, tabel dimensi diizinkan untuk memiliki dimensi juga. Dengan kata lain, skema *snowflake* merupakan model data dimensional yang memiliki sebuah tabel fakta di tengah, dikelilingi oleh tabel-tabel dimensi yang juga memiliki dimensi (sudah dinormalisasi).



Gambar 2.6 Contoh *Snowflake Schema* (Connolly. 2005)

Menurut Ponniah (2001), keuntungan dan kerugian dari *snowflake schema* adalah:

a) memerlukan tempat penyimpanan yang kecil;

b) struktur ternormalisasi sehingga mudah di-*update* dan di-*maintain;*

c) skema kurang intuitif dan terlalu kompleks untuk *end-user;*

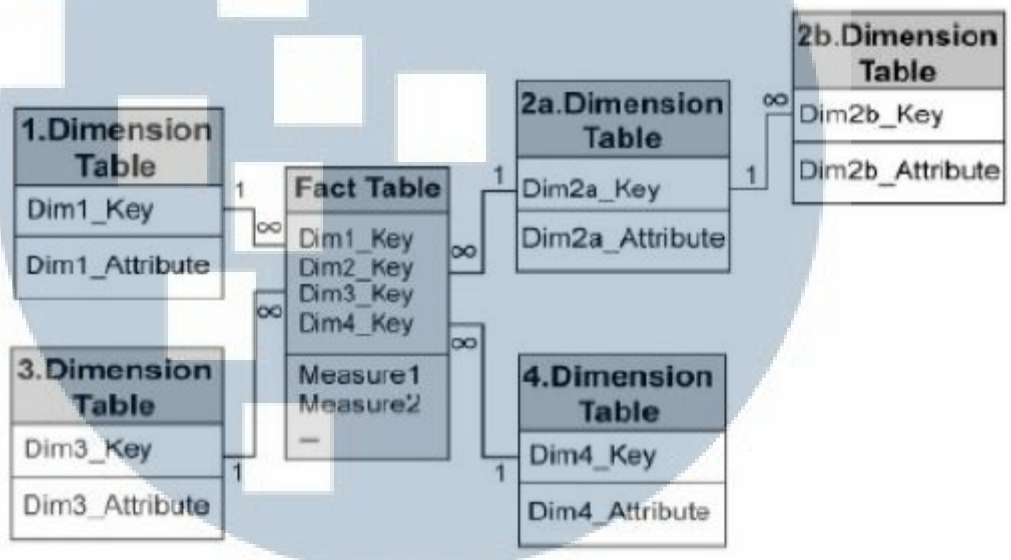
d) sulit untuk melihat isi skema;

e) performa *query* menurun karena adanya hubungan tambahan.

3) Starflake Schema

Variasi lain dari kedua model *database* di atas menggabungkan karakteristik dari keduanya. Kombinasi tersebut disebut skema *starflake*. Dengan kata lain,

skema *starflake* merupakan model data dimensional yang memiliki sebuah *tabel* fakta di tengah, dikelilingi baik oleh tabel-tabel yang sudah dinormalisasi maupun belum dinormalisasi.



Gambar 2.7 Contoh Starflake Schema

2.3 ETL (*Extract, Transform, Load*)

Kimball (2002) menjelaskan bahwa ETL adalah kumpulan proses dimana sumber data operasional disiapkan untuk dimasukkan ke dalam *data warehouse*. ETL merupakan proses utama dari proses *staging area* sebelum data ditampilkan atau di-*query*. Proses ini terdiri dari proses ekstraksi data dari sumber data, mengubah bentuknya, lalu mengeluarkannya, serta membuat *index* dan menerbitkannya.

Dengan kata lain, ETL berfungsi untuk mengambil data dari sumber data ke tempat penyimpanan sementara, kemudian melakukan transformasi data tersebut sesuai dengan kebutuhan, dan akhinya mengirimkan data yang sudah

ditransformasikan tersebut ke tempat penampungan akhir sebelum akhirnya ditampilkan ataupun diolah kembali.

2.4 *Data Mart*

Kimball memaparkan bahwa *data mart* adalah himpunan bagian logikal dan fisikal dari area presentasi *data warehouse* yang bertujuan untuk menjawab masalah-masalah khusus yang terdapat dalam sebuah bisnis. Inmon mendukungnya dengan menjelaskan bahwa *data mart* merupakan struktur data yang berpusat pada departemen-departemen tertentu. Data tersebut diambil dari *data warehouse* dimana data sudah didenormalisasikan untuk memenuhi kebutuhan informasi dari suatu departemen. Connolly dan Begg juga menjelaskan hal yang sama dengan mengatakan bahwa *data mart* merupakan himpunan bagian dari *data warehouse* untuk mendukung kebutuhan dari fungsi bisnis tertentu. *Data mart* bisa berdiri sendiri atau terhubung ke *data warehouse* pusat perusahaan.

2.5 Metodologi perancangan *Data Warehouse*

Dalam merancang *data warehouse*, ada dua pendekatan yang umum digunakan, *top-down* dan *bottom-up*. Salah satu pencetus pedekatan *top-down* adalah William H. Inmon, sedangkan salah satu pencetus pendekatan *bottom-up* adalah Ralph Kimball.

Menurut Inmon, *data warehouse* dirancang sebagai tempat penyimpanan terpusat untuk seluruh organisasi. Masing-masing *data mart* dibangun dengan menggunakan data yang ada di dalam *data warehouse* tersebut. Dengan kata lain, rancangan ini

mendahulukan pembuatan *data warehouse* terlebih dahulu, disusul dengan *data mart*s.

Kelebihan dari pendekatan *top-down* ini adalah:

1) *Data warehouse* langsung mencakup seluruh organisasi.
2) Kerangka *data warehouse* lebih terstruktur.
3) Penyimpanan data menjadi terpusat.
4) Kontrol informasi dapat dilakukan secara tersentralisasi

Akan tetapi, pendekatan ini memiliki kelemahan yaitu:

1) Waktu implementasi lebih lama.
2) Resiko kegagalan relatif tinggi karena rancangan yang rumit.
3) Membutuhkan biaya yang relatif besar

Kimball mengemukakan bahwa *data mart* merupakan *data warehouse* yang berorientasi pada subjek atau departemen. Oleh sebab itu, pendekatan ini digunakan untuk perancangan di tingkat departemen atau fungsi bisnis tertentu, baru kemudian diintegrasikan menjadi *data warehouse* organisasi secara keseluruhan.

Kelebihan dari pendekatan *bottom-up* ini adalah:

1) Implementasi lebih mudah dikelola dan lebih cepat memberikan hasil.
2) Resiko kegagalan relatif kecil.
3) *Data mart* yang lebih penting dapat dijadwalkan lebih awal.

Kelemahan dari pendekatan bottom-up adalah:

1) Memungkinkan terjadinya duplikasi data di tiap data mart yang berbeda.
2) Data bisa menjadi tidak konsisten dan sulit direkonsiliasi.

3) Banyak antarmuka yang perlu dikelola.

2.6 *Online Analytical Processing* (OLAP)

Connoly dan Begg menyebutkan bahwa OLAP merupakan sebuah terminologi yang menjelaskan teknologi yang digunakan pada tampilan multi-dimensi dari kumpulan data untuk menyediakan akses informasi yang cepat demi mendukung proses analisis. OLAP juga berarti perpaduan dinamis, analisis dan konsolidasi dari data multidimensi berukuran besar. OLAP memungkinkan *user* untuk mendapatkan pengertian dan pengetahuan yang mendalam mengenai berbagai aspek dari data dengan akses yang cepat, konsisten, serta interaktif melalui variasi tampilan data.

OLAP berbeda dengan *online transactional processing* (OLTP) seperti disimpulkan oleh Vercellis (2009) dalam tabel:

Tabel 2.1 Perbedaan OLTP dan OLAP

Karateristik	**OLTP**	**OLAP**
Volatilitas	Data dinamis	Data statis
Waktu	Data saat ini	Data saat ini dan historis
Dimensi waktu	Implisit dan terkini	Eksplisit dan varian
Granularity	Data yang detil	Data agregasi dan konsolidasi
Update	Berlanjut dan tidak regular	Periodik dan regulat
Aktivitas	Berulang kali	Tidak dapat diprediksi
Fleksibilitas	Rendah	Tinggi
Kinerja	Tinggi	Rendah untuk *query* yang kompleks
User	Bagian operasional	*Knowledge workers*

Fungi	Operasional	Analisis
Tujuan penggunaan	Transaksi	*Query* kompleks dan pendukung keputusan
Prioritas	Kinerja tinggi	Fleksibilitas tinggi
Metrik	Rata-rata transaksi	Respon efektif
Ukuran data	MB hingga GB	GB hingga TB

Menurut Connolly dan Begg, ada beberapa tipe OLAP , yaitu:

1) Multidimensional OLAP (MOLAP)

MOLAP menggunakan struktur data tertentu dan *database* multidimensi untuk mengorganisir, menjelajah, dan menganalisa data.

2) *Relational* OLAP (ROLAP)

ROLAP menggunakan lapisan *metadata* untuk mendukung manajemen data sehingga menghindari kebutuhan untuk membuat struktur data multidimensi yang statis.

3) *Hybrid* OLAP (HOLAP)

HOLAP menyediakan kemampuan analisis yang terbatas baik dengan *relational database management system* (RDBMS) ataupun *server* MOLAP. HOLAP *tools* mengirimkan data tertentu langsung dari DBMS atau melalui server MOLAP dalam bentuk data *cube* dimana datanya disimpan, dianalisa, dan dirawat secara lokal.

4) *Desktop* OLAP (DOLAP)

DOLAP *tools* menyimpan data OLAP dalam *client-based file* dan mendukung proses multidimensi menggunakan mesin multidimensi *client*.

2.7 *Business intelligence*(BI)

Williams (2007) menyatakan, BI mengombinasikan produk, teknologi, dan metode untuk mengorganisir informasi kunci yang dibutuhkan manajemen untuk meningkatkan keuntungan dan performa. Secara umum, BI merupakan informasi bisnis dan analisa bisnis dari proses bisnis utama yang menentukan pengambilan keputusan untuk meningkatkan performa bisnis. Secara khusus, BI meningkatkan aset informasi dari proses-proses bisnis utama sehingga performa bisnis meningkat.

Stackowiak (2007) menjelaskan BI sebagai akses yang tepat ke data atau informasi yang dibutuhkan untuk membuat keputusan bisnis yang tepat pada saat yang tepat. Data tersebut dapat berupa data mentah ataupun data yang telah dianalisis sebelumnya. Akses informasi tersebut memungkinkan pengelolaan bisnis dilakukan berdasarkan fakta, bukan mengandalkan intuisi.

2.8 *Dashboard*

Rasmussen (2009) menjelaskan bahwa sebuah *dashboard* yang dirancang dengan baik dapat menyediakan informasi kunci yang diperlukan penggunanya untuk memonitor bagian yang menjadi tanggung jawabnya, kemudian dapat menemukan masalah dan mengambil tindakan untuk meningkatkan performa organisasinya.

Data warehouse dan OLAP merupakan dua teknologi fundamental yang mendukung kesuksesan jangka panjang dari *dashboard*. Kedua teknologi tersebut memungkinkan *dashboard* untuk:

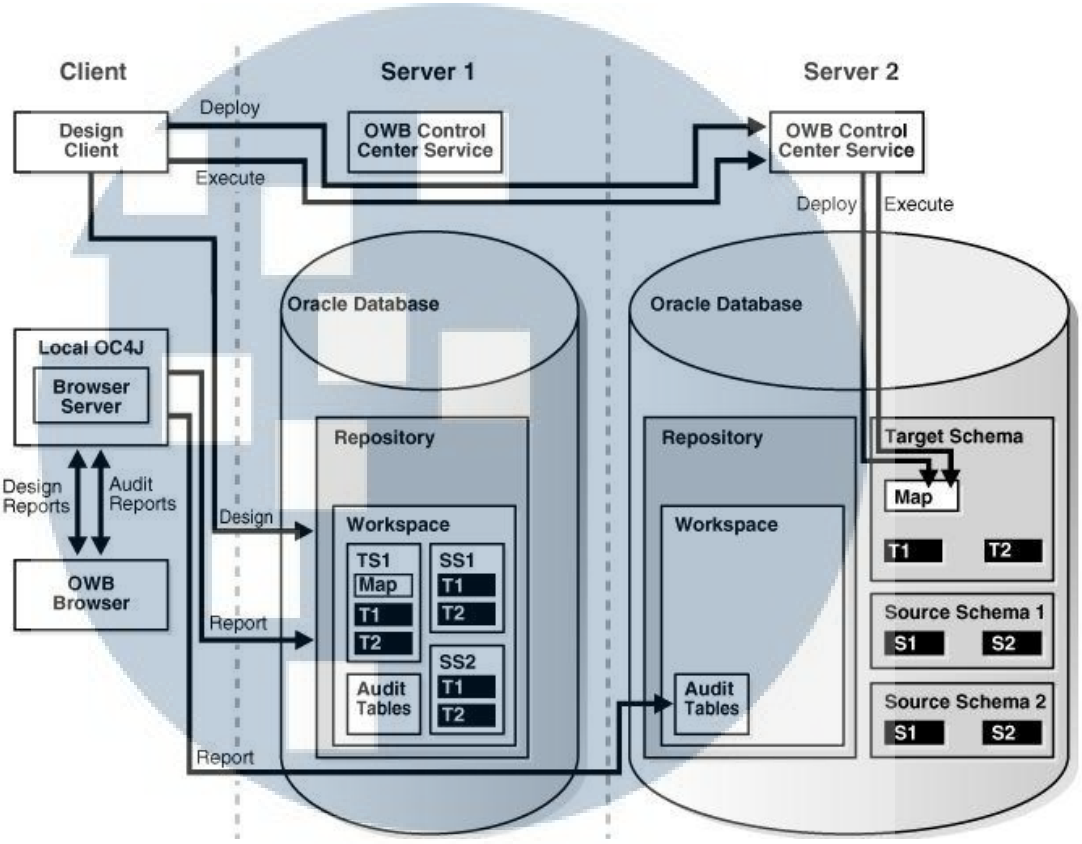
1) menampilkan data yang berasal dari berbagai sumber;

2) menampilkan hasil pengukuran dari perhitungan yang sederhana ataupun kompleks;
3) menyediakan informasi baru ke layar dengan proses yang sebentar;
4) menawarkan *drill down* dari *summary data* ke detail transaksi.

2.9 Oracle *Warehouse Builder*

Oracle Warehouse Builder (OWB) merupakan sebuah *Business intelligence Tool* yang menyediakan solusi terintegrasi untuk mendesain dan mengimplementasikan *data warehouse, datamart,* dan aplikasi *business intelligence*. OWB mendukung siklus manajemen informasi seutuhnya mulai dari *design, build, extract, transform,* dan *load, integrate,* serta *maintain.*

Komponen arsitektur OWB terbagi menjadi dua bagian, *server* dan *client*. Dari sisi *server,* komponen tersebut terdiri dari OWB *repository*, *workspaces*, *control center service*, *control center agent* (J2EE *Runtime*), dan *target schemas*. Dari sisi *client*, terdiri dari *control center manager*, *design center*, dan *repository browser*. Arsitektur OWB ini diilustrasikan seperti pada gambar 2.7.



Gambar 2.8 Arsitektur Oracle Warehouse Builder

Fitur utama dari arsitektur OWB adalah **OWB *repository*** yang memiliki sebuah *database instance* terpadu dan dilengkapi dengan sebuah *schema* dan *database objects*. *Repository* schema, bernama Oracle Warehouse BuilderSYS, dibuat pada saat Oracle *Database* di-*install*. Agar dapat digunakan, akun Oracle Warehouse BuilderSYS dan Oracle Warehouse BuilderSYS_AUDIT perlu di-*unlock*.

Untuk mulai menggunakan OWB, paling sedikit satu ***workspace*** harus dibuat. Pada penggunaannya, *user* mengakses *workspace*-nya masing-masing, bukan *repository* secara keseluruhan. Masing-masing *workspace* ini disimpan dalam sebuah *repository schema*.

Setiap *workspace* memiliki sebuah *default control center* yang dimulai dan diakhiri oleh ***control center service*** yang bersesuaian. *Control center* menyimpan detail informasi mengenai setiap *deployment* dan eksekusi, yang dapat diakses per *object* ataupun per job. Hanya satu *control center* yang aktif setiap waktu tertentu. Control center tersebut adalah yang terkait dengan konfigurasi yang aktif pada saat itu.

Control center agent bekerja pada Oracle Containers for J2EE (OC4J) server. Beberapa kemampuan OWB yang terkait dengan akses ke non-Oracle data, seperti Code Templates dan Web services, mengandalkan kode Java untuk mengeksekusi proses di luar *database*, di OC4J server yang disebut Java atau J2EE Runtime.

Data di dalam OWB project disimpan dalam ***target schema*** di *server*. Data tersebut disimpan dalam bentuk data objects seperti tabel, *views*, dimensi, dan *cube*. Untuk merancang *target schema*, pertama-tama perlu dibuat *target module* yang mengandung semua *data objects*. *Target module* adalah *container* yang menyimpan definisi *metadata* dari semua *data warehouse objects*. Setiap *target module* berkorespondensi dengan satu *target location* yang merepresentasikan lokasi fisik di mana *objects* tersebut disimpan.

Control center manager digunakan untuk melakukan *deploy* terhadap *design objects* dan secara bersamaan mengeksekusi kode yang di-*generate*. *Control center manager* merupakan antarmuka *client* yang berinteraksi dengan *target schema* melalui *control center service*. *Control center manager* memungkinkan untuk melihat dan mengelola seluruh aspek *deployment* dan eksekusi, termasuk status dan hasilnya.

OWB menggunakan hanya satu *control center manager* dan *control center service* yang terkait untuk setiap *database instance*.

Design center merupakan tempat utama untuk merancang, mengelola, dan membangun solusi integrasi data. *Design center* ini menyediakan antarmuka untuk mengakses *tools* lainnya dan melakukan perintah-perintah pada *objects* yang terpilih. *Design center* juga menyediakan *editors* dan *tools* yang spesific untuk bekerja pada masing-masing *objects*.

Repository browser menyediakan akses berbasis web ke *workspaces* untuk melihat detil *desain* dan *metadata* properties, serta membuat laporan mengenai *workspace objects* dan *workspace data*. Dengan *repository browser*, laporan-laporan yang dapat dilihat antara lain *timing* untuk setiap *mapping* dan aliran proses, detil aktivitas untuk setiap aliran proses, detil eror, serta informasi *deployment*.

2.10 Oracle *Business intelligence Enterprise Edition* (OBIEE)

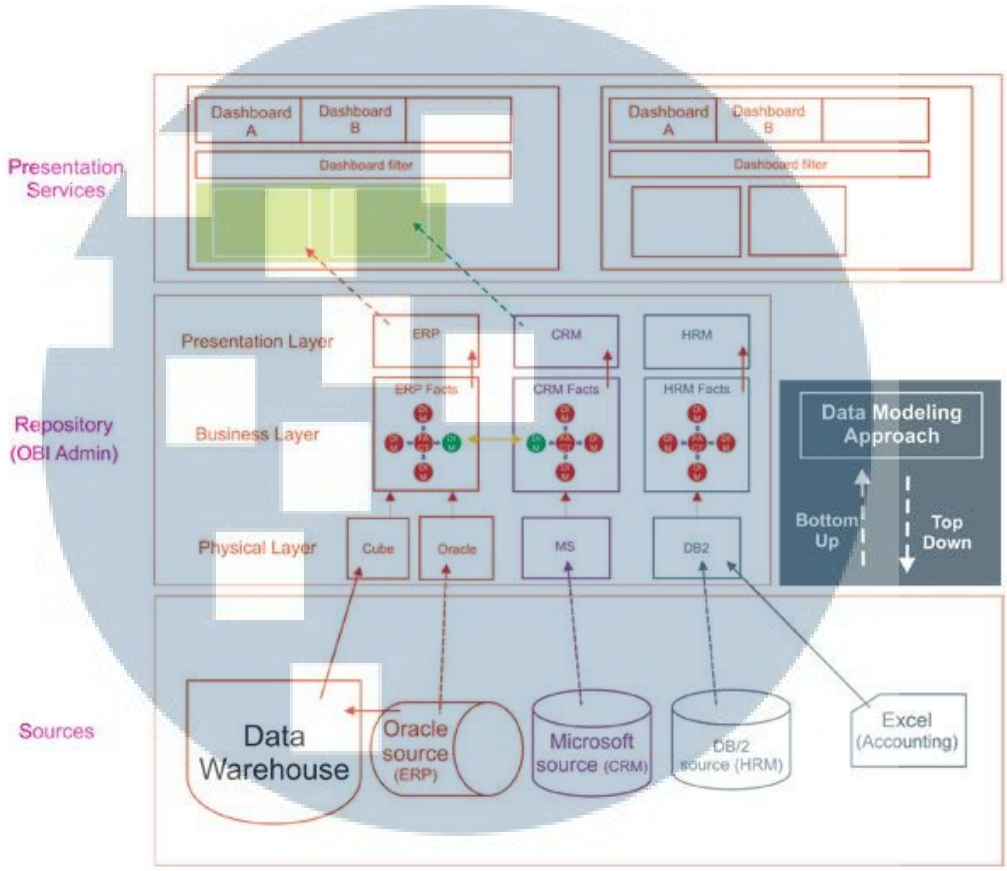
OBIEE merupakan rangkaian produk BI yang komprehensif untuk skala perusahaan. OBIEE menawarkan seluruh kemampuan BI termasuk *interactive dashboards*, *full ad hoc*, proaktif *intelligence* dan *alerts*, penyediaan laporan *enterprise* dan keuangan, prediksi *intelligence* yang *real-time*, analisis yang tidak terkoneksi, dan lain-lain.

OBIEE memiliki komponen-komponen sebagai berikut:

1) Oracle BI *Server*: lapisan model dan abstraksi bisnis perusahaan;
2) Oracle BI *Answers*: *ad-hoc query* dan *reporting;*

3) Oracle BI *Interactive Dashboards*: *dashboard* yang interaktif untuk mengakses konten BI dan aplikasi;
4) Oracle BI *Delivers*: *monitoring* dan *alerting* aktivitas bisnis yang proaktif;
5) Oracle BI *Disconected Analytics*: fungsi analisis yang lengkap untuk *mobile professionals;*
6) Oracle BI *Publisher* (sebelumnya XML *Publisher*): pembuatan dan pendistribusian laporan perusahaan;
7) Oracle BI *Breifing Books*: *snapshots* dari halaman *dashboard* untuk dilihat dan dibagikan secara *offline;*
8) Hyperion *Interactive Reporting*: ad-*hoc reporting* yang intuitif dan interaktif;
9) Hyperion SQR *Production Reporting*: pembuatan *report* dengan jumlah besar dan memiliki format yang cocok untuk presentasi;
10) Hyperion *Financial Reporting*: saranan pembuatan *report* untuk manajemen dan keuangan;
11) Hyperion *Web Analysis*: analisis, presentasi, dan laporan OLAP berbasis web.

Pemodelan data pada OBIEE terdiri dari *physical layer*, *business model layer*, dan *presentation layer*.



Gambar 2.9 *Data Modeling* pada OBIEE

Physical layer berisi tentang sumber data fisik. *Business model layer* menggambarkan hubungan secara logis. Setiap model bisnis mengandung tabel logikal yang dipetakan pada sumber data yang ada di *physical layer*. *Presentation layer* menampilkan informasi yang dimodelkan di *business model layer*.

2.11 Toad for Oracle

Toad for Oracle merupakan aplikasi perangkat lunak dari Quest Software untuk administrasi dan pengembangan berbagai *database* relasional. Toad pada mulanya

dikembangkan oleh pengembang *Oracle* untuk meringankan pekerjaannya dan disebut *Tool for Oracle Application Developers* atau disingkat TOAD.