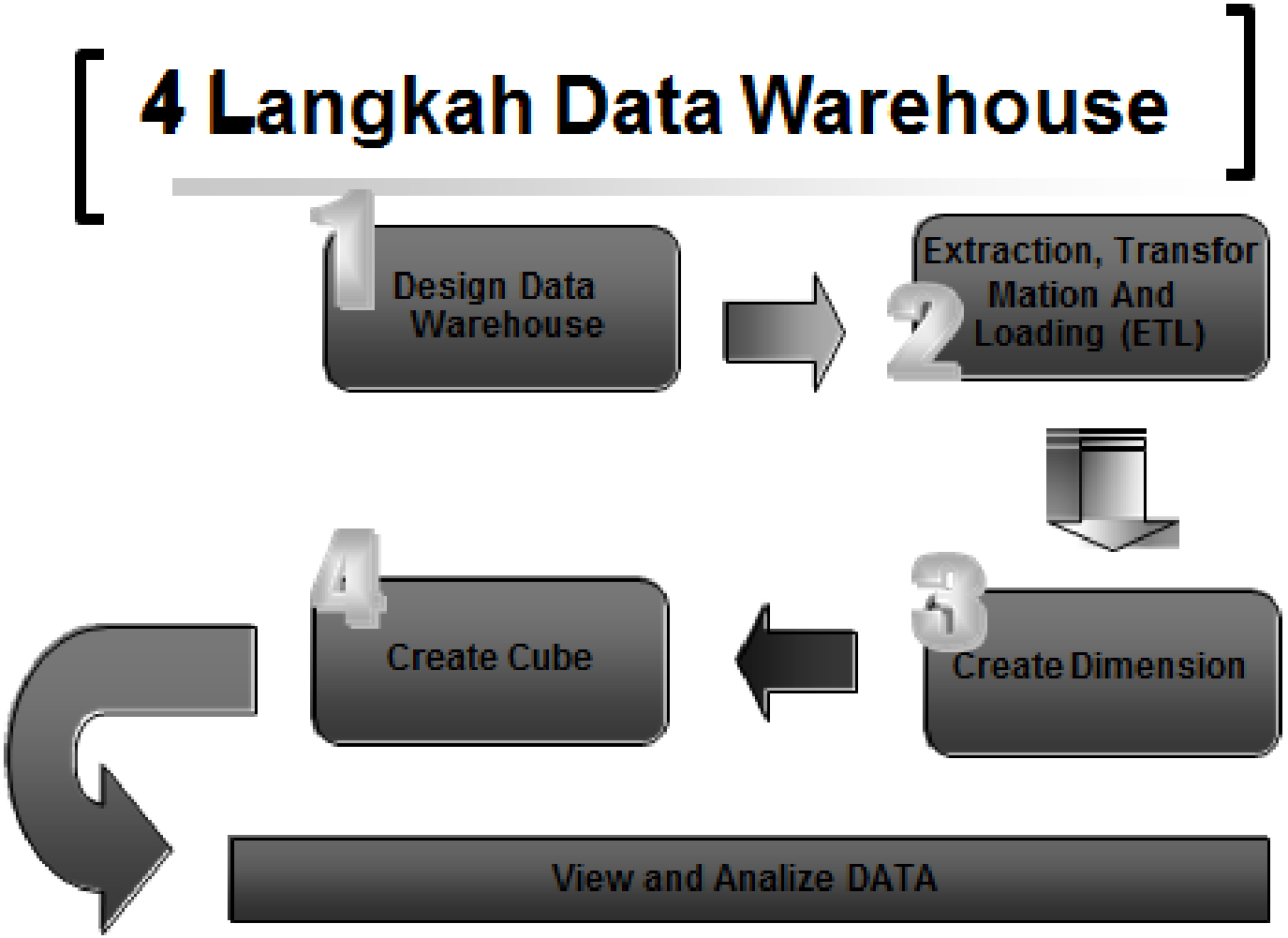
DESAIN WAREHOUSE

LANGKAH MEMBANGUN WAREHOUSE



1. Desain Datawarehouse

Logical Design

- Menggunakan ER Diagram
- Mencari objek-objek penting(entitas)
- Mendefinisikan atribute
- Membuat relasi

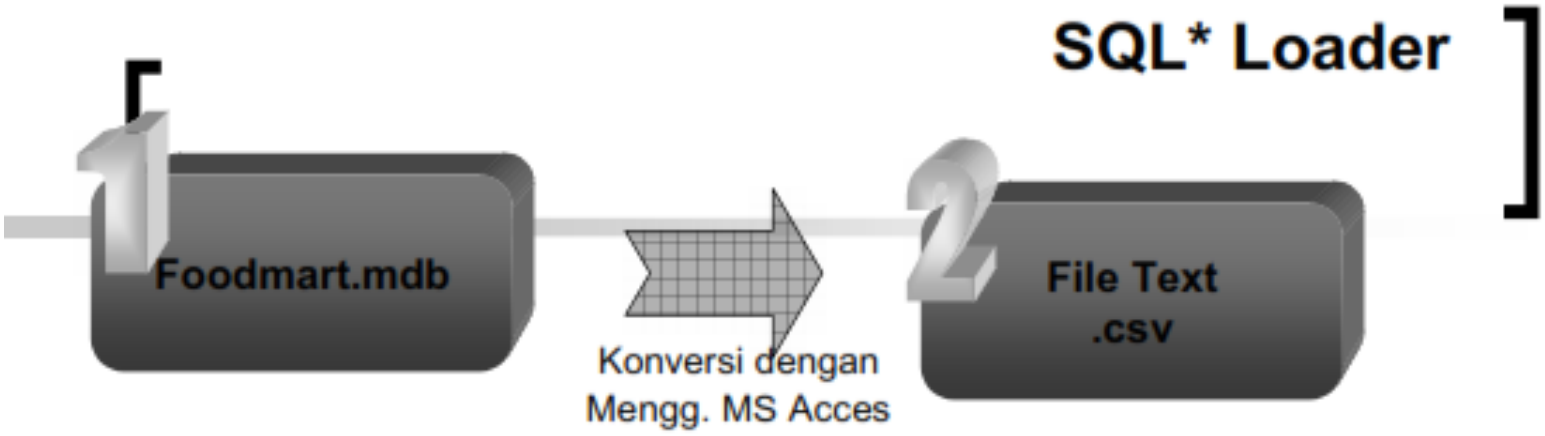
Physical Design

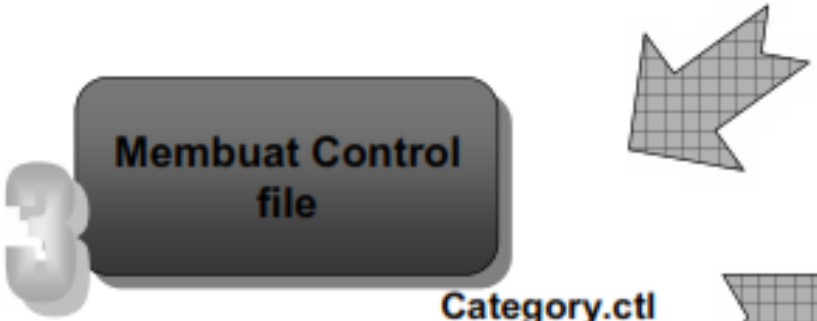
- Merubah dari logical design menjadi objek-objek database
- Tablespace, Table, integrity constraint, dll

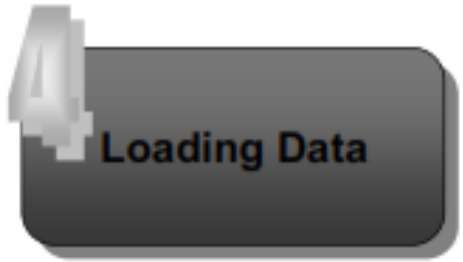
2. Extraction, Transformation and Loading (ETL)

- Adalah proses pengekstrakan data dari sumber data yang kemudian dimasukkan ke dalam data warehouse.
- dilakukan secara periodik untuk kebutuhan bisnis dengan analisa data yang akurat.
- Menggunakan Data Transformation
- Service (DTS) dari MS SQL Server
- Menggunakan SQL* Loader dari Oracle

SQL* Loader







Category.ctl

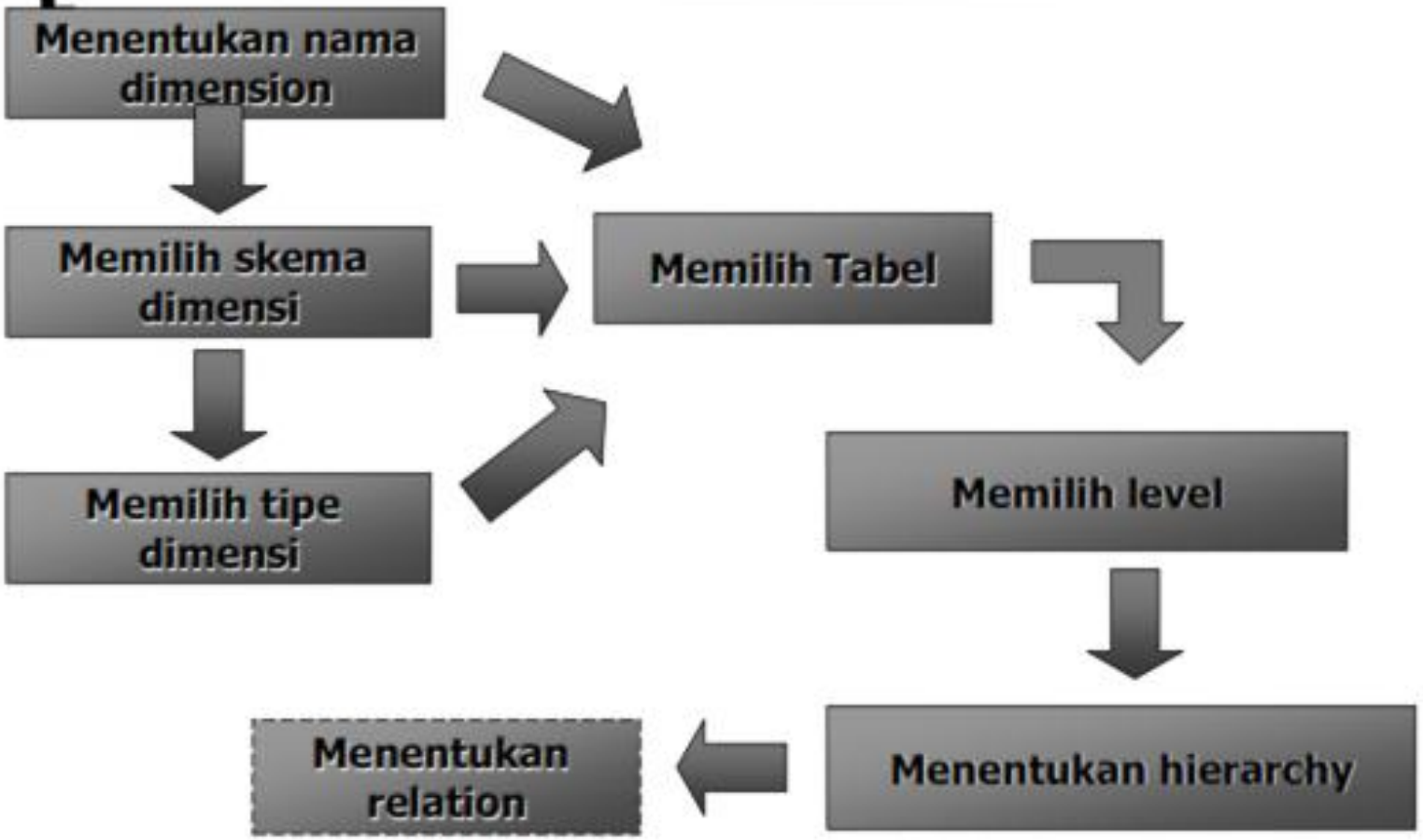
```
load data
  infile
'f:\category\category.csv'
  into table category
  fields terminated by ','
  (
category_id,category_parent,
category_description,categor
y_rollup )
```

```
C:> SQLLDR userid =
anang/anang control =
category.ctl
```

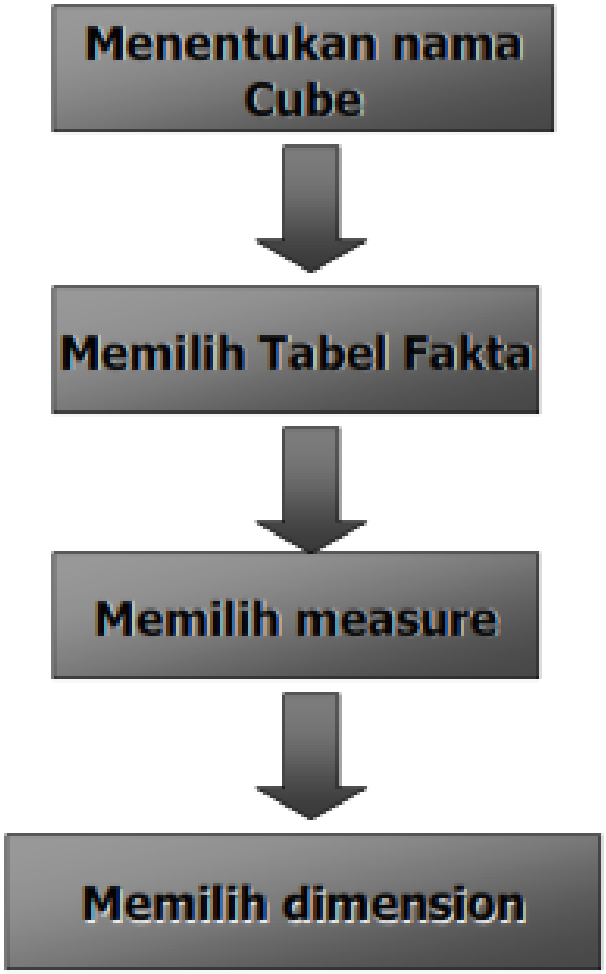
3. Membuat Dimension

- Dimension adalah sebuah struktur yang terbentuk dari satu atau lebih hirarki yang mengkategorisasi data
- Dimensi terbentuk dari satu atau lebih tabel. Setiap kolomnya merepresentasikan level pada *hierarchy*

Alur pembuatan Dimension



4. Membuat Cube



Desain Warehouse

- Pengantar
- Multidimensional Data Model
- Pertimbangan dalam Membuat Desain
- Implementasi Desain
- Mengetes Desain
- Contoh Desain

Pengantar

- Ingat: Database Warehouse TERPISAH dari
- Database untuk Operasional

Data Warehouse vs Data Transaksi

	Warehouse (OLAP)	Transaksi (OLTP)
Desain	Multidimensional Data Model	Entity Relationship Diagram (ERD)
Tujuan	Efisiensi Waktu Query	Efisiensi Space/Storage

Apakah *Multidimensional Modeling*?

- Subject Oriented
  - Melihat data dari **berbagai perspektif** (Stok, Penjualan, dll)
- Berisikan data-data yang:
  - Telah ter**valid**asi
  - **Historik**al (contoh: data dua tahun terakhir)
  - Ter**integrasi**
  - Mudah Di**akses**
- Direpresentasikan dalam bentuk **Data Cube**

Apakah Data Cube

- Adalah **representasi** kumpulan data dalam multi-dimensi
- Meskipun dinamakan cube, namun dapat merepresentasikan data dalam **N-dimensi**

Contoh Data Cube

- 0 dimensi:
  - Total penjualan sampai sekarang

Total	XXX

- 1 dimensi:
  - Total penjualan untuk **waktu** tertentu

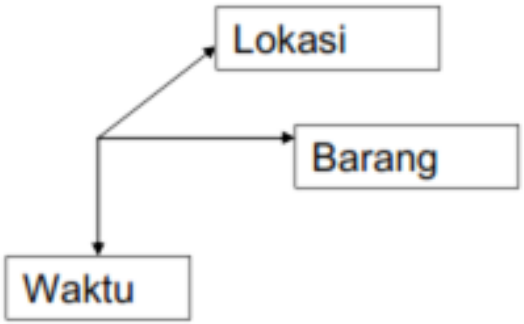
Waktu 1	AAA
Waktu 2	BBB
Waktu 3	CCC

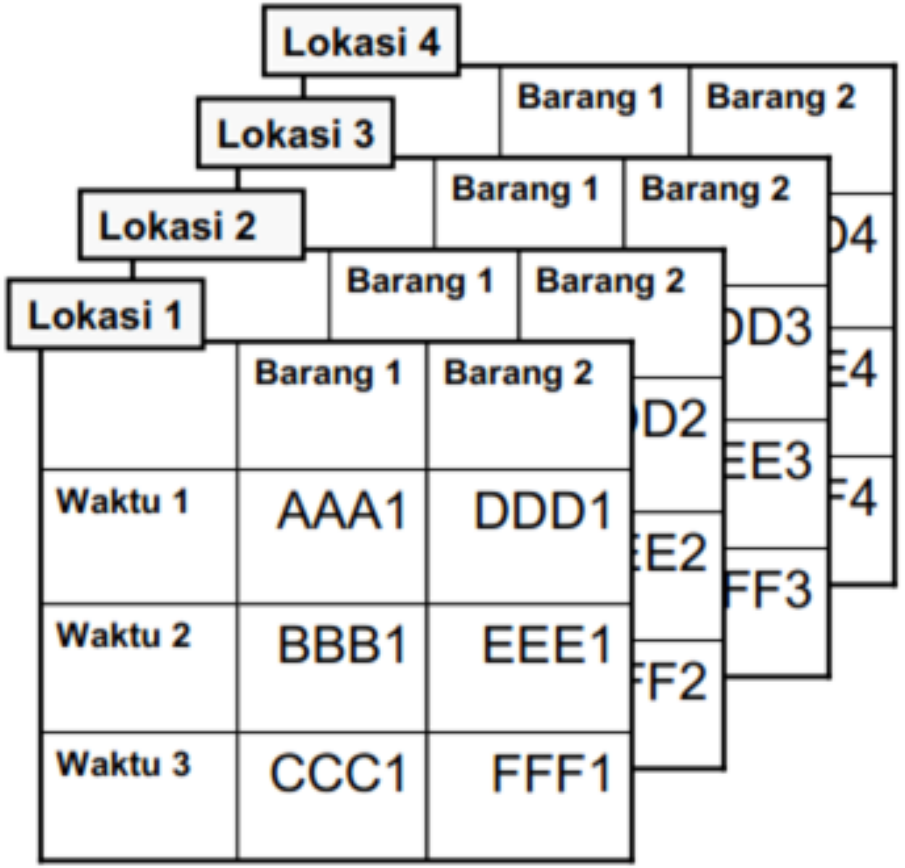
- 2 dimensi:
  - Total penjualan pada waktu tertentu untuk **barang** tertentu

	Barang 1	**Barang 2**
Waktu 1	AAA	DDD
Waktu 2	BBB	EEE
Waktu 3	CCC	FFF

Contoh Data Cube

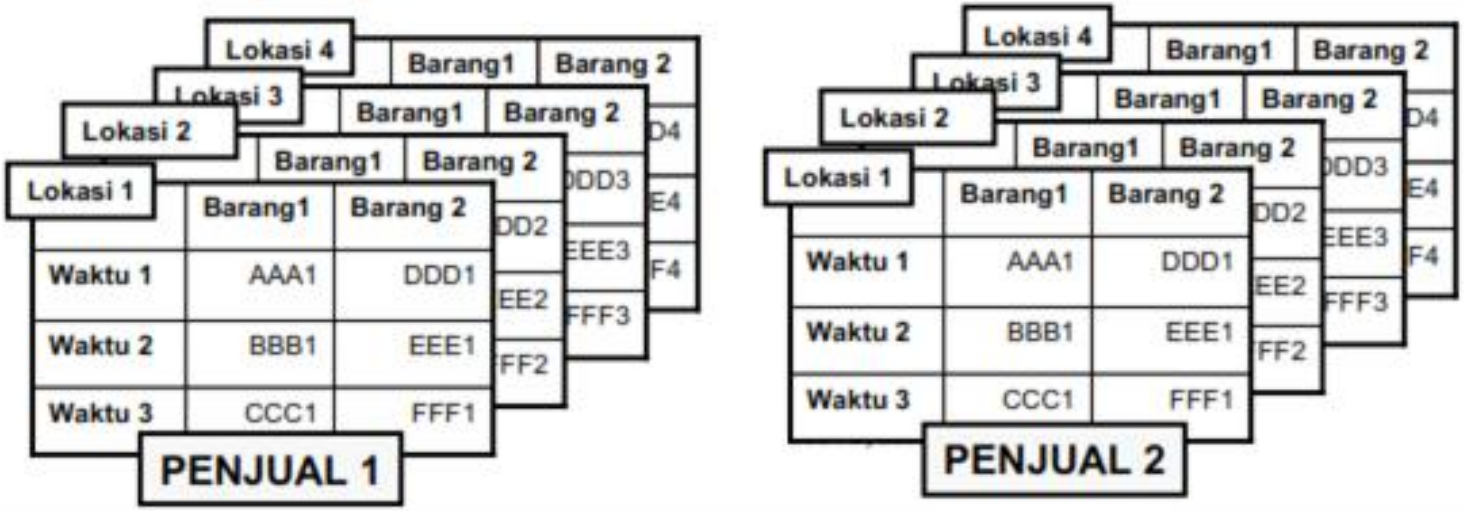
- 3 dimensi:
  - Total penjualan pada **waktu** tertentu untuk **barang** pada **lokasi** tertentu





Contoh Data Cube

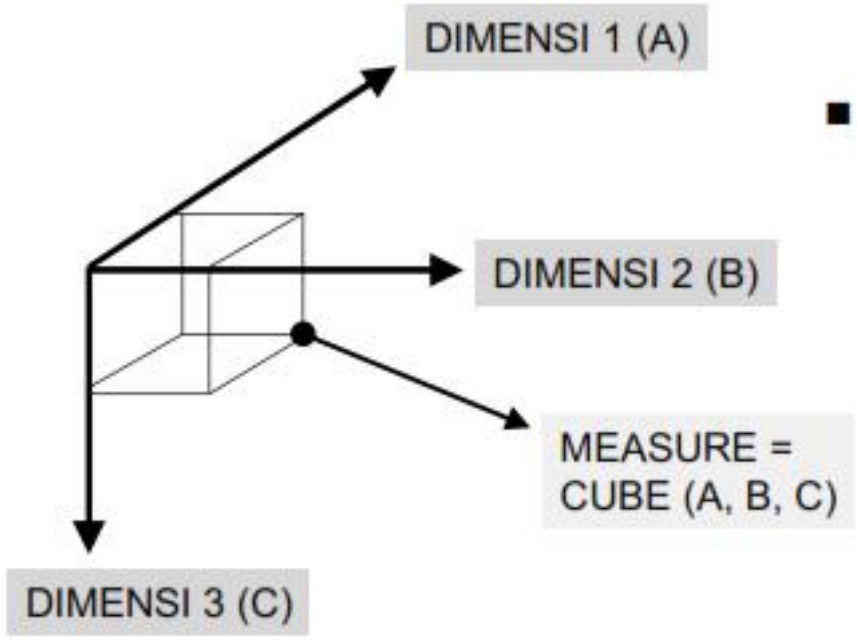
- 4 dimensi:
  - Total penjualan pada waktu tertentu untuk barang pada lokasi tertentu oleh **penjual** tertentu



Bagaimana Membuat *Multidimensional Data Model*?

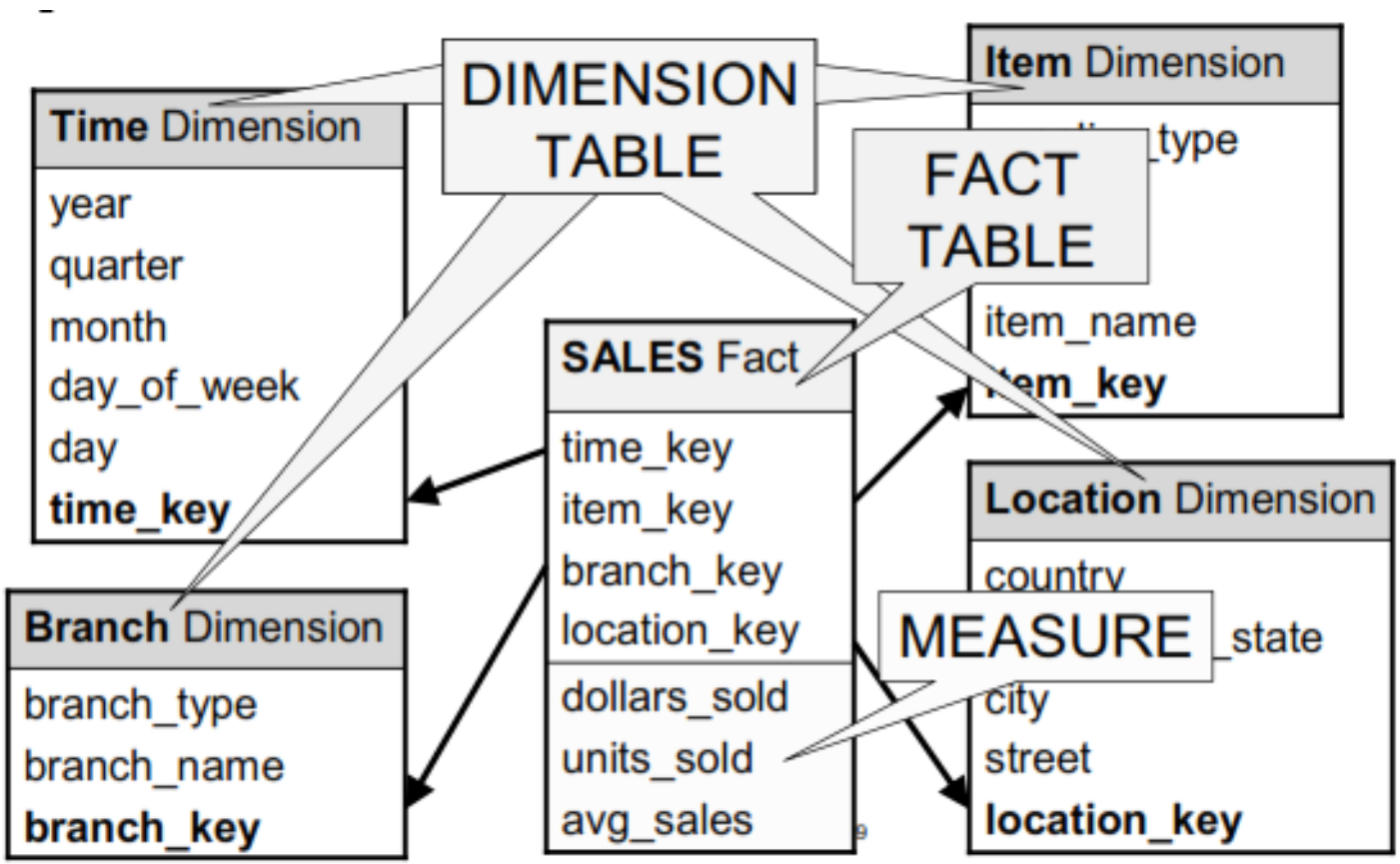
- **Tabel** dalam Model Data Multidimensi dibagi menjadi 2 macam:
  - Fact Table => Measure
  - Dimension Table => Atribut Sumbu
- **Skema** Model Data Multidimensi dibagi menjadi 3:
  - Star Schema
  - Snow Flake Schema
  - Fact Constellation Schema

Data Cube dan Skema Multidimensi

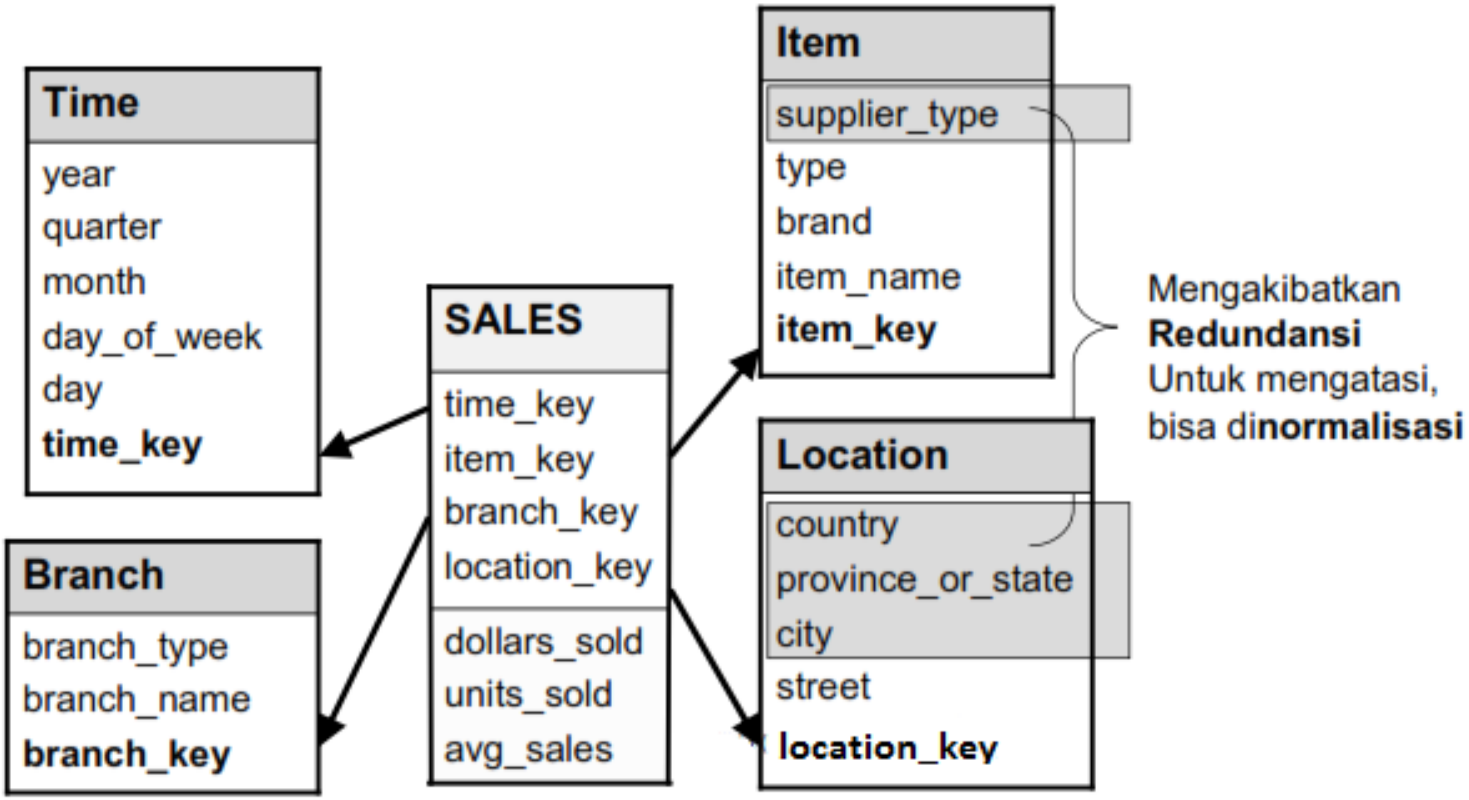


- **Fact** Table
  menyimpan Measure
- **Dimension** Table
  menyimpan Atribut Sumbu Cube / Dimensi
  - Dimensi 1 (A)
  - Dimensi 2 (B)
  - Dimensi 3 (C)

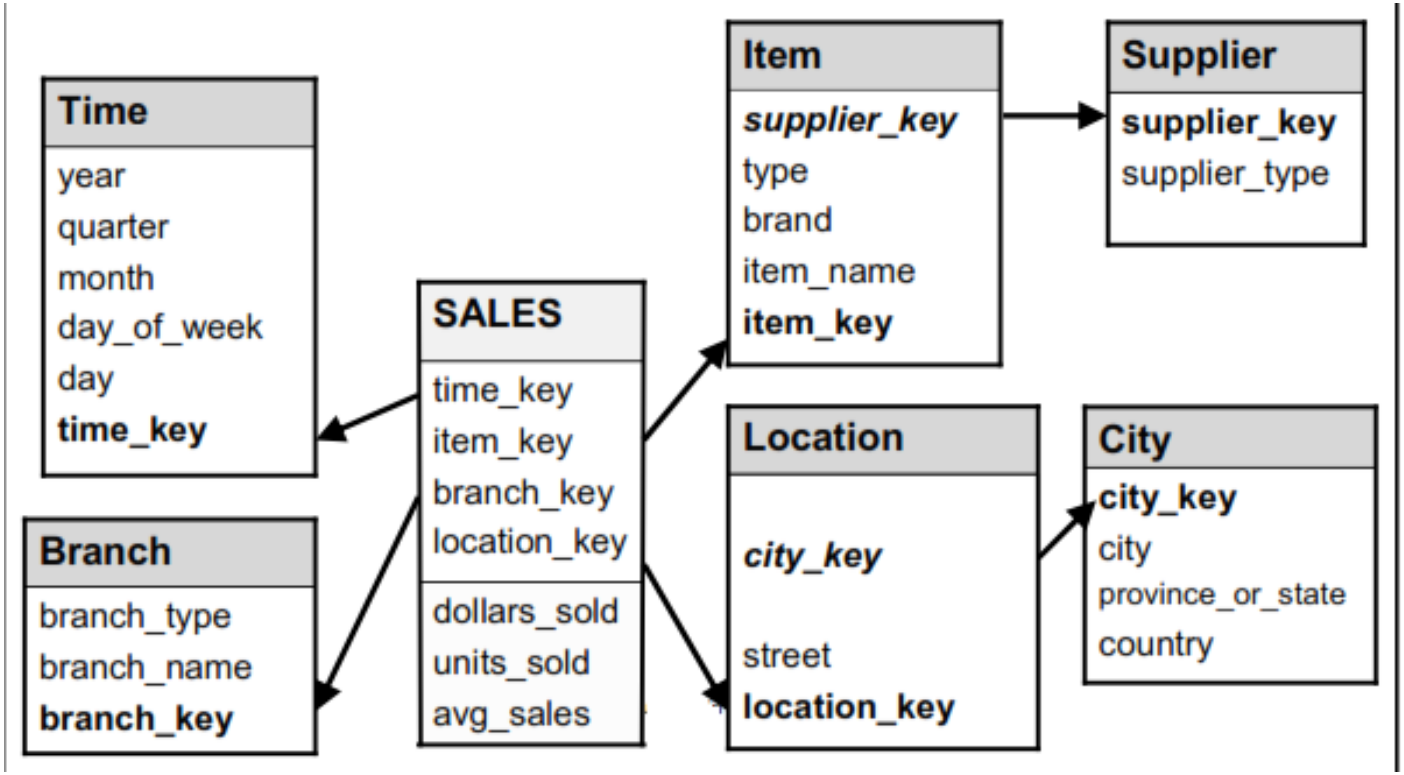
Star Schema


DIMENSION TABLE
FACT TABLE
MEASURE
Time Dimension
year
quarter
month
day_of_week
day
time_key
Item Dimension
type
item_name
em_key
SALES Fact
time_key
item_key
branch_key
location_key
dollars_sold
units_sold
avg_sales
Location Dimension
country
_state
city
street
location_key
Branch Dimension
branch_type
branch_name
branch_key

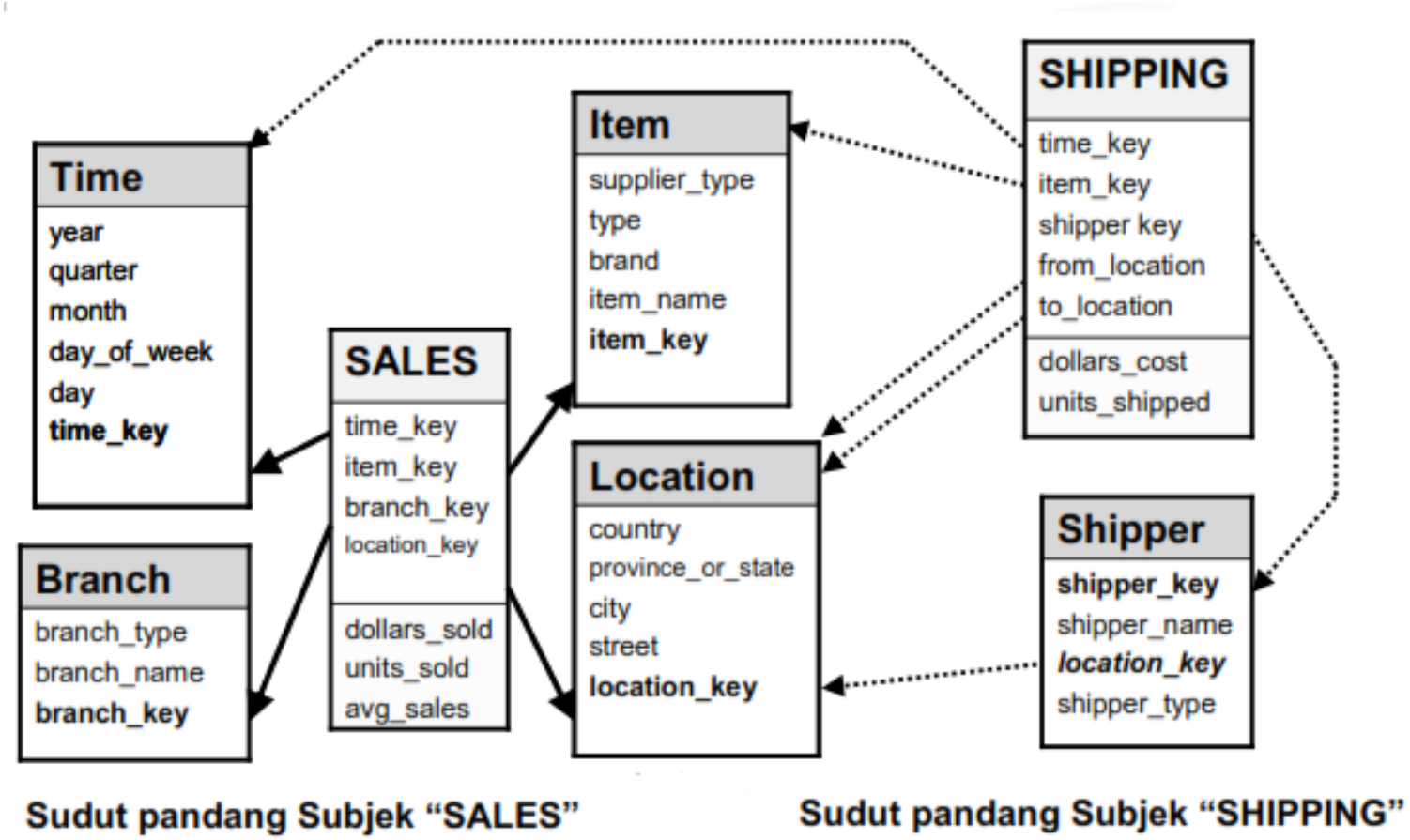
Star Schema


Time
year
quarter
month
day_of_week
day
time_key
Item
supplier_type
type
brand
item_name
item_key
SALES
time_key
item_key
branch_key
location_key
dollars_sold
units_sold
avg_sales
Location
country
province_or_state
city
street
location_key
Branch
branch_type
branch_name
branch_key
Mengakibatkan
Redundansi
Untuk mengatasi,
bisa dinormalisasi

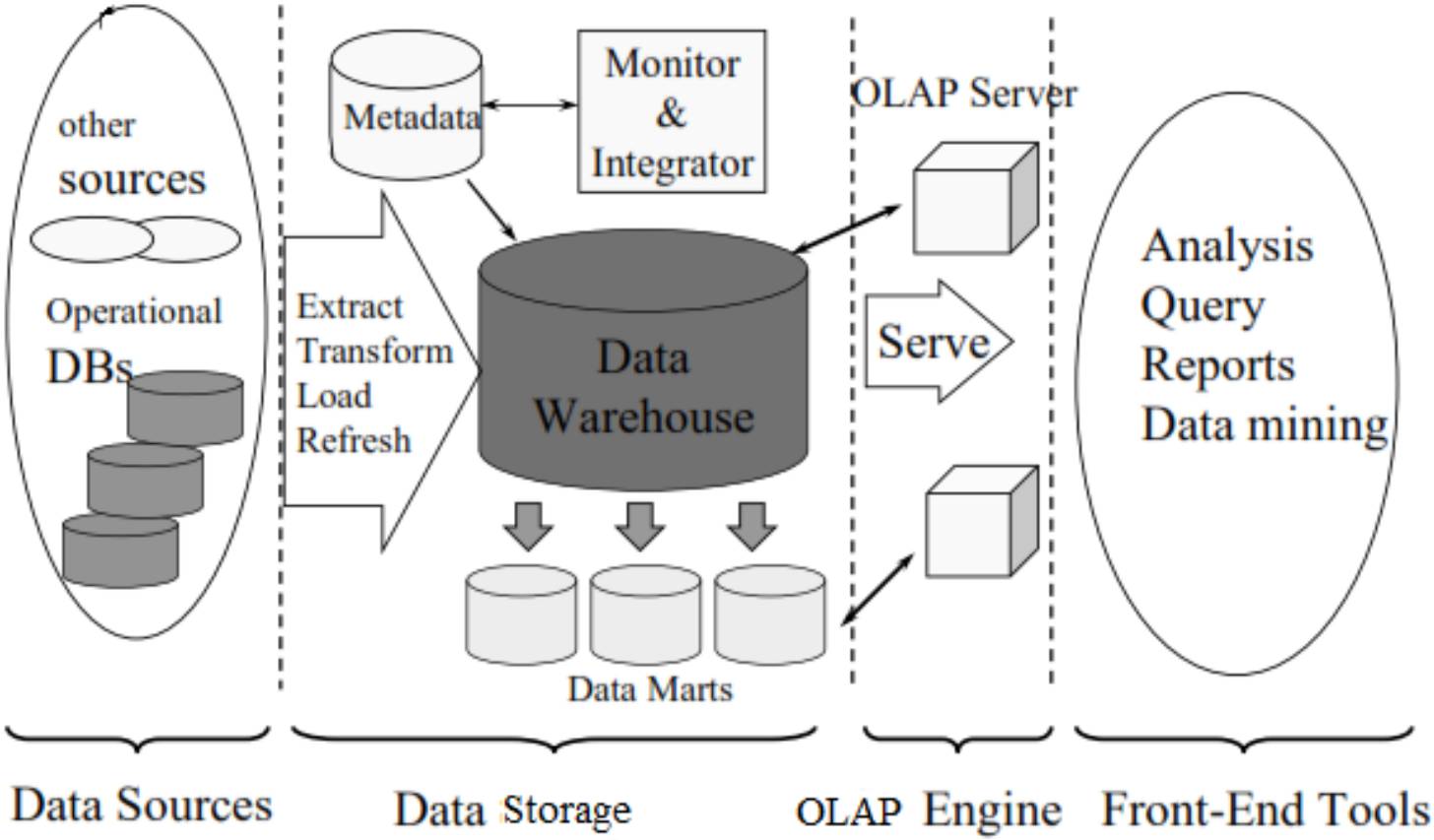
Snowflake Schema



Fact Constellation



Arsitektur Data Warehouse
Multi-Tiered



Pertimbangan dalam Membuat Desain Warehouse

- Desain untuk Pengelolaan
  - Mudah di **Backup** secara Teratur
  - Ketika **Loading** new data
  - Ketika **Aggregating** new data
  - Ketika Melakukan Aktifitas **Pemeliharaan** Data, contoh: Indexing dan Archiving
- Desain untuk Performa
  - Tentukan tipe, dimana, berapa banyak ruang yang dibutuhkan untuk **indeks**

Implementasi

- Satu Database atau Lebih?
- Kesepakatan dalam Aturan Penamaan?
- Membuat Database
- Menentukan Skema untuk Database
- Mengatur *Data File* dan *Tablespace*
- Membuat Tabel Fact dan Tabel Dimensi
- Konstrain
- Indeks
- Partisi
- Membuat View
- Keamanan

Testing

- Dilakukan sebelum rilis produksi
- Yang perlu di tes antara lain:
  - Waktu yang dibutuhkan untuk melakukan load data
  - Pembersihan data dan transformasi
  - Waktu respon *query*
  - Data *summary* yang dibutuhkan
  - Waktu yang dibutuhkan untuk tugas-tugas pengelolaan (manajemen)