BAB III

PENGEMBANGAN DATA WAREHOUSE

A. Anatomi Data Warehouse

Penerapan awal dari arsitektur *data warehouse* dibuat berdasarkan konsep bahwa *data warehouse* mengambil data dari berbagai sumber dan memindahkannya ke dalam pusat pengumpulan data yang besar. Konsep ini sebenarnya lebih cenderung kepada sebuah lingkungan *mainframe* yang terpusat.

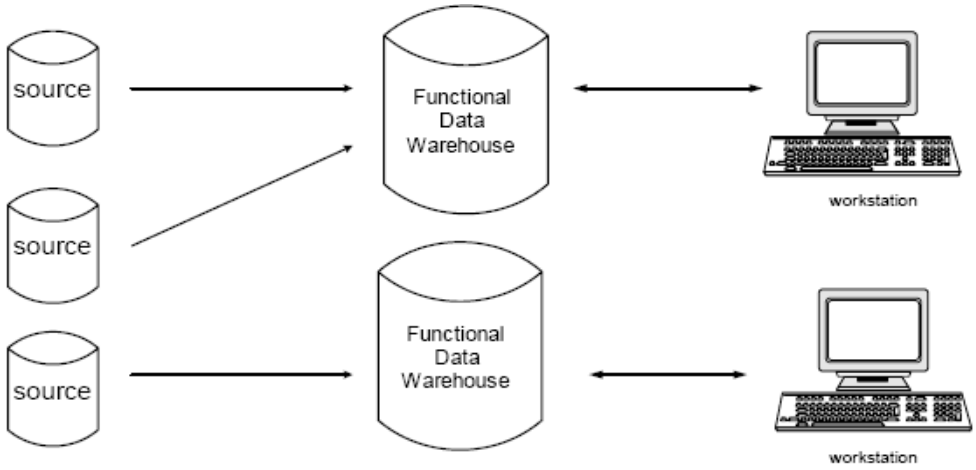
Keunggulan teknologi *Client Server* memungkinkan *data warehouse* diterapkan dalam berbagai macam cara untuk menampung kebutuhan pemakai sistem secara lebih proposional. Dalam suatu kasus, misalkan saja pemakai tertentu perlu menggabungkan data dari sebuah sistem pengumpulan data yang statis dengan data dari sistem operasional yang dinamis hanya dengan sebuah *query* saja.

Berikut ini adalah tiga jenis dasar sistem *Data Warehouse :*

1. Data Warehouse Fungsional

Kata operasional disini merupakan basisdata yang diperoleh dari kegiatan sehari-hari. Data warehouse dibuat lebih dari satu dan dikelompokkan berdasar fungsi-fungsi yang ada di dalam perusahaan seperti fungsi keuangan(*financial),*marketing,personalia dan lain-lain.

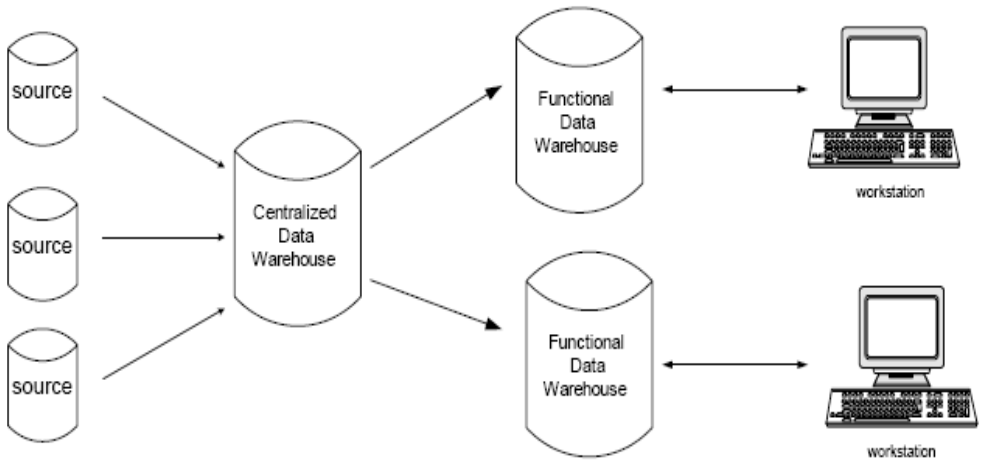
Keuntungan dari bentuk data warehouse seperti ini adalah, sistem mudah dibangun dengan biaya relatif murah sedangkan kerugiannya adalah resiko kehilangan konsistensi data dan terbatasnya kemampuan dalam pengumpulan data bagi pengguna.



Gambar 3.1 Bentuk data warehouse fungsional

2. Data Warehouse Terpusat

Bentuk ini terlihat seperti bentuk data warehouse fungsional, namun terlebih dahulu sumber data dikumpulkan dalam satu tempat terpusat, kemudian data disebar ke dalam fungsinya masing-masing, sesuai kebutuhan persuhaan. Data warehouse terpusat ini, biasa digunakan oleh perusahaan yang belum memiliki jaringan eksternal.



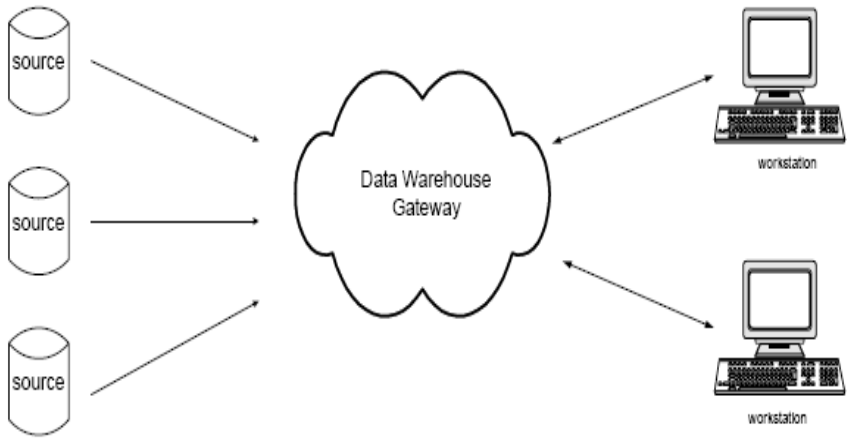
Gambar 3.2 Bentuk data warehouse terpusat

Keuntungan dari bentuk ini adalah data benar-benar terpadu karena konsistensinya yang tinggi sedang kerugiannya adalah biaya yang mahal serta memerlukan waktu yang cukup lama untuk membangunnya.

3. Data warehouse terdistribusi

Pada data warehouse terdistribusi ini, digunakan gateway yang berfungsi sebagai jembatan penghubung antara data warehouse dengan workstation yang menggunakan sistem beraneka ragam. Dengan sistem terdistribusi seperti ini memungkinkan perusahaan dapat mengakses sumber data yang berada diluar lokasi perusahaan(eksternal).

Keuntungannya adalah data tetap konsisten karena sebelum data digunakan data terlebih dahulu di sesuaikan atau mengalami proses sinkronisasi. Sedangkan kerugiannya adalah lebih kompleks untuk diterapkan karena sistem operasi dikelola secara terpisah juga biayanya yang paling mahal dibandingkan dengan dua bentuk data warehouse lainnya.



Gambar 3.3 Bentuk data warehouse terpusat

B. Arsitekur Data Warehouse

Menurut Poe, arsitektur adalah sekumpulan atau struktur yang memberikan kerangka untuk keseluruhan rancangan suatu sistem atau produk. Ada arsitektur *client-server*, arsitektur *networking* dan masih banyak arsitektur lainnya. Arsitektur data menyediakan kerangka dengan mengidentifikasikan dan memahami bagaimana data akan pindah melalui sistem dan digunakan dalam perusahaan. Arsitektur data untuk *data warehouse* mempunyai komponen utama yaitu *read-only* basisdata.

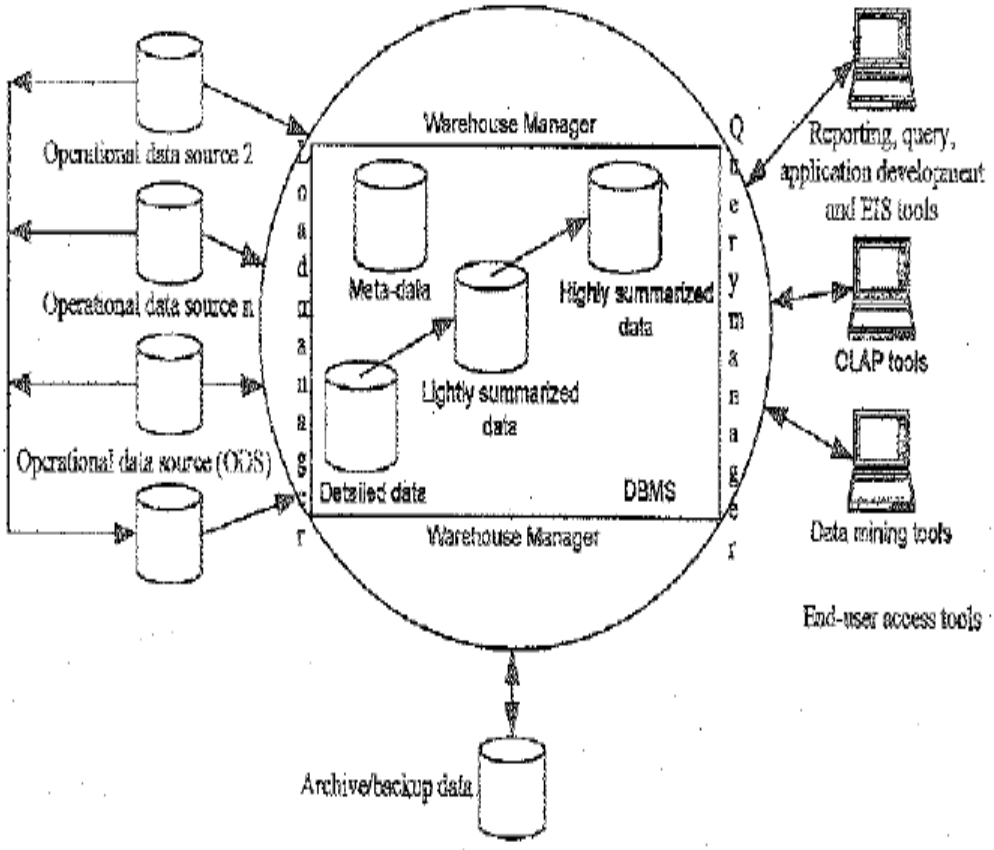
Karakterisitik arsitektur *data warehouse (*Poe) :

- Data diambil dari sistem asal (sistem informasi yang ada), basisdata dan file.
- Data dari sistem asal diintegrasikan dan ditransformasi sebelum disimpan ke dalam *Basisdata Management System* (DBMS) seperti Oracle, Ms SQL Server, Sybase dan masih banyak yang lainnya.
- *Data warehouse* merupakan sebuah basisdata terpisah bersifat hanya dapat dibaca yang dibuat khusus untuk mendukung pengambilan keputusan
- Pemakai mengakses *data warehouse* melalui aplikasi *front end tool*

Arsitektur dan komponen utama dari *data warehouse* dapat dilihat pada gambar 3.4. Komponen utama datawarehouse meliputi:

- *Operational Data*, Sumber data dari *data warehouse* dapat diambil langsung dari *mainframe,* basis data relasional seperti Oracle, Ms SQL server dan sebagainya. Selain itu dapat melaluo *Operational Data Source(*ODS). ODS menampung data yang diekstrak dari sistem utama atau sumber-sumber data yang ada dan kemudian data hasil ekstrasi tersebut dibersihkan.
- *Load manager,* disebut sebagai komponen *front-end* yang bertugas melakukan seluruh operasi yang berhubungan dengan ekstrasi dan me-*load* data ke *warehouse.*

- *Warehouse Manager*, komponen ini melakukan seluruh operasi-operasi yang berhubungan dengan kegiatan manajemen data di dalam *warehouse.* Operasi-operasi tersebut meliputi :
  - Analisis terhadap data untuk memastikan konsistensi
  - Transformasi dan penggabungan sumber data dari tempat penyimpanan sementara menjadi tabel-tabel *data warehouse.*
  - Penciptaan indeks-indeks dan view berdasarkan tabel-tabel dasar
  - Melakukan denormalisasi dan agregasi jika diperlukan
  - *Backing-Up* dan mengarsipkan data



Gambar 3.4 Komponen utama data warehouse

Sumber : Conolly,T.M.,Begg

- *Query manager*, juga disebut komponen *back-end*, melakukan operasi-operasi yang berhubungan dengan manajemen *user queries.* Operasi-operasi yang dilakukan oleh komponen ini termasuk mengarahkan *query* kepada tabel-tabel yang tepat dan menjadwalkan eksekusi dari query tersebut.
- *End-user Access Tools*, prinsip atau tujuan utama dari dibangunnya *data warehouse* adalah untuk menyediakan informasi bisnis kepada *user-user* untuk dapat melakukan pengambilan keputusan secara cepat dan tepat.*User* ini berinteraksi dengan *warehouse* melalui *end-user access tools. Data warehouse* harus secara efisien mendukung secara khusus kebutuhan *user* serta secara rutin melakukan analisis. Performa yang baik dapat dicapai dengan merencanakan dahulu keperluan-keperluan untuk melakukan *joins, summations* dan laporan-laporan per periode dengan *end-users.*

Berdasarkan kategori yang dikemukakan oleh Berson dan Smith terdapat lima grup utama dari tools tersebut, antara lain :

- *Reporting and query tools*
- *Application development tools*
- *Executive information System (EIS) tools*
- *Online Analytical Processing (OLAP) tools*
- *Data mining tools*

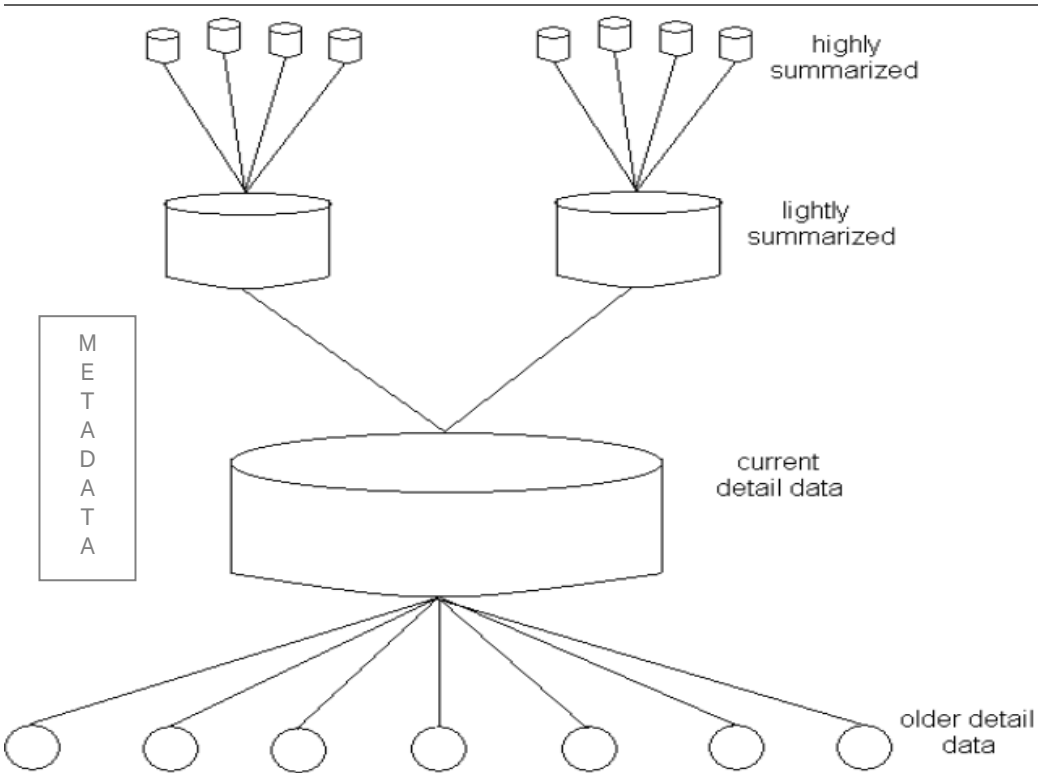
Arsitektur dan infrastruktur dari *data warehouse* sangat erat hubungannya dan satu dengan lainnya saling berkaitan.

C. Infrastruktur Data Warehouse

Infrastruktur *data warehouse* adalah software, hardware, pelatihan dan komponen-komponen lainnya yang memberikan dukungan yang dibutuhkan untuk mengimplementasikan *data warehouse*(Poe).

Salah satu instrumen yang mempengaruhi keberhasilan pengembangan *data warehouse* adalah pengidentifikasian arsitektur mana yang terbaik dan infrasruktur apa yang dibutuhkan. Arsitektur yang sama, mungkin memerlukan infrastruktur yang berbeda, tergantung pada lingkunan perusahaan ataupun organisasi.

D. Struktur Data Warehouse



Gambar 3.5 Komponen utama data warehouse

Seperti yang kita lihat sebelumnya pada arsitektur *data warehouse*, ada beberapa struktur yang spesifik terdapat pada bagian *warehouse manager*. Bagian tersebut merupakan struktur data warehouse.

Menurut Poe, Vidette, data warehouse memiliki struktur yang spesifik dan mempunyai perbedaan dalam tingkatan detail data dan umur data.

Komponen dari struktur data warehouse adalah:

1. ***Current detail data***

   Current detail data merupakan data detil yang aktif saat ini,mencerminkan keadaan yang sedang berjalan dan merupakan level terendah dalam *data warehouse.* Didalam area ini warehouse menyimpan seluruh detail data yang terdapat pada skema basis data. Jumlah data sangat besar sehingga memerlukan *storage* yang besar pula dan dapat diakses secara cepat. Dampak negatif yang ditimbulkan adalah kerumitan untuk mengatur data menjadi meningkat dan biaya yang diperlukan menjadi mahal.

   Berikut ini beberapa alasan mengapa *current detail data* menjadi perhatian utama :

   - Menggambarkan kejadian yang baru terjadi dan selalu menjadi perhatian utama
   - Sangat banyak jumlahnya dan disimpan pada tingkat penyimpanan terendah.
   - Hampir selalu disimpan dalam *storage* karena cepat di akses tetapi mahal dan kompleks dalam pengaturannya.
   - Bisa digunakan dalam membuat rekapitulasi sehingga *current detail data* harus akurat.

2. ***Older detail data***

   Data ini merupakan data historis dari *current detail data*, dapat berupa hasil cadangan atau archive data yang disimpan dalam *storage* terpisah. Karena bersifat *back-up(*cadangan), maka biasanya data disimpan dalam *storage* alternatif seperti *tape-desk.*

   Data ini biasanya memilki tingkat frekuensi akses yang rendah. Penyusunan file atau *directory* dari data ini di susun berdasarkan umur dari data yang bertujuan mempermudah untuk pencarian atau pengaksesan kembali.

3. ***Lighlty summarized data***

   Data ini merupakan ringkasan atau rangkuman dari *current detail data.* Data ini dirangkum berdasar periode atau dimensi lainnya sesuai dengan kebutuhan.

   Ringkasan dari *current detail data* belum bersifat total *summary*.Data-data ini memiliki detil tingkatan yang lebih tinggi dan mendukung kebutuhan *warehouse* pada tingkat departemen. Tingkatan data ini di sebut juga dengan *data mart.* Akses terhadap data jenis ini banyak digunakan untuk view suatu kondisi yang sedang atau sudah berjalan.

4. ***Highly summarized data***

   Data ini merupakan tingkat lanjutan dari *Lightly summarized data,* merupakan hasil ringkasan yang bersifat totalitas, dapat di akses misal untuk melakukan analisis perbandingan data berdasarkan urutan waktu tertentu dan analisis menggunakan data multidimensi.

5. ***Metadata***

   Metadata bukan merupakan data hasil kegiatan seperti keempat jenis data diatas. Menurut Poe, metadata adalah ‘data tentang data’ dan

menyediakan informasi tentang struktur data dan hubungan antara struktur data di dalam atau antara *storage*(tempat penyimpanan data).

Metadata berisikan data yang menyimpan proses perpindahan data meliputi basisdata structure,contents,detail data dan summary data, matrics,versioning, aging criteria,versioning, transformation criteria. Metadata khusus dan memegang peranan yang sangat penting dalam data warehouse.

Metadata sendiri mengandung :

- *Struktur data*,
  Sebuah direktori yang membantu user untuk melakukan analisis *Decission Support System* dalam pencarian letak/lokasi dalam data warehouse.
- *Algoritma*, Algoritma digunakan untuk summary data. Metadata sendiri merupakan panduan untuk algoritma dalam melakukan pemrosesan *summary* data antara *current detail data* dengan *lightly summarized data* dan antara *lightly summarized data* dengan hightly summaried data.
- *Mapping*, Sebagai panduan pemetaan(*mapping)* data pada saat data di transform/diubah dari lingkup operasional menjadi lingkup data warehouse.

E. Metodologi Perancangan Basisdata untuk *Data Warehouse*

Menurut Kimball ada sembilan tahap metodologi dalam perancangan basisdata untuk *data warehouse*, yaitu :

1 . Pemilihan proses

Data mart yang pertama kali dibangun haruslah data mart yang dapat dikirim tepat waktu dan dapat menjawab semua pertanyaan bisnis yang

penting. Pilihan terbaik untuk data mart yang pertama adalah yang berhubungan dengan *sales,* misal *property sales, property leasing,property advertising.*

2. Pemilihan sumber

   Untuk memutuskan secara pasti apa yang diwakili atau direpresentasikan oleh sebuah tabel fakta. Misal, jika sumber dari sebuah tabel fakta properti sale adalah properti sale individual maka sumber dari sebuah dimensi pelanggan berisi rincian pelanggan yang membeli properti utama

3 . Mengidentifikasi dimensi

   Set dimensi yang dibangun dengan baik, memberikan kemudahan untuk memahami dan menggunakan data mart. Dimensi ini penting untuk menggambarkan fakta-fakta yang terdapat pada tabel fakta. Misal, setiap data pelanggan pada tabel dimensi pembeli dilengkapi dengan id_pelanggan,no_pelanggan, tipe_pelanggan, tempat_tinggal, dan lain sebagainya. Jika ada dimensi yang muncul pada dua data mart, kedua data mart tersebut harus berdimensi sama, atau paling tidak salah satunya berupa subset matematis dari yang lainnya. Jika sebuah dimensi digunakan pada dua data mart atau lebih,dan dimensi ini tidak disinkronisasi,maka keseluruhan *data warehouse* akan gagal, karena dua data mart tidak bisa digunakan secara bersama-sama

4. Pemilihan fakta

   Sumber dari sebuah tabel fakta menentukan fakta mana yang bisa digunakan dalam data mart. Semua fakta harus diekspresikan pada tingkat yang telah ditentukan oleh sumber

5. Menyimpan pre-kalkulasi di tabel fakta

   Hal ini terjadi apabila fakta kehilangan statement

6. Melengkapi tabel dimensi

   Pada tahap ini kita menambahkan keterangan selengkap-lengkapnya pada tabel dimensi. Keterangannya harus bersifat intuitif dan mudah dipahami oleh pengguna

7. Pemilihan durasi basisdata

   Misalnya pada suatu perusahaan asuransi, mengharuskan data disimpan selama 10 tahun atau lebih

8. Menelusuri perubahan dimensi yang perlahan

   Ada tiga tipe perubahan dimensi yang perlahan, yaitu :

   - Tipe 1. Atribut dimensi yang telah berubah tertulis ulang
   - Tipe 2. Atribut dimensi yang telah berubah menimbulkan sebuah dimensi baru
   - Tipe 3. Atribut dimensi yang telah berubah menimbulkan alternatif sehingga nilai atribut lama dan yang baru dapat diakses secara bersama pada dimensi yang sama.

9. Menentukan prioritas dan mode *query*

   Pada tahap ini kita menggunakan perancangan fisik.Dengan langkah-langkah tadi, seharusnya kita bisa membangun sebuah *data warehouse* yang baik.

F. Model untuk Data Warehouse

Berikut di bawah ini adalah penjelasan dari model untuk data warehouse

1. Model Dimensional

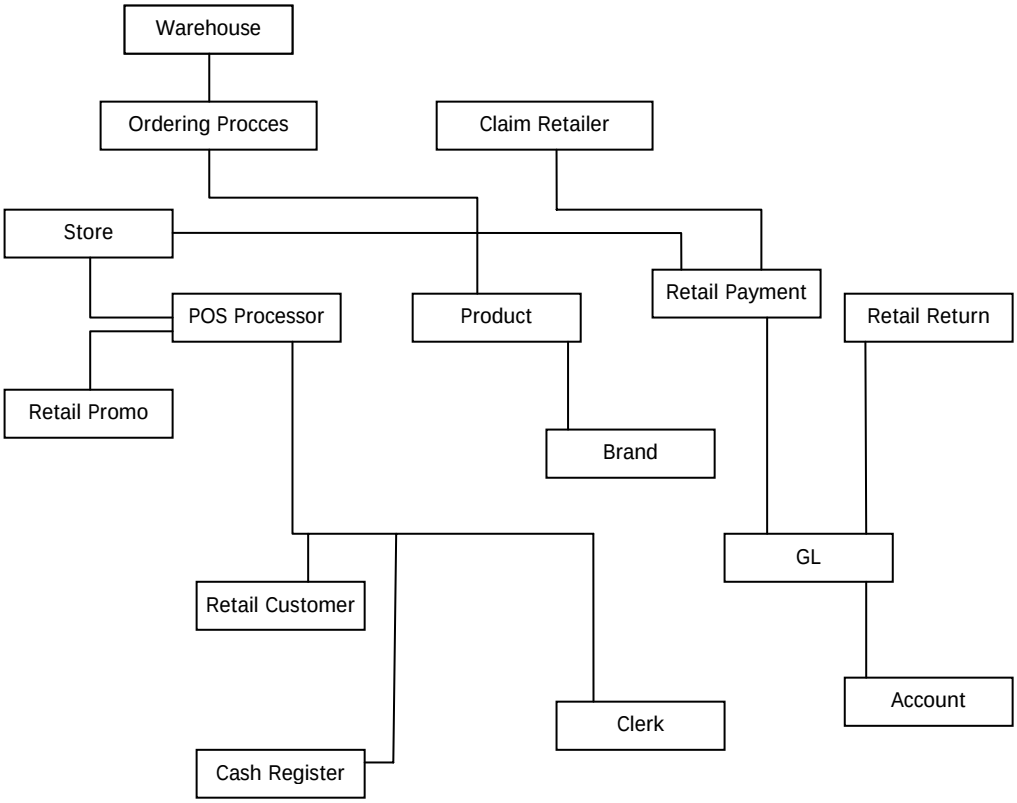
Model dimensional merupakan rancangan logikal yang bertujuan untuk menampilkan data dalam bentuk standar dan intuitif yang memperbolehkan akses dengan performa yang tinggi.

Model dimensional menggunakan konsep model hubungan antar entity (ER) dengan beberapa batasan yang penting. Setiap model dimensi terdiri dari sebuah tabel dengan sebuah komposit *primary key,* disebut dengan table fakta, dan satu set table yang lebih kecil disebut table dimensi. Setiap table dimensi memiliki sebuah *simple primary key* yang merespon tepat pada satu komponen *primary key* pada tabel fakta. Dengan kata lain *primary key* pada table fakta terdiri dari dua atau lebih *foreign key.* Struktur karakteristik ini disebut dengan skema bintang atau join bintang.

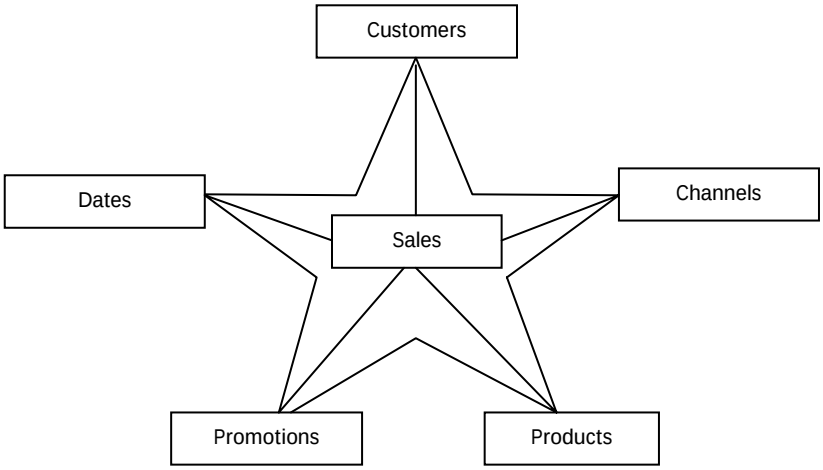
Fitur terpenting dalam model dimensional ini adalah semua *natural keys* diganti dengan kunci pengganti(*surrogate keys).* Maksudnya yaitu setiap kali join antar table fakta dengan table dimensi selalu didasari kunci pengganti. Kegunaan dari kunci pengganti adalah memperbolehkan data pada *data warehouse* untuk memiliki beberapa kebebasan dalam penggunaan data, tidak seperti halnya yang diproduksi oleh sistem OLTP.

Sebuah sistem OLTP memerlukan normalisasi untuk mengurangi redudansi, validasi untuk input data, mendukung volume yang besar dari transaksi yang bergerak sangat cepat. Model OLTP sering terlihat seperti jaring laba-laba yang terdiri atas ratusan bahkan ribuan tabel sehingga sulit untuk dimengerti.

Sebaliknya, dimension model yang sering digunakan pada *data warehouse* adalah skema bintang atau *snowflake* yang mudah dimengerti dan sesuai dengan kebutuhan bisnis, mendukung *query* sederhana dan menyediakan performa *query* yang superior dengan meminimalisasi tabel-tabel join. Berikut contoh perbandingan diagram antara model data OLTP dengan dimension table data warehouse :



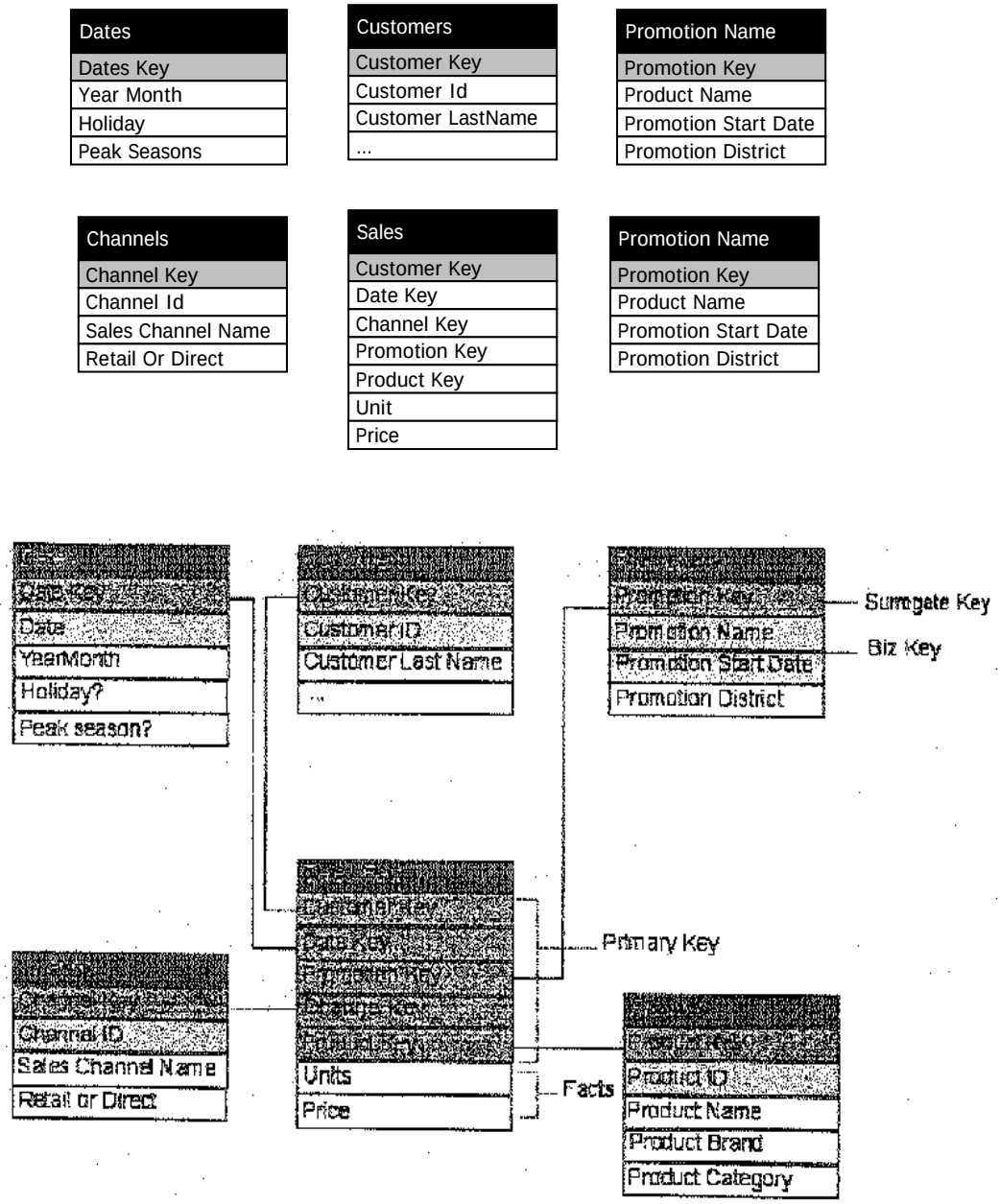
Gambar 3.6. Model data OLTP



Gambar 3.7. Dimension Model

2. Schema Bintang

Skema bintang merupakan struktuk logikal yang memiliki tabel fakta yang terdiri atas data faktual ditengahnya, dan dikelilingi oleh tabel-tabel dimensi yang berisi referensi data.



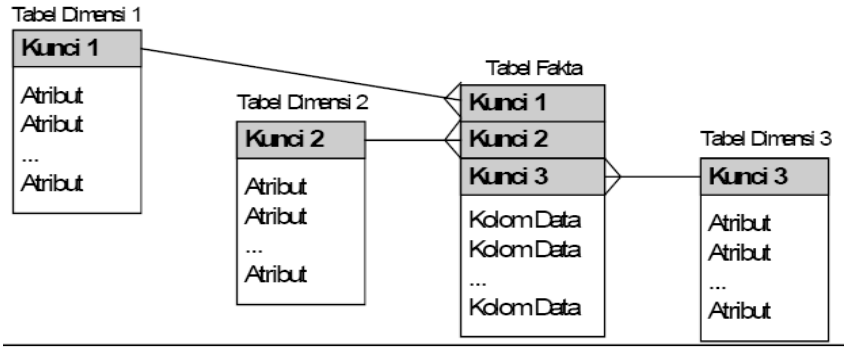
Gambar 3.8 Schema Bintang

Jenis-jenis Skema Bintang

- Skema bintang sederhana

Dalam skema ini, setiap table harus memiliki *primary key* yang terdiri dari satu kolom atau lebih.

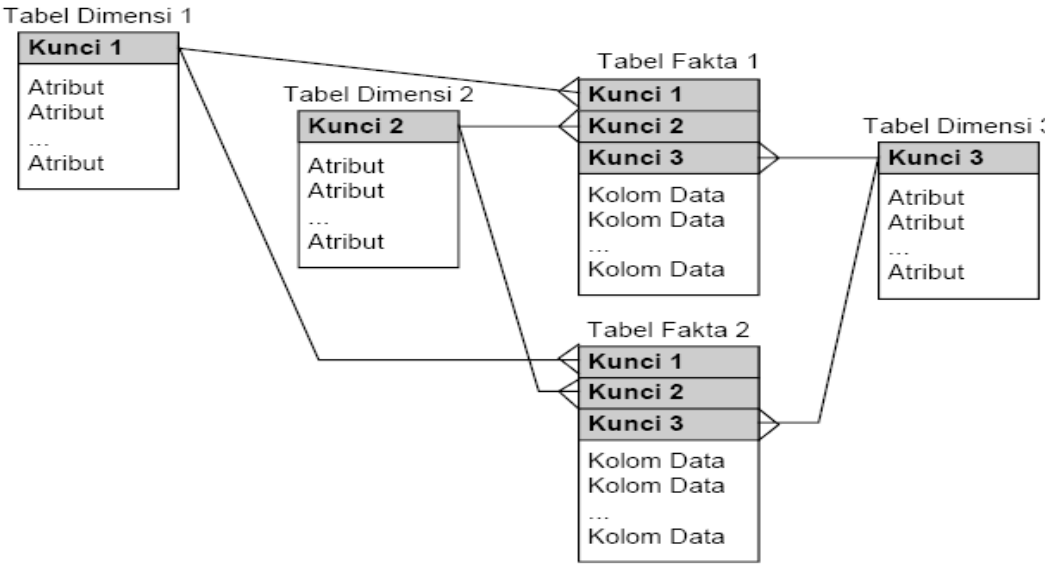
Primary key dari table fakta terdiri dari satu atau lebih *foreign key*.*Foreign key* merupakan *primary key* pada table lain.



Gambar 3.8 Schema bintang sederhana

- Skema bintang dengan banyak table fakta

  Skema bintang juga bisa terdiri dari satu atau lebih table fakta. Dikarenakan karena table fakta tersebut ada banyak, misalnya disamping penjualan terdapat table fakta *forecasting* dan *result.* Walaupun terdapat lebih dari satu table fakta, mereka tetap menggunakan table dimensi bersama-sama.



Gambar 3.9 Skema bintang dengan banyak tabel fakta

Adapun ketentuan dalam pembacaan skema bintang adalah :

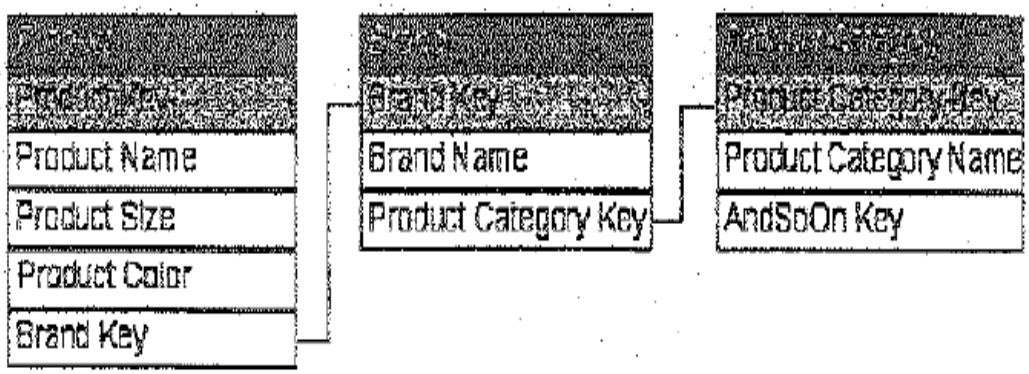
- Bagian yang ada di bawah judul tabel merupakan kolom-kolom tabel tersebut
- Primary key dan Foreign key diberi kotak
- Primary key diarsir sedang Foreign key yang bukan primary tidak
- Foreign key yang berhubungan ditunjukkan dengan garis yang menghubungkan tabel.

Kolom yang bukan kunci disebut kolom data pada table fakta dan atribut pada table dimensi

3 Snowflake Schema

Merupakan varian dari skema bintang dimana table-table dimensi tidak terdapat data yang di denormalisasi. Dengan kata lain satu atau lebih table dimensi tidak bergabung secara langsung kepada table fakta tapi pada table dimensi lainnya. Sebagai contoh, sebuah dimensi yang

mendeskripsikan produk dapat dipisahkan menjadi tiga table *(snowflaked)* seperti contoh dibawah ini :



Gambar 3.10. Snowflake Schemes

4. Star atau Snowflake

Keduanya merupakan model-model dimensional, perbedaannya terletak pada implementasi fisikal. Skema snowflake memberi kemudahan pada perawatan dimensi, dikarenakan strukturnya yang lebih normalisasi. Sedangkan skema bintang lebih efisien serta sederhana dalam membuat query dan mudah diakses secara langsung oleh pengguna.

Adapun starflake merupakan gabungan diantara keduanya. Keuntungan menggunakan masing-masing model tersebut dalam data warehouse antara lain :

- Efisien dalam hal mengakses data
- Dapat beradaptasi terhadap kebutuhan-kebutuhan user
- Bersifat fleksibel terhadap perubahan yang terjadi khususnya perubahan yang mengarah pada perkembangan
- Memiliki kemampuan dalam memodelkan situasi bisnis secara umum
- Meskipun skema yang dihasilkan sangat kompleks, tetapi pemrosesan query dapat diperkirakan, hal ini dikarenakan pada level terendah, setiap table fakta harus di query secara independen.