Topic D

# Data Warehouse

D

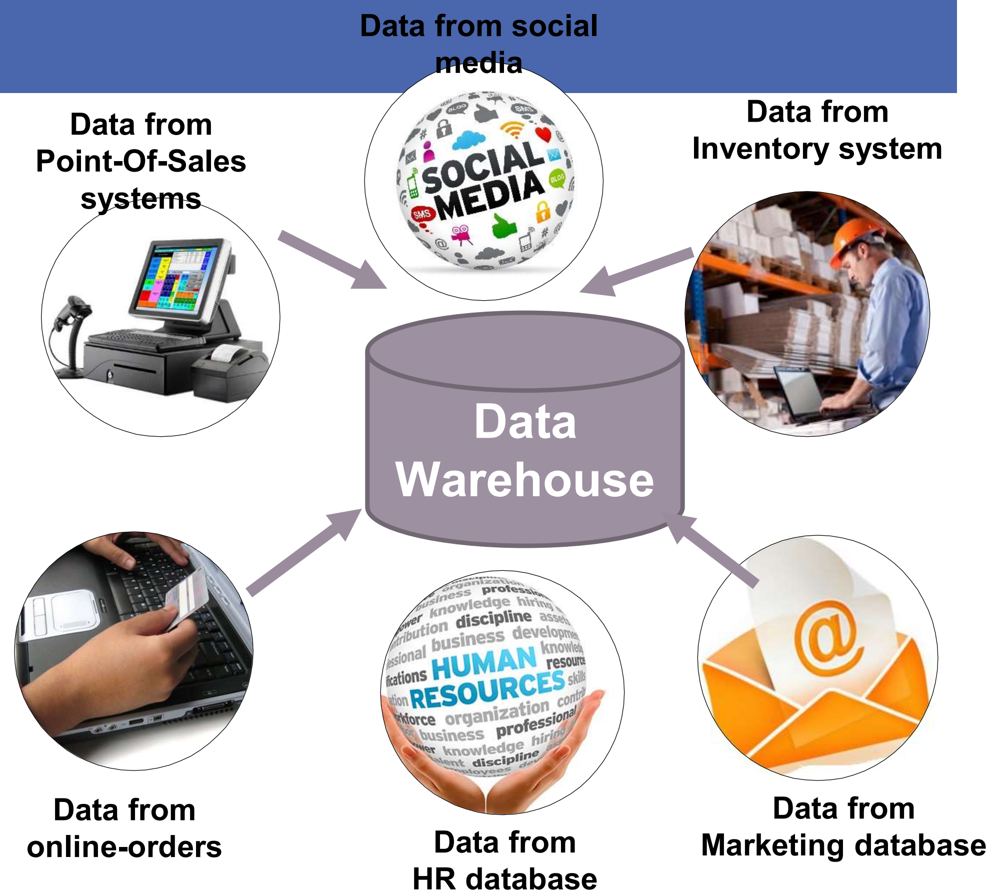
# Data Warehouse

## CONTENT

- What is a data warehouse?
- Why do we need a data warehouse?
- Characteristics of a data warehouse
- Dimension Model
- Slowly Changing Dimension
- Fast Changing Dimension

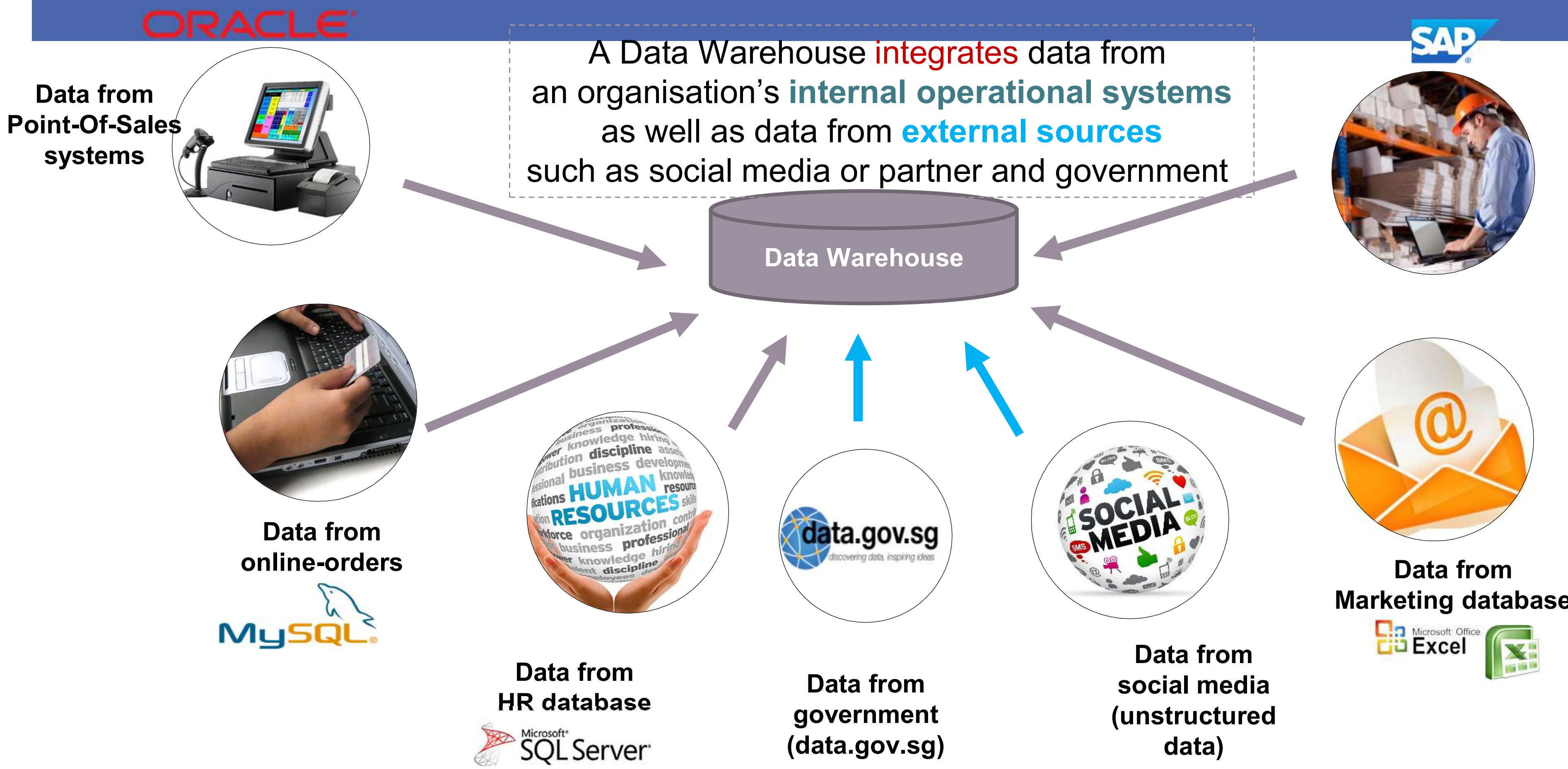
# What Is a Data Warehouse (DW)?

- A very huge database specially designed to enable Business Intelligence activities
- Data stored in the DW is uploaded from internal operational systems such as sales, marketing etc and/or external sources like social media, government data,weather data etc
- DW keeps historical data which can be used to create annual / quarterly trends reports for management



# What is a data warehouse?

A Data Warehouse integrates data from an organisation's **internal operational systems** as well as data from **external sources** such as social media or partner and government



# Why do we nee a data warehouse? – The Problem

- In this information age, many companies use IT systems to store their valuable company data
- Many companies do not use one single IT system to keep the data, but often use disparate systems that store data in vastly different ways
- E.g. ordering system using Oracle database, marketing database using SQL Server etc

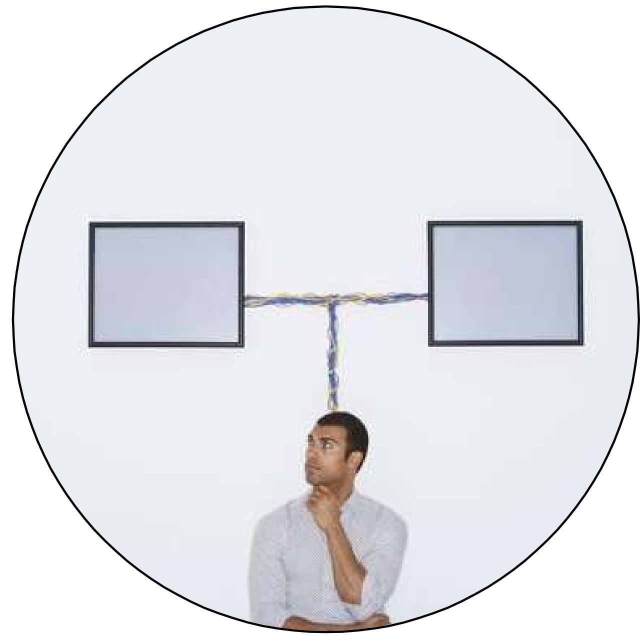
Data from
Point-Of-Sales
systems

Data from
Inventory system

Data from
Marketing database

# Why do we nee a data warehouse? - Good decisions need data

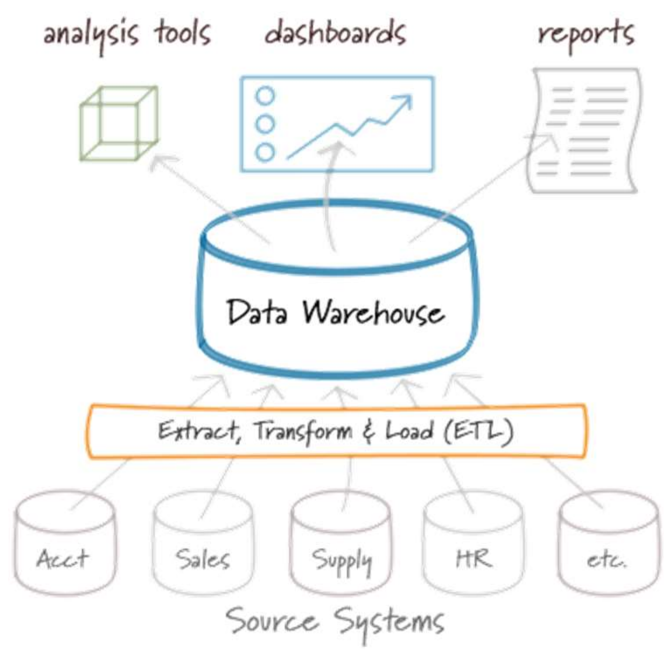
- To stay competitive, companies need to be able to make quick and well-informed decisions
- To do so, companies need easy and speedy access to relevant data that are stored in their various operational systems
- As such, instead of relying on real data and facts based on historical trends to drive strategic direction, companies end up making decisions based on experience, limited and sometimes outdated information



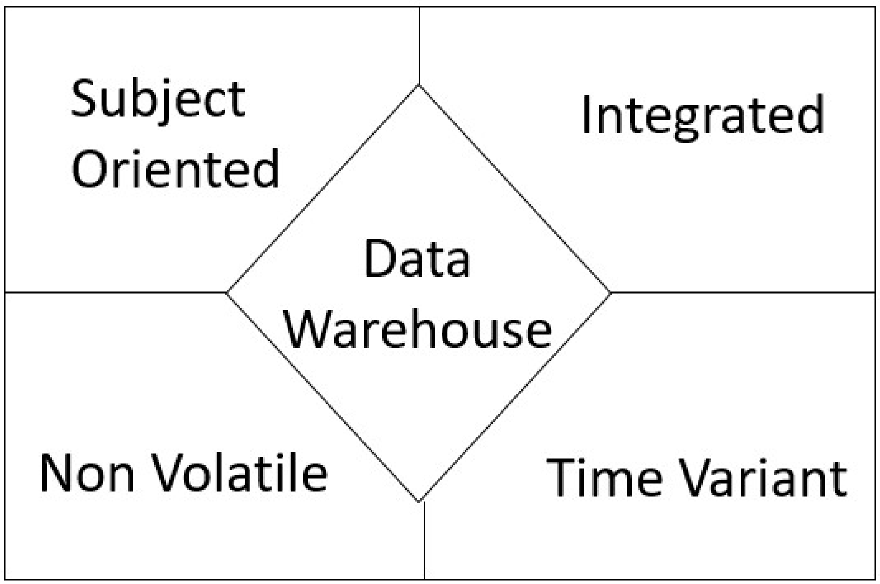
Companies need
to make good decisions
to stay competitive

# Why do we nee a data warehouse? - The Solution

- To solve this problem of 'unreachable' fragmented data', more and more companies have turned to the use of data warehouses
- By building a data warehouse, a company can integrate all its most important data residing in different systems to a central location – with a SINGLE VERSION OF TRUTH
- After the data warehouse has been built, the company can then use BI tools to access the data to help them make **the best strategic decisions**



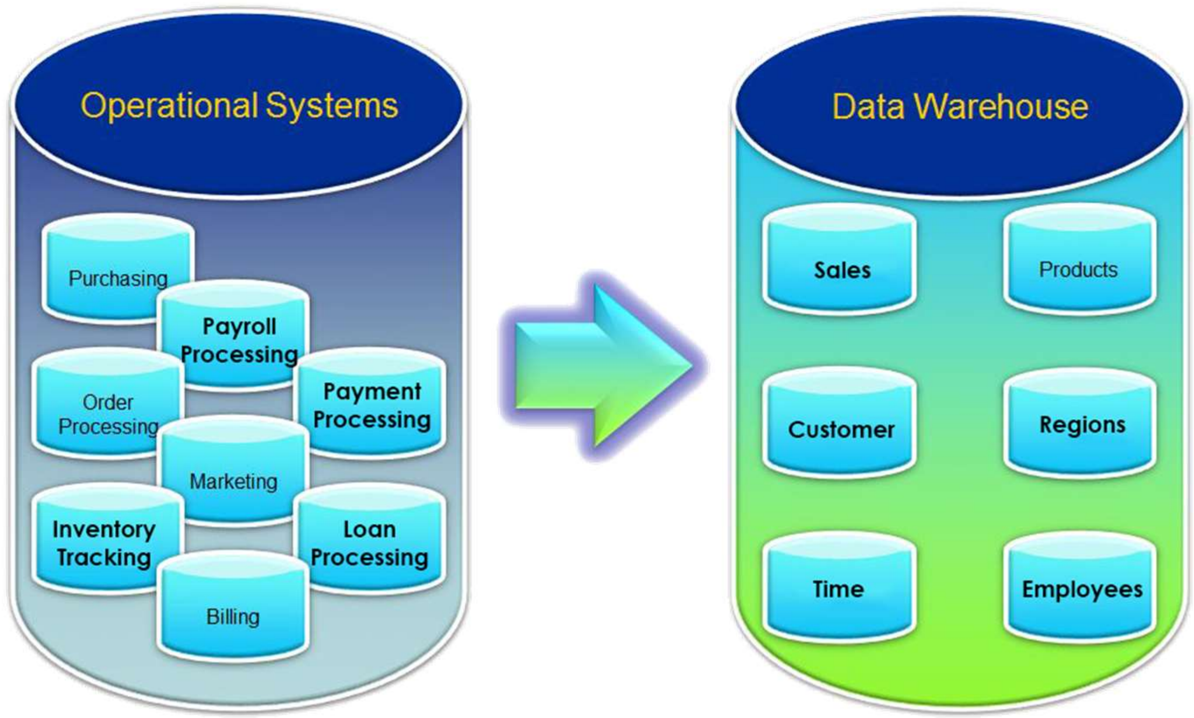
# Characteristics of a data warehouse




A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process

# Characteristics of a data warehouse – Subject Oriented

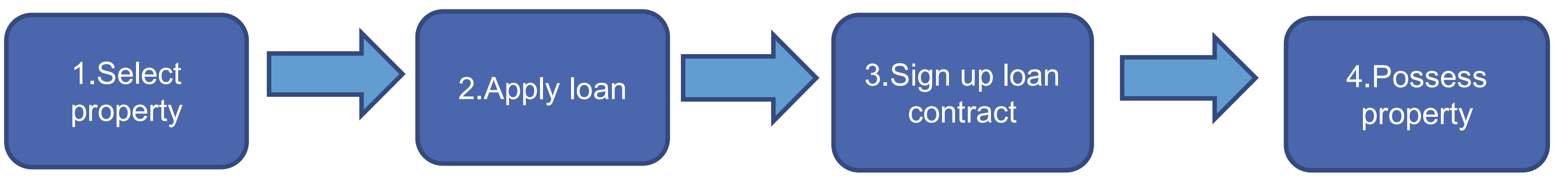
- Operational systems organise their data to cater to the process
- Data warehouse organize the data based on the subject



# Characteristics of a data warehouse – Process vs Subject

Let take an example of a process-oriented situation.

- Assume we are talking about people buying properties. The process is simplified as follows:



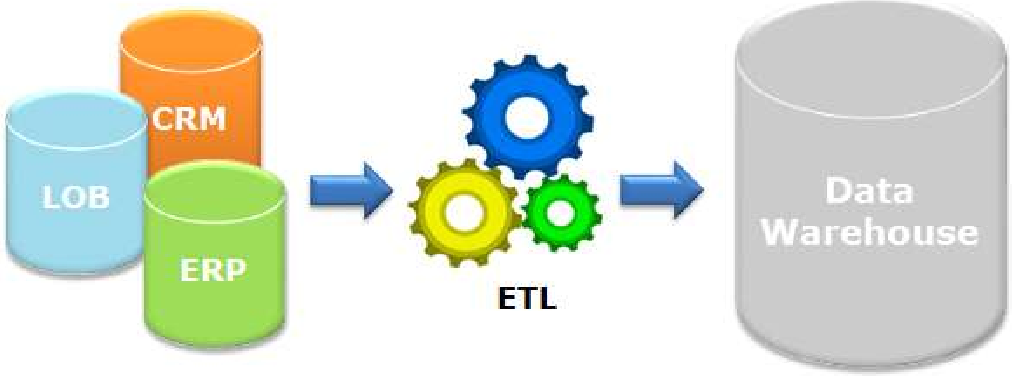

- Process 1 – Buyers, Property Agents, Property Agencies, Developers
- Process 2 – Bankers, Banks, Loans,
- Process 3 – Lawyers, Law Firms, Banks, Buyers
- Process 4 – Buyers, Developers, Building Control Authority, Income Tax Authority
- Each process has many applications to support the process. All the data from the processes are collected into a database designed for transactional purposes.

# Characteristics of a data warehouse – Process vs Subject

- In DW, we are concerned with the data surrounding the subjects.
- From the previous slide, we have the following subjects
  - Buyer
  - Banks, Bankers, Loans
  - Property Agencies, Property Agents
  - Developer, Property
  - Law Firms, Lawyers, Contracts
  - Building Control Authority
  - Income Tax Authority
- In DW, we need to collect the data pertaining to each of these subjects. The process is not important in DW.

# Characteristics of a data warehouse - Integrated

Constructed by integrating data from multiple heterogenous sources: Relational databases, flat files, on-line transactional records.

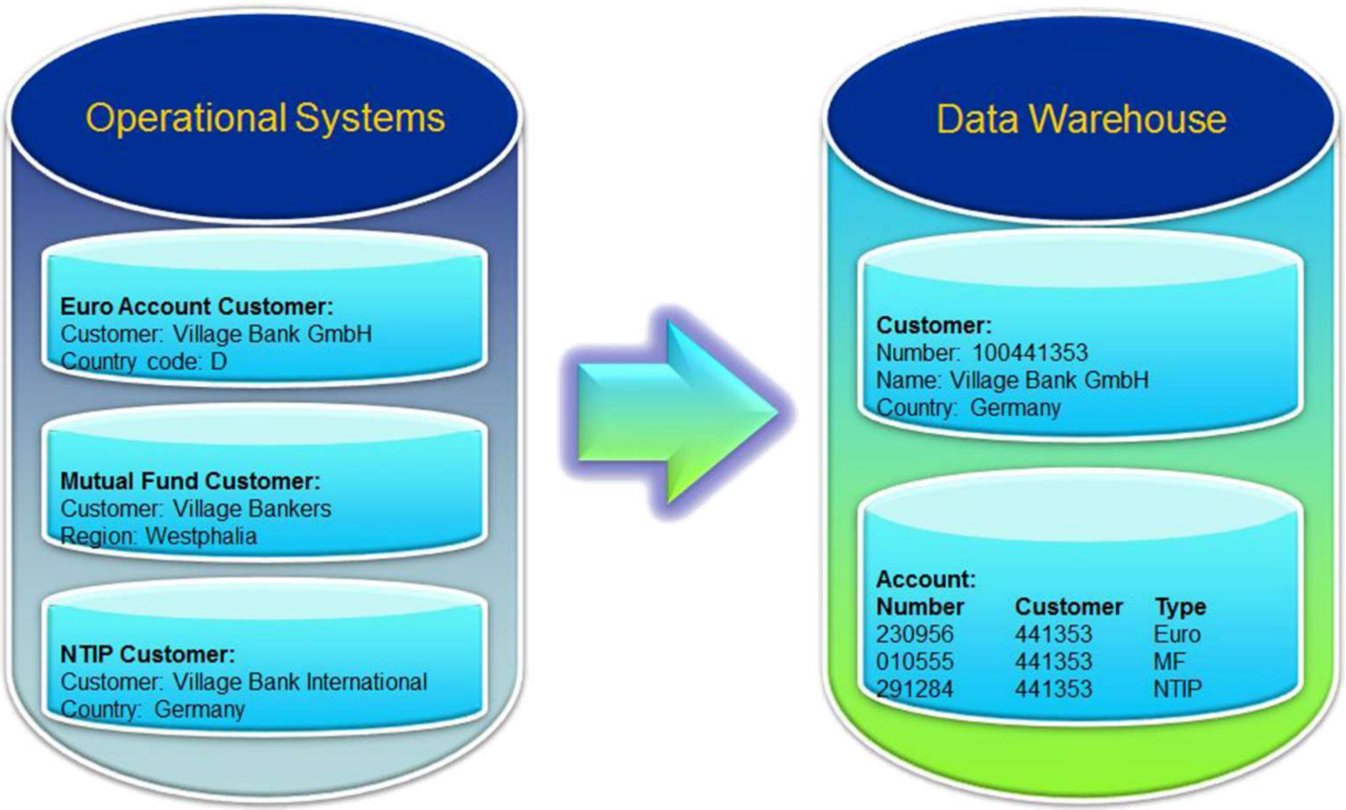



Data cleansing and data integration techniques are applied before data is stored in the warehouse.

- Ensure consistency in naming conventions, encoding among different data sources e.g. always “Singapore” instead of “SG“, “S’pore”

# Characteristics of a data warehouse - Integrated

Integration of data within a warehouse is accomplished by making the data consistent in format, naming and other aspects



# Characteristics of a data warehouse – Time Variant

- A data warehouse keeps historical data
- For example, one can retrieve data from 3 months, 6 months, 12 months, or even older data from a DW
- This contrasts with a transactions system, where often only the most recent data is kept
- For example, a transaction system may hold the most recent address of a customer, but a data warehouse can hold all addresses associated with a customer.

**OPERATIONAL**

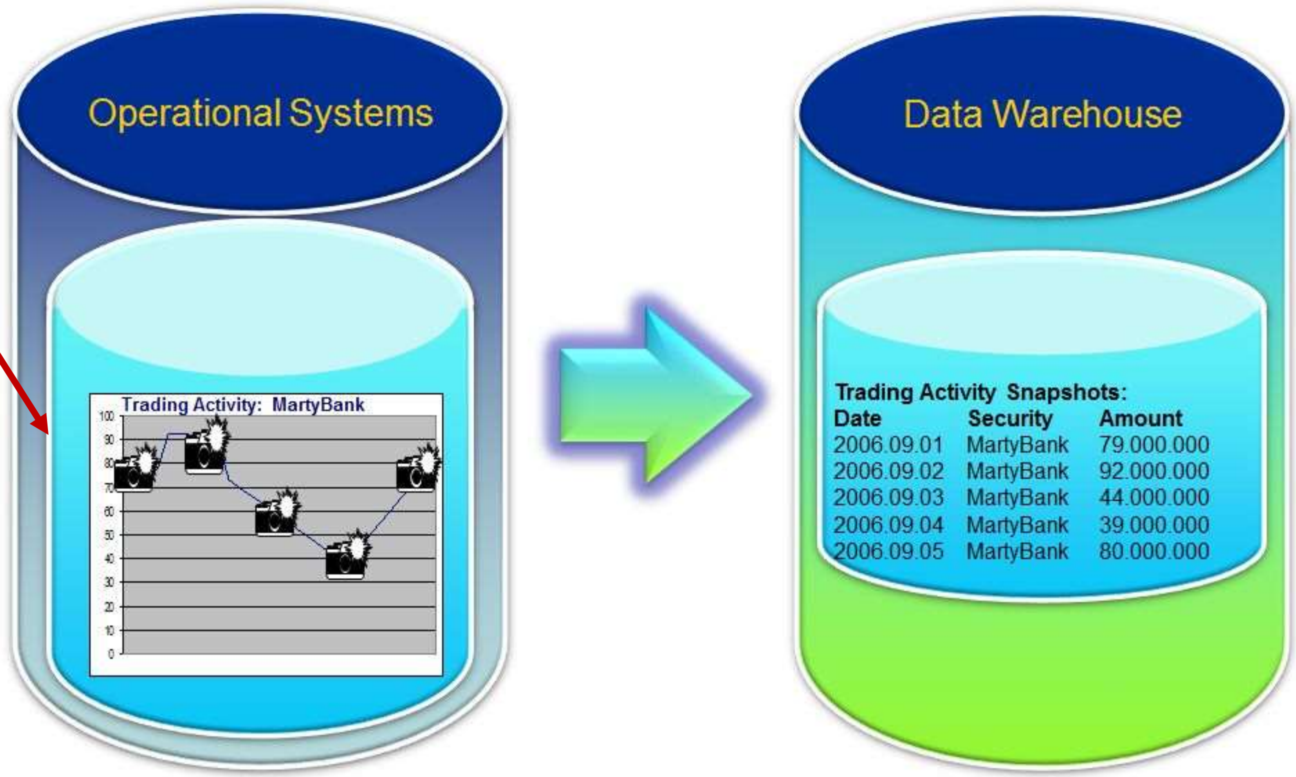
Time horizon: 6 months – 1 year
Records are updated
Not compulsory for key structure to contain an element of time

**DATA WAREHOUSE**

Time horizon: 5-10 years
Sophisticated snapshots of data
Key structure contains an element of time

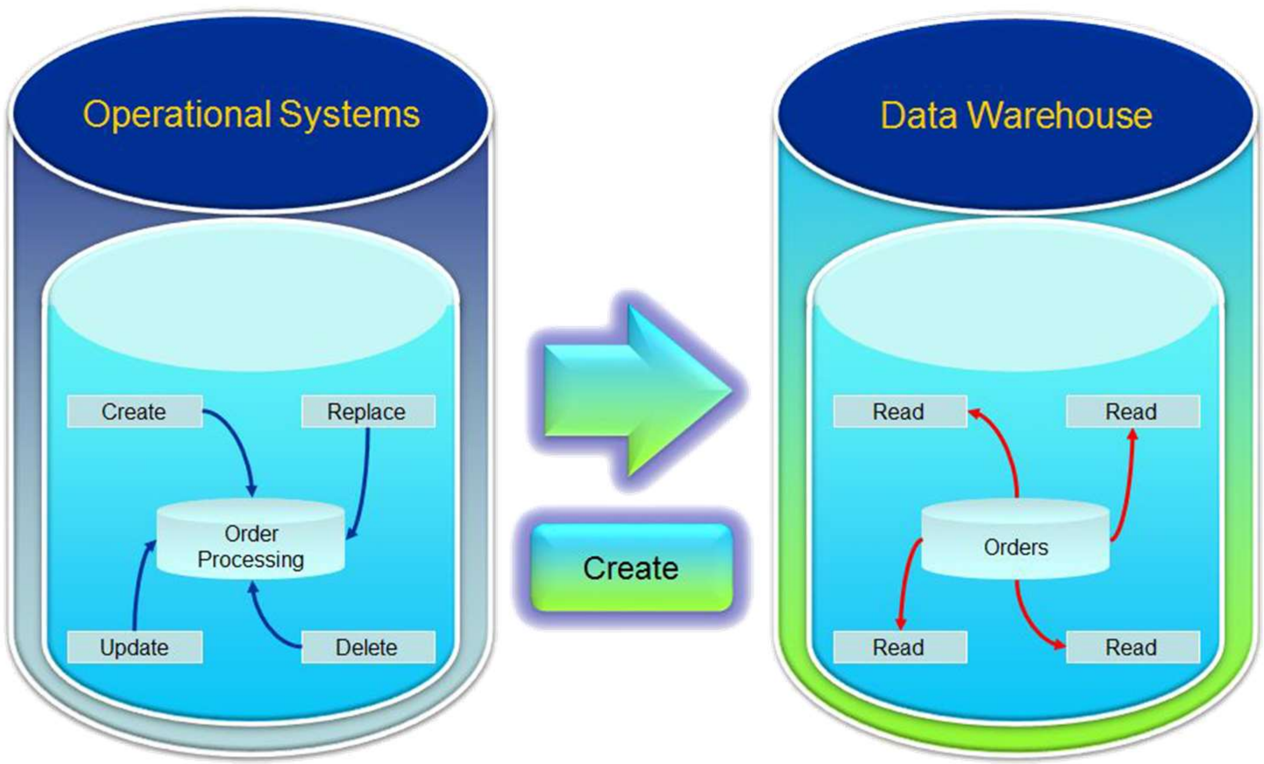
# Characteristics of a data warehouse – Time Variant

- "Snapshots" of operational data are taken at different times regularly (e.g. once a day) and stored in the DW
- Hence, company can retrieve "time-variant" data

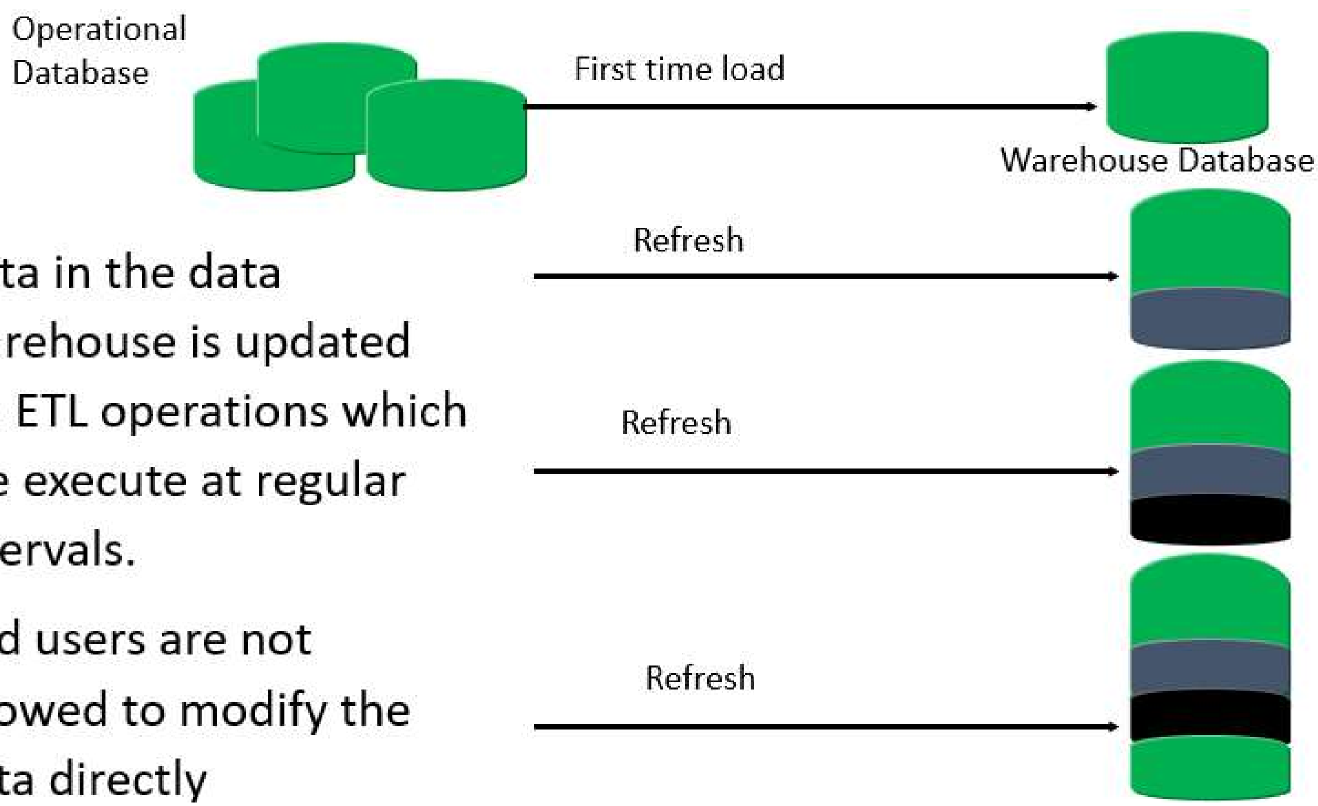


# Characteristics of a data warehouse – Non Volatile

- Non-volatility means that after the data warehouse is loaded there are no changes, inserts, or deletes performed against the informational database.



# Characteristics of a data warehouse – Non Volatile




- Data in the data warehouse is updated via ETL operations which are execute at regular intervals.
- End users are not allowed to modify the data directly

# Dimensional Model

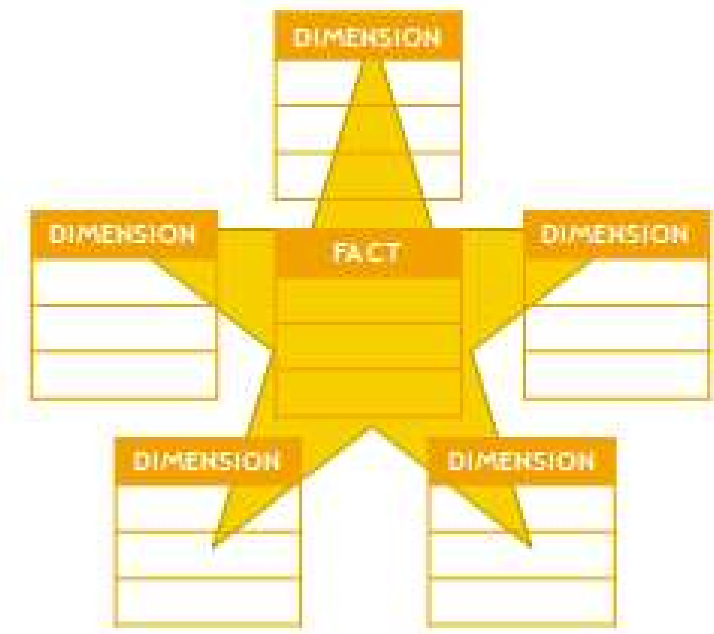
- Dimension modelling is a database design technique that used to support Online Analytical Processing (OLAP) IT systems.
- OLAP systems are IT software that companies use to ask questions about the data related to their business.
- E.g. Business Intelligence systems that churn out monthly sales reports for the management of a hotel.
- Such systems require that the query operations (SELECT) execute at high speed.
- Unlike OLTP systems, OLAP systems DO NOT support transactional operations (add/update/delete) as they are non-volatile.
- To achieve high speed of query operations, the database of an OLAP system should reduce the need for tables to be joined as that will slow down the speed of access

# Dimensional Model

- To reduce the need for tables to be joined, records with repetitive data is kept in the same table as this will eliminate the need to use JOINS to retrieve the necessary data
- The main feature of the Dimensional modelling technique is that the tables are denormalized
- The advantage of this technique is that the IT system is able to perform query operations very fast and the design is more intuitive to the business user
- The disadvantage of this technique is that it produces a lot of redundant data

# Dimensional Model

- Dimensional Modelling is a logical design technique to present data in a framework that is intuitive and allows for high performance access.
- It uses relational model but with some modifications.
- Every DM consists of a table with a multi-part key called the Fact table and a set of smaller tables called Dimension tables.



# Dimensional Model - Components of a Star Schema

A. **Fact**

A numeric value that measures a data event such as how much was spent on a sales transaction

**Fact table**

A central table in a data warehouse that contains several of facts

B. **Dimension**

A collection of information about a business entity such as a customer / product

For example, a customer dimension's attributes could include first and last name, birth date

**Dimension table**

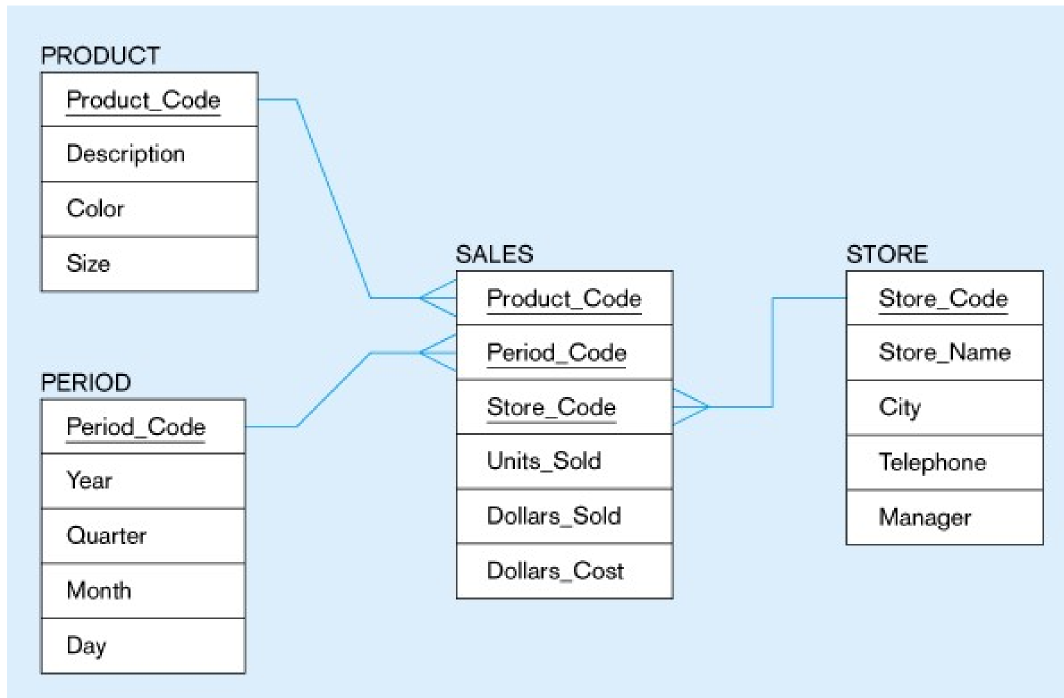
A table in a data warehouse / mart that stores the data of a dimension e.g. Customer dimension table, Product Dimension table

D

# Dimensional Model - Fact

Fact table has the following characteristics:

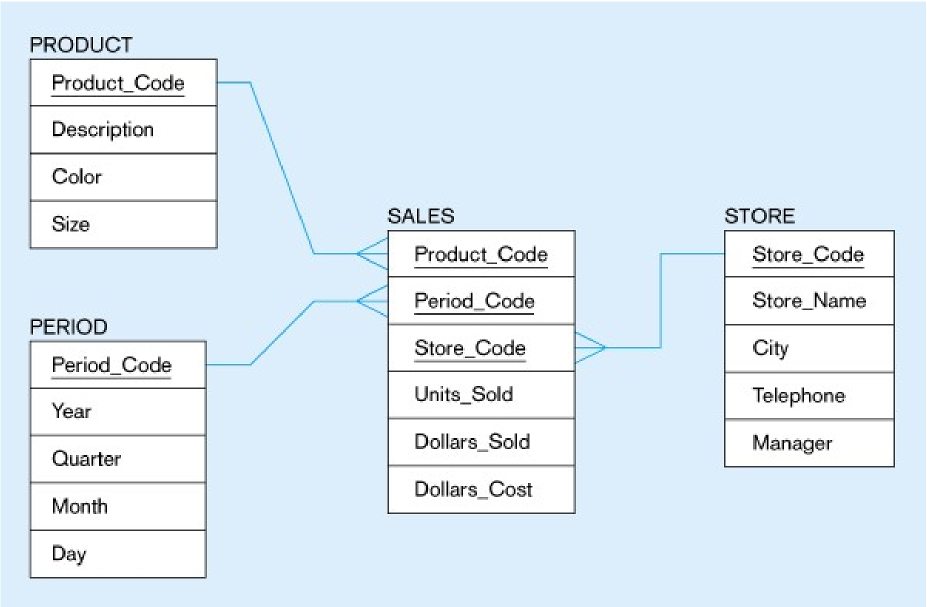
- has a many-to-one relationship to the dimension table
- primary key is multi-part
- has many more rows than those in the dimension tables
- contains primarily numeric data (a.k.a measures) and very few character data



# Dimensional Model - Dimensions

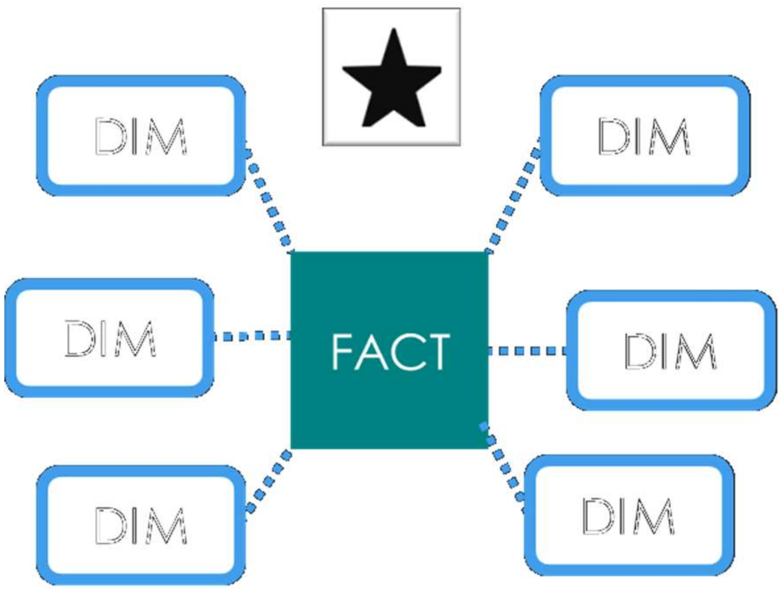
Dimension tables has the following characteristics:

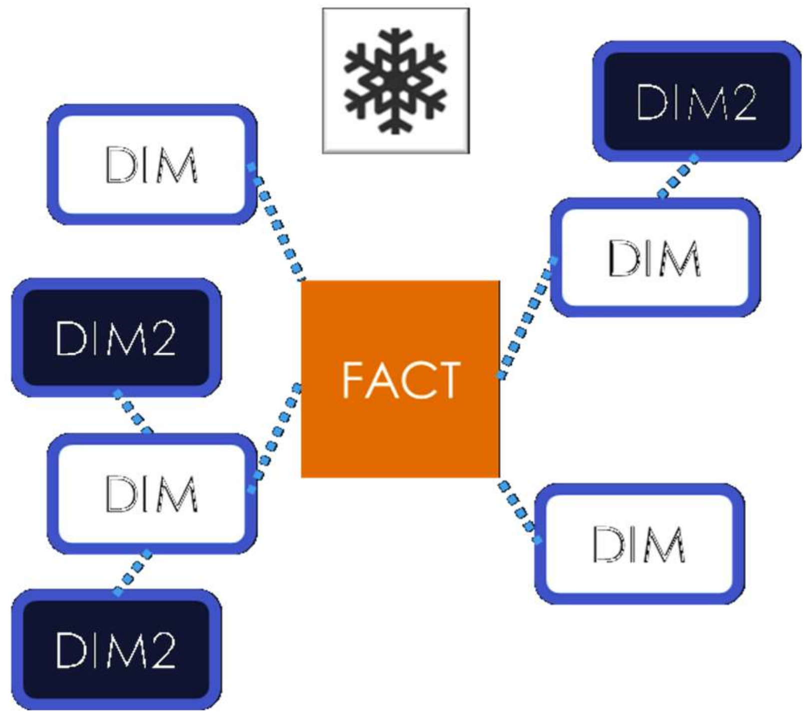
- has one-to-many relationship to fact table
- primary key is single-part
- fewer rows than fact tables
- contains primarily character data



# Dimensional Model – Star and Snowflake Schemas

A dimensional model may be based on star schema or a snowflake schema



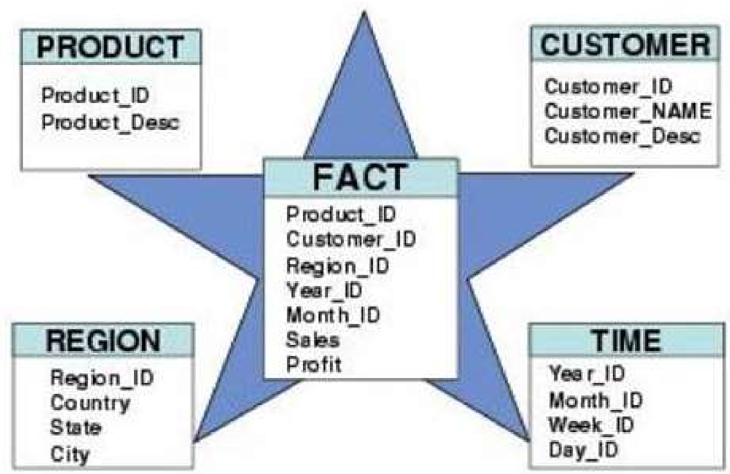



# Dimensional Model - Components of a Star Schema

- A **star schema** is characterized as having a fact table in the centre and is surrounded by many dimension tables that contain de-normalized description of the facts

## Benefits of a star schema

- A star schema is a highly de-normalized set of tables, means
- query results come back faster as there would be less JOIN statements in SQL



# Dimensional Model - Characteristics of Snowflake Schema

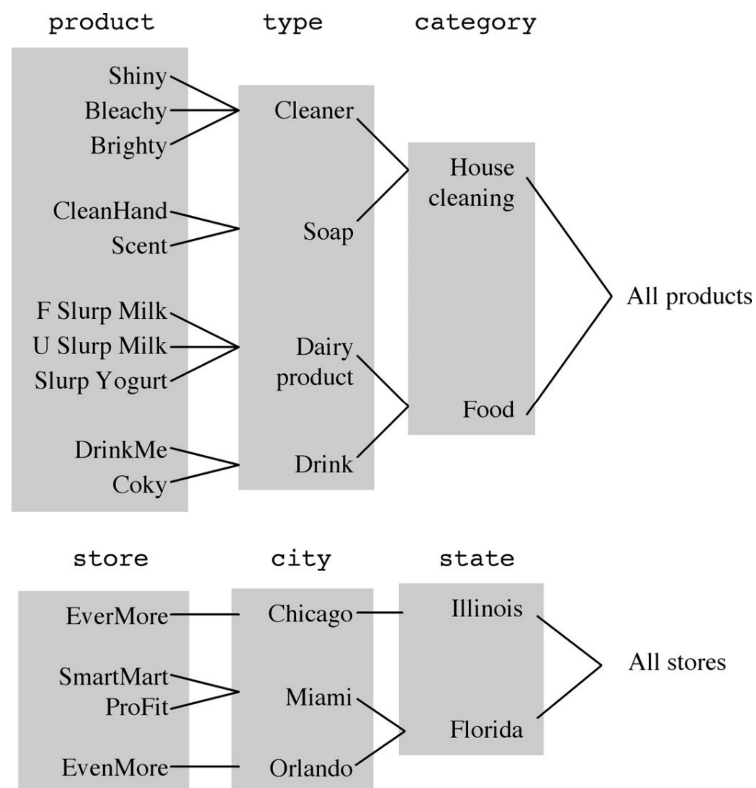
- Result of either business environment or third-party application or design which cannot be implemented by a star schema

- Dimension tables are of two types: primary and secondary
- Primary dimension tables are joined to the fact table
- Secondary dimension tables are joined to the primary dimension tables
- It can increase query performance if the users are accessing data stored in the primary dimension tables 80% of the time
- Snowflake schema may need a number of joins to apply a specific textual constraint to a query

D

# Dimensional Model - Dimension Hierarchies

- A useful feature you can build into a Star or Snowflake dimensional model is the "Hierarchy" component
- A hierarchy is a logical structure that uses ordered levels to aggregate data (consolidate related data under one group)
- E.g. in a Time dimension, a hierarchy can be used to aggregate data from the Month level to the Quarter level, from the Quarter level to the Year level
- E.g. In a Product dimension, a hierarchy might be used to aggregate data from the Model level to the Brand level, from the Brand level to the Category level

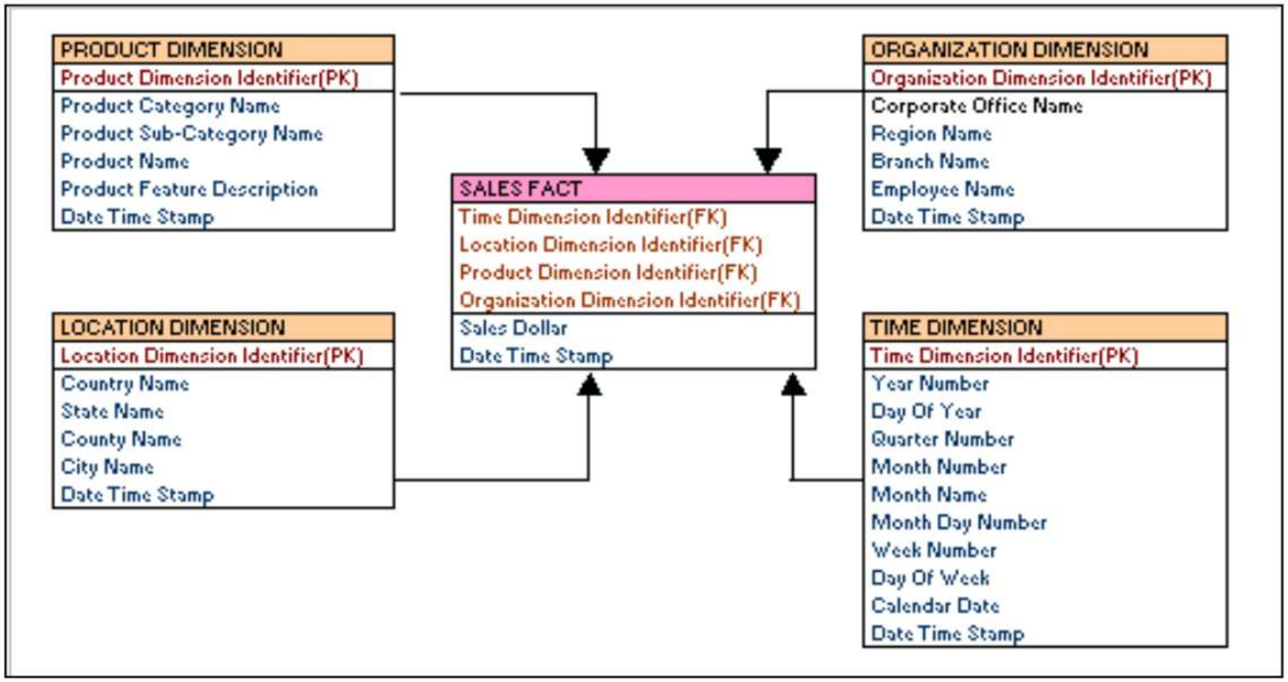


# Dimensional Model - Steps in designing a Star Schema

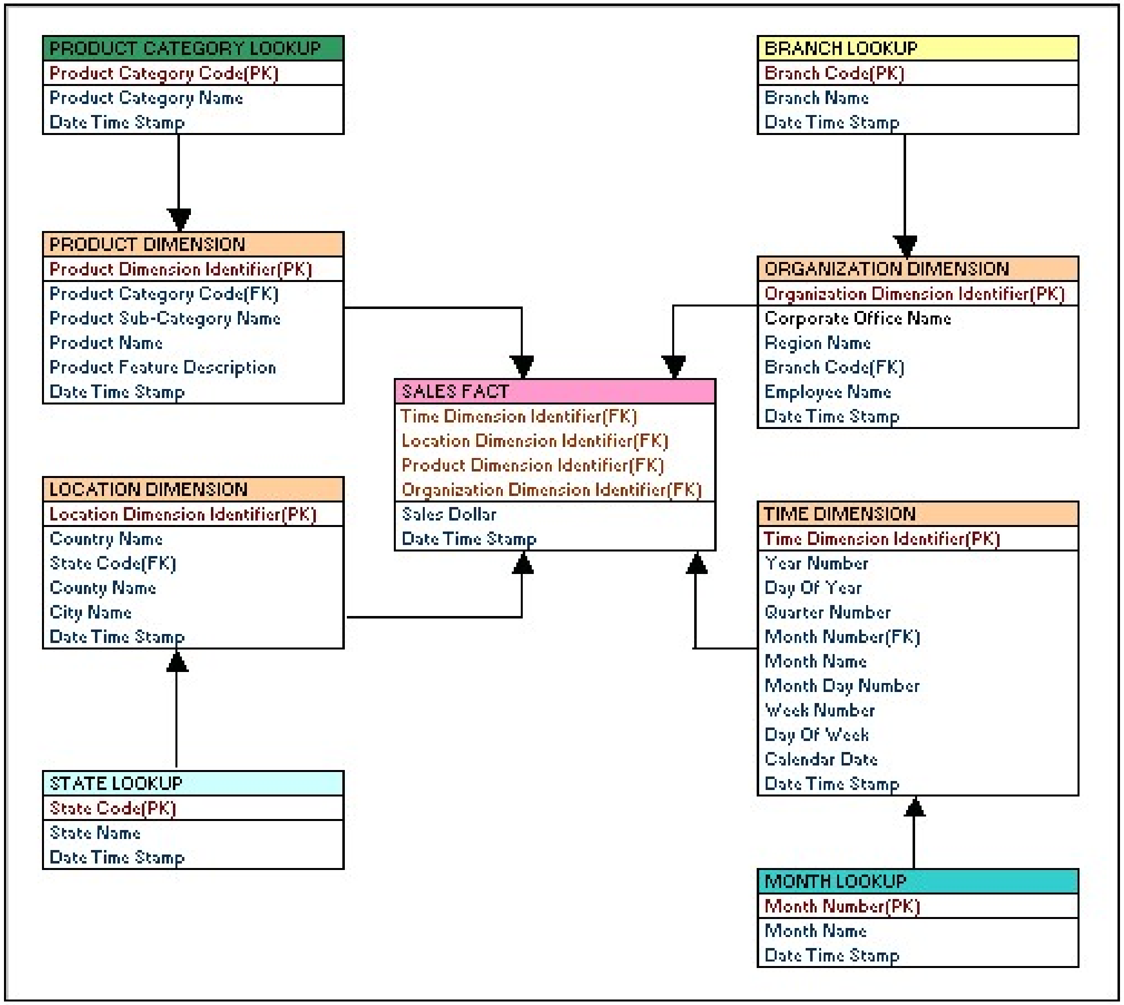
- Identify a business process for analysis (likes sales).
- Identify measures or facts (sales dollar).
- Identify dimensions for facts (product dimension, location dimension, time dimension, organization dimensions).
- List the columns that describe each dimensions (region name, branch name)
- Determine the lowest level of summary in a fact.

# Dimensional Model: Examples

## Example of Star Schema:



# Example of Snowflake Schema:



# Dimensional Model: Star Schema & Snowflake Schema

- In a star schema every dimension will have a primary key.
- In a star schema, a dimension table will not have any parent table.
- Whereas in a snowflake schema, a dimension table will have one or more parent tables.
- Hierarchy for dimensions are stored in the dimension table itself in a star schema.
- Whereas hierarchies are broken into separate tables in snowflake schema.

# Dimensional Model - Surrogate Keys Vs Natural keys

- **Surrogate keys**. A system-generated attribute is added to each entity type table and made the primary key. This attribute is often an artificial number. The primary key for each relationship table consists of identifiers from the related entity types.
- **Natural keys**. A unique combination of application attributes is used to identify each entity. The primary key for each relationship table consists of primary keys from the related entity types.
  - The natural keys can be used as candidate keys in the Dimensions to facilitate the ETL process.

# Dimensional Model - Why use Surrogate Keys

- According to Ralph Kimball:
  - Surrogate keys " One of the primary benefits of surrogate keys is that they buffer the data warehouse environment for operational changes"
  - Example, imagine you have used the Product code as key and the operation system re-uses product code. What do you now do with the rest of the old data?

- **Natural keys (NK)** or **Business keys** are generally alphanumeric values that is not suitable for index as traversing become slower. For example, prod123, prod231 and etc.
  - Business keys are often reused after sometime. It will cause the problem as in data warehouse we maintain historic data as well as current data.
  - For example, product codes can be revised and reused after few years. It will become difficult to differentiate current products and historic products. To avoid such a situation, surrogate keys are used.

# Dimensional Model – Slowing Changing Dimension

- Dimensions that change over time are called slowing changing dimensions.
- Most of the fundamental measurements we store in our fact tables are time series, which we carefully annotate with time stamps and foreign keys connecting to calendar date dimensions. But the effects of time are not isolated just to these activity-based time stamps.
- All other dimensions that connect to fact tables, including fundamental entities such as Customer, Product, Service, Terms, Location and Employee, are also affected by the passage of time.

# Dimension Model – Slowing Changing Dimension

- Sometimes the revised description merely corrects an error in the data.
- But many times the revised description represents a true change at a point in time of the description of a dimension member, such as Customer or Product.
- Because these changes arrive unexpectedly, sporadically and far less frequently than fact table measurements, we call this topic slowly changing dimensions (SCDs).

# Dimension Model – Slowing Changing Dimension

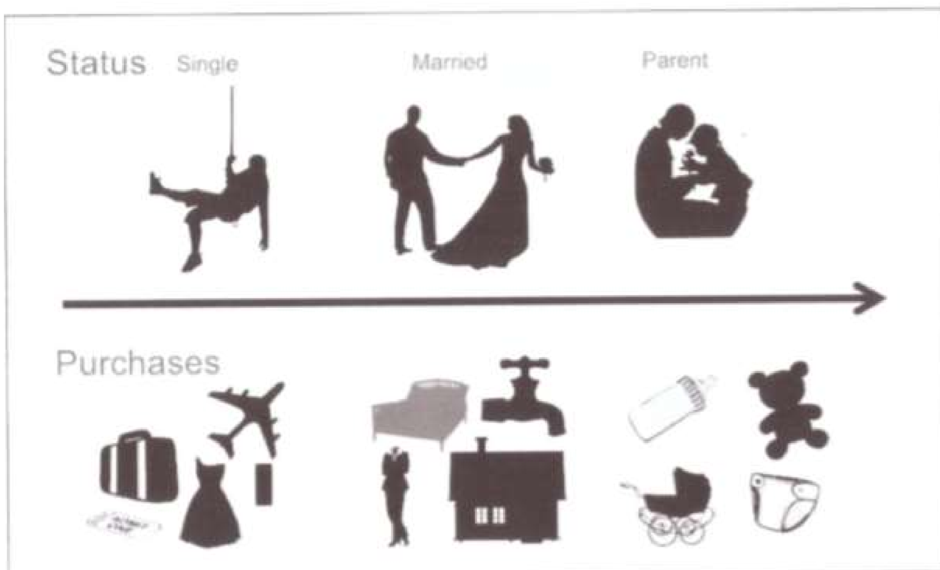



*Figure 2-4. Shopping data with slowly changing dimensions*

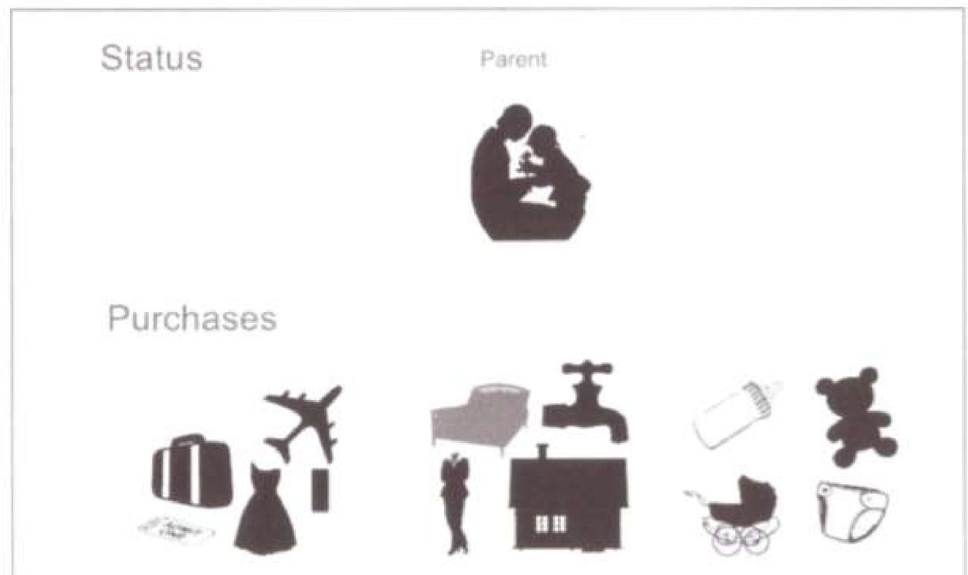



*Figure 2-5. Shopping data without slowly changing dimensions*

Source: Alex Gorelik, The Enterprise Data Lake, O'Reilly, 2019

# Dimension Model – Slowly Changing Dimension

- **Type 0** - The passive method
- **Type 1** - Overwriting the old value
- **Type 2** - Creating a new additional record
- **Type 3** - Adding a new column
- **Type 4** - Using historical table
- **Type 5** - Combine approaches of types 1,2,3

# Dimension Model – Slowly Changing Dimension

**Type 0** - The passive method. In this method no special action is performed upon dimensional changes. Some dimension data can remain the same as it was first time inserted, others may be overwritten.

**Type 1** - Overwriting the old value. In this method no history of dimension changes is kept in the database. The old dimension value is simply overwritten be the new one. This type is easy to maintain and is often use for data which changes are caused by processing corrections(e.g. removal special characters, correcting spelling errors).

Before the change:

| Customer_ID | Customer_Name | Customer_Type |
|---|---|---|
| 1 | Cust_1 | Corporate |

After the change:

| Customer_ID | Customer_Name | Customer_Type |
|---|---|---|
| 1 | Cust_1 | Retail |

D

# Dimension Model – Slowly Changing Dimension

**Type 2** - Creating a new additional record.

In this methodology all history of dimension changes is kept in the database. You capture attribute change by adding a new row with a new surrogate key to the dimension table.

Both the prior and new rows contain as attributes the natural key(or other durable identifier). Also 'effective date' and 'current indicator' columns are used in this method. There could be only one record with current indicator set to 'Y'. For 'effective date' columns, i.e. start_date and end_date, the end_date for current record usually is set to value 9999-12-31.

Before the change:

| Customer_ID | Customer_Name | Customer_Type | Start_Date | End_Date | Current_Flag |
|---|---|---|---|---|---|
| 1 | Cust_1 | Corporate | 22-07-2010 | 31-12-9999 | Y |

After the change:

| Customer_ID | Customer_Name | Customer_Type | Start_Date | End_Date | Current_Flag |
|---|---|---|---|---|---|
| 1 | Cust_1 | Corporate | 22-07-2010 | 17-05-2012 | N |
| 2 | Cust_1 | Retail | 18-05-2012 | 31-12-9999 | Y |

# Dimensional Model – Slowly Changing Dimension

- **Type 3** - Adding a new column.

In this type usually only the current and previous value of dimension is kept in the database. The new value is loaded into 'current/new' column and the old one into 'old/previous' column.

Generally the history is limited to the number of column created for storing historical data. This is the least commonly needed technique.

Before the change:

| Customer_ID | Customer_Name | Current_Type | Previous_Type |
|---|---|---|---|
| 1 | Cust_1 | Corporate | Corporate |

After the change:

| Customer_ID | Customer_Name | Current_Type | Previous_Type |
|---|---|---|---|
| 1 | Cust_1 | Retail | Corporate |

D

# Dimensional Model – Slowly Changing Dimension

- **Type 4** - Using historical table.

In this method a separate historical table is used to track all dimension's attribute historical changes for each of the dimension.

The 'main' dimension table keeps only the current data e.g. customer and customer_history tables.

Current table:

| Customer_ID | Customer_Name | Customer_Type |
|---|---|---|
| 1 | Cust_1 | Corporate |

Historical table:

| Customer_ID | Customer_Name | Customer_Type | Start_Date | End_Date |
|---|---|---|---|---|
| 1 | Cust_1 | Retail | 01-01-2010 | 21-07-2010 |
| 1 | Cust_1 | Oher | 22-07-2010 | 17-05-2012 |
| 1 | Cust_1 | Corporate | 18-05-2012 | 31-12-9999 |

# Dimensional Model – Slowly Changing Dimension

- **Type 5** - Combine approaches of types 1,2,3.

In this type we have in dimension table such additional columns as:

current_type - for keeping current value of the attribute. All history records for given item of attribute have the same current value.

historical_type - for keeping historical value of the attribute. All history records for given item of attribute could have different values.

- start_date - for keeping start date of 'effective date' of attribute's history.
- end_date - for keeping end date of 'effective date' of attribute's history.
- current_flag - for keeping information about the most recent record.

| Customer_ID | Customer_Name | Current_Type | Historical_Type | Start_Date | End_Date | Current_Flag |
|---|---|---|---|---|---|---|
| 1 | Cust_1 | Corporate | Retail | 01-01-2010 | 21-07-2010 | N |
| 2 | Cust_1 | Corporate | Other | 22-07-2010 | 17-05-2012 | N |
| 3 | Cust_1 | Corporate | Corporate | 18-05-2012 | 31-12-9999 | Y |

In this method to capture attribute change we add a new record as in type 2. The current_type information is overwritten with the new one as in type 1. We store the history in a historical_column as in type 3.

# Dimension Model – Fast Changing Dimension

- A dimension is a fast changing or rapidly changing dimension if one or more of its attributes in the table changes very fast and in many rows.
- As you know slowly changing dimension type 2 is used to preserve the history for the changes. But the problem with type 2 is, with every change in the dimension attribute, it adds new row to the table.
- If in case there are dimensions that are changing a lot, table become larger and may cause serious performance issues. Hence, use of the type 2 may not be the wise decision to implement the rapidly changing dimensions.

# Dimension Model – Fast Changing Dimension

- Separate Rapidly Changing Attribute by Implementing Junk Dimension
- There may be some attribute that may be changing rapidly and other not. The idea here is to separate the rapidly changing attribute from the slowly changing ones and move those attribute to another table called junk dimension and maintain the slowly changing attribute in same table. In this way, we can handle situation of increasing table size.

D

# Dimension Model – Fast Changing Dimension

| PATIENT |
| --- |
| Patient_id |
| Name |
| Gender |
| Marital_status |
| Weight |
| BMI |

| PATIENT_JNK_DIM |
| --- |
| Pat_SK |
| Weight |
| BMI |

- The attribute like patient_id, Name, Gender, Marital_status will not change or changes very rarely. And attribute like weight and BMI (body mass index) changes every month based on the patient visit to hospital.

- So, we need to separate the weight column out of the patient table otherwise we end up filling the table if we use SCD type 2 on PATIENT dimension. We can put the weight column which is rapidly changing into junk dimension table.

Pat_SK is the surrogate key and acts as a primary key for junk dimension table. Below is how our PATIENT table looks after removing weight and BMI from it.

| PATIENT |
| --- |
| Patient_id |
| Name |
| Gender |
| Marital_status |

# Dimension Model – Fast Changing Dimension

Link Junk Dimension and PATIENT Table

- In this step, we must link the junk dimension and patient table. Keep in mind; we cannot simply refer the junk dimension table by adding its primary key to patient table as foreign key. Because any changes made to junk dimension will have to reflect in the patient table, this obviously increases the data in patient dimension.
- Instead, we create one more table called mini dimension that acts as a bridge between Patient and Junk dimension. We can also add the columns such as start and end date to track the change history.

| PATIENT_MINI_DIM |
| --- |
| Pat_id |
| Pat_SK |
| START_DATE |
| END_DATE |

This table is just bridge between two tables and does not require any surrogate key in it.

Below is the diagrammatic representation of the Rapidly Changing Dimension implementation