# Definition

* Data warehouse
* Data lake
* Data Virtualization

# What is Data Warehouse

* Datawarehouse and Database are not same.
* Datawarehouse is built on top of some type of database or Multidimensional Database called Cube
* Datawarehouse is a use, and database is platform
* Data inside the datawarehouse comes from elsewhere. (operational devices, other sources).
* This mean we are not creating data in Datawarehouse, as part of some transaction- Student registering for a class. We don’t do this in Datawarehouse.
* The more sources, the more complex data warehouse environment
* Data is copied in Datawarehouse, not moved.
* There are rules which govern how we built, Organise and Store in Datawarehouse
* Bill Inmon in 1990 came with rules,
  + **Integrated Environment**, mean various sources are coping there data , it pool data from transactional system.
    - **intigrity means there will be only one version of anything**
  + **Subject Oriented** regardless of how many systems are sending data, and which type of data is being sent. We need to re organize data based on subject
  + **Time Variant**- it should contain historical data and current data
  + **Non-Volatile** – refreshing the data warehouse mean loading new data in system. Now data will remain same in system until it’s not been refreshed.

# Why Data Warehouse

* To Support data driven decision making. Rather than based on experiences, hunches, intuition.
  + We should have data for
    - Past
    - Present
    - Future (Prediction)
    - Understand unknown – using analytics to understand which is not shown in data
  + This analysis is called **Business Intelligence**
* One stop shopping- Data we need is available at single place rather than scattered among transaction /operational system

# Compare Datawarehouse with Data Lake

**Differences**

* Datawarehouse
  + Relational Database are used for
    - Datawarehouse, as Datawarehouse is built on top of relational database
    - For transactional system (inserting daily sales transaction)
  + Multidimensional Database called Cube (it’s a special purpose database not just relational) also helps to build DW.
* Data lake
  + IT has been built on top of Big Data.
  + Big Data Support 3 V
    - Volume
    - Velocity Rapid intake of new and changed Data
    - Verity Big data support
      * Semi Structured data like text messages, blog, emails.
      * Structured datatype like, date datetime, short date, int, char
      * Non-Structured like audio, Video

# Compare Datawarehouse with Data Virtualization

* Extract files- We Extract the Data from various system and then we used to analyse it.
  + - Lots of time used to gather and reorganize data
* DDBMS- Distributed Database management system
  + Like Hadoop – data present in various database, and we maintain an index which helps to redirect the query to various system.
* Data Virtualization – rather than having separated database for our designated data, we will access data directly from there source
* Data Warehouse – keep designated data in one database

Data virtualization should meet below criteria else switch to data warehouse

* + - Relaxation on query running time –
    - Simple transformation
    - Handful of data source

# Compare Data warehouse with DataMart

Lets say we built a DW for a Flipkart, so company may have various department

HR

Account

Employee

Sales

Daily transactional

Stores

Quantity

Now all these data base will be source for DW , from DW we pull data regarding Future sales Prediction , this further processed data will be inserted to data mart

Datamart is action specific databases.

## Datamart of two type

* Dependent – Which gets data from Datawarehouse
  + Uniform data across the data marts
* Independent – It does not need DW instead it takes data from various sources. Same like old fashioned extract files.
  + In DataMart data organized dimensionally, rather than according to reports that needs to be produced
  + Dimensional data made up of facts and dimension.
  + DM is a small scale DW.

# Difference between DW or ODS

* Integration
  + Same as DW, it integrates data from various sources
* Subject oriented
  + ODS focus mainly on Tell me what is happening right now, it pull data based on need for BI
  + DW also pull data based on need, but it helps to decision making and prediction.
* Time variant
  + ODS maintains only current operational data
  + Real time data feed from source system
  + Batch oriented Data feed in DW
* Non-Volatile

ODS are now less popular as we have faster DW, with lack of latency

Superseded by Big Data Velocity

# Data warehouse environments

Data source 🡪 ETL 🡪Datawarehouse🡪 Data marts

Datamarts has subsets of data from Datawarehouse for different set of users

**BI+ (DW/DL/DV/ODS) = Data analytics**

# Architecture of Datawarehouse

* **Centralize data warehouse**-Having a single database to support BI
* **Data Marts** – small scale DW
* **Component based warehouse-**Multiple components like DW and data marts work together. And there are many DW and DM.
* **Cubes-** Special type of Database
* **Operational data Store**
* Layers
  + Staging layer
    - Persistent (Keep older data)
    - Non – Persistent (Delete Older data)

## Centralize data warehouse

* Only one database is needed, which fetch data from multiple sources
* One stop shopping, users uses single database for task like BI , report creation

**Challenges**

Technology – RDB face challenges with Data Volume

Design Engineering and data model

Internal Organization, Internal personal challenges

## Understand your Pick

* Data Warehouse
  + Centralize  
    Default Option   
    One Stop Shopping  
    Modern Technology  
    *It requires High Degree of Inter Organizational Co-operation (no body wants to share data)  
    High data Governance  
    Ripple Effect – small change will affect other environment.*
    - EDW  
      Enterprise Data warehouse
      * Relational
      * Special Database (Cube)
    - Data Lake  
      Here we can use technology like Hadoop ,
      * Hadoop
      * Special Database (Cube)
      * Other (AWS S3, Google)
  + Component-Based   
    Decomposition- Divide Data into multiple components, we can isolate portion of data. This will helps to make some Datawarehouse private.  
    Mix and Match Decomposition- we can set up new DW with latest technology , we can use new BI technologies on Newly set up DW.  
    Bolt Together Component  
    Overcome Org challenges (data Privacy, create a DW and use its data for building another DW)  
    *Often Inconsistent Data*

*Difficult to Cross integrate the components .its difficult to have data mart from multiple Datawarehouse, as data will be inconsistent*

* + - Architected
      * DW+DM
        + Dependent DM (CIF) corporation integration factory(how has access to which part of data warehouse /Data mart)
        + Front End DM- Data will be moving from DM to DW
      * DM Only
        + DW Bus
    - Non-Architected
      * Federated EDW- all DataMart are part of a Datawarehouse

## Including Multidimensional database or Cube in DW

Just like RDBMS is a source for DW, Cube can also be a data source.

## What is a cube?

* It is not a RDBMS
* It follows the idea of dimensionality
* Alternative of DW, here we have facts and for context we have dimensionality.
* IT possible we can have DW built on top of RDBMS but data mart are built on top of cube.
* Or we can have mix and match, one DM is on RDBMS and other are on cube

## Advantage

* Fast query response time
* Modest data volumes
* More vendor variation than RDBMS

## Disadvantage

Less flexible than RDBMS

## Role of Staging Layer

1. Staging layer is a mirror copy of source system.
2. we copy all columns with same data type from there sources to their respective staging table
3. primary focus is on extract and we use it Before user access layer
   1. User access layer.
      1. It’s a DW or DM.
      2. all data modelling technique like star schema , snowflake schema , fact , dimension , fact table , dimension table, all work on user access layer.
   2. Source-- Extract 🡪 Staging-- Transform 🡪 DW (User access layer)
   3. Why we don’t transform the data in staging table directly
      1. If we have same software in source we can do it else we have to do one to one mapping between source and staging.
      2. If source is sql server / oracle we should have many to one.
4. Two type of staging layer
   1. Persistent
      1. We add new data on top of existing data in staging table
   2. Non-Persistent
      1. We delete the existing data from staging table before loading the new data.
      2. Advantage
         1. Less storage space required
      3. Disadvantage
         1. Need to go back to the source system if rebuild of user access layer required.
         2. Data QA required source system

## Introduction to ETL

### Compare ETL /ELT

* 1. Extract, transform, load
     1. Data move from source---extract🡪 staging--Transform🡪DW(user access layer)🡪 load in DW
     2. Challenges
        1. Significant business analysis before adding the data in DW. This will need computing power in DW server
        2. Significant data modelling before loading the data
  2. Extract load, transform
     1. In data lake or big data
        1. Source—Extract 🡪 load (Data lake) 🡪 Transform ( DM or cube)
     2. Advantage
        1. We can use computing for of our analytics server
        2. Schema on read / Schema on write
           1. **Schema on write** – Schema is already set before doing the any operation.

**Set table**

**Set primary key/foreign key**

**Create schema**

Do operation like insert, select

* + - * 1. Schema on read-Create the schema when reading the data.

It helps to store unstructured data.

Here we just need to store the data in database.

Later on you can read what ever you want to extract from this unstructured data.

* + - 1. We can defer creation of schema by loading the data in database/data lake/aws s3 bucket

1. So if we are using RDBMS Cube to build DW then we can use ETL only , incase we use BIGDATA technologies then we can use ELT also.

### Variation of ETL

* 1. Initial ETL
     1. Do it one time only, at first time set up
     2. You bring only relevant data
     3. Bring probably needed data for BI and analytics
     4. Bring Historical data
     5. Only big data technologies bring all data
  2. Incremental ETL
     1. Do it in regular bases, incrementally refreshes DW
     2. Bring new data
     3. Bring modified data
     4. You never delete the data, keep it for historical purpose. We will highlight that this data is no longer relevant
     5. ETL Pattern
        1. Append
        2. In place Update.
           1. You are not appending; you are going to existing data and update it.
           2. In dimension modelling, we use in place update for type 1 Slowly changing Dimension.
        3. Complete replacement
           1. Overwrite a portion of data
        4. Rolling append
           1. Maintain a certain period of data, if you roil in the new data, you will delete the older data beyond some time limit.

### Explore the Role of Data Transmission

* 1. Data Transmission Goal
     1. Uniformity – keep data uniform when reading from new source and keeping in existing DW
     2. Restructuring- keep engineered data
  2. Common Transformation model (Two or more source, data will be extracted to single staging table)
     1. Data Value Unification
        1. Transform values of column to keep it same , if it exist in different format in source DB. Ex. Abbreviation in DB1, complete name in DB2
     2. Data type and size unification
        1. String (20) in DB1, String (40) in DB2
     3. De- Duplication
        1. Verify in existing DW , if data is already copied from DB1. So check it based on some KEY.
     4. Dropping column (Vertical slicing)
        1. Copy only relevant column
     5. Value – based row filtering ( Horizontal slicing )
        1. Filter the ROW . lets say we want to register only those student who have got 80% in 12th
     6. Correcting Known Error
        1. If we know some error is coming in source data , then we can go and correct it using transformation , lets say if phone no contain + sign then remove it.

# Design Engineering

* 1. How do we use Data warehouse?
     1. Making data – driven decision om
        1. Past
        2. Present
        3. Future
        4. The Unknown
  2. BI Category Data model

Basic Reporting Dimensional

OLAP Dimensional

Predictive Analytics Data Mining /Speceialize

Exploratory Analytics Data mining/Specialize

* 1. To make Decision we need
     1. Measurement (Fact)
     2. Context (Dimension)
        1. For context for specific thing
        2. By – Slicing the information using By

Provide Sales information for march month by store, and item category .

* 1. Fact
     1. Something which we can measure
     2. We don’t not use logical fact here like sun rise from east
     3. Question?
        1. Which fact should go to same fact table
        2. Different type of fact table
        3. How to connect fact table and dimensional tables

1. Type of Fact
   1. Additive
      1. Which can be add in any situation .these fact like amount , marks.
   2. Semi Additive
      1. Cumulative sum,
      2. Some time you can add them , some time you can’t
      3. These are not transaction but something which already exist and we measure it , like temperature, water level, humidity etc
   3. Non additive
      1. GPA like 3.3
      2. Grade
      3. Margin
      4. Ratio n
      5. Percentage
      6. You can calculate aggregative average like lets sat student grade in each class , 8, 5,6, 7, 8, 3, 5, 6
         1. You can store average of this grade and keep it
         2. 8/total sum , 5 /total sum, etc
   4. Dimension
      1. Context for a fact
      2. Product
      3. Product family
      4. Product category

Technically we have one hierarchy and 3 level

* 1. Modelling technique
     1. Star Schema
        1. Here data from all the level of one hierarchy will go into same table
        2. Other table will have other hierarchy.
        3. Only one level away from fact table.
        4. Data is not normalized.
     2. Snowflake Schema
        1. We will have as much table as much we have level in hierarchy
        2. One or more level away from a fact table.
        3. Data is normalized.

Both the schema has same dimension but different table representation.

* 1. Keys in modelling
     1. Data source will have primary, foreign key we call them **natural key**.
     2. But we should always create new key called **Surrogate key** in DW when we are copying rows.
     3. Snowflake schema will have PK in base table , second table , third table, and foreign key in secondary and third table in dimension table
     4. Star schema will only have one PK and No FK in Dimensional table
     5. Key will be different for fact table
     6. In fact-table PK will be combination of all foreign key of dimensional table even if natural key is also present
        1. Tuition payment fact ( student key , data key , tuition id , tuition payment)
     7. Question for dimension table
        1. Use surrogate/ natural key as primary & foreign key
           1. Add surrogate key as data brought in DW.
        2. Keep or discard natural key on dimension table
           1. Keep surrogate as primary key and natural as secondary key
        3. Keep or discard natural key from fact table
           1. Discard natural key.

# Dimensional modelling

1. Design dimension table for star and snowflake schema
   1. Dimension – context (Column Name)
   2. Dimension table – contect table, table
   3. Dimension <> dimension table
   4. Table – RDBMS
   5. RDBMS table require PK and FK
   6. DW PK = surrogate key

Faculty\_dim (facultyKey, facultyId, FacultyName)

Student\_Dim ( StudentKEy, Student ID, Student Name)

Include as much information in dimensional table

* 1. Hierarchal vs Flat Dimension
     1. College 🡨 Department 🡨 Faculty
     2. Student -----Flat dimension
  2. When, Where, What , From

1. Fact – measurement

Fact table – Fact + context

1. Type of fact table
   1. Transaction
      1. Record fact from a transaction
      2. One or more facts can be stored in a fact table, with governing some rule
      3. **Tuition payment Fact[ tutionPayment, Student\_Key, Date\_Key]**
      4. Rule
         1. Fact should be at same level mean they should have same dimension name (column name) if student has one extra dimension called campus then this tule will not obliged
         2. Fact should occur simultaneously
         3. Tuition bill, tuition payment are at same level, if we check the dimension table but they are in not occurring simultaneously.
         4. Tuition bill amount and activity bill can come in same fact table, as they are at same level, and they occur simultaneously
      5. Why not to violate rule
         1. This will complicate the data analysis
         2. Require SQL workaround to do it
         3. Billing and payment are two different business process\
         4. Business process belong to there respective fact table
   2. Periodic Snapshot
      1. Track a measurement at regular interval of time

Two type of periodic snapshot fact table

* + - 1. Aggregated result of regular transaction
         1. **Meal card payment periodic fact (student key , payment key , campus food key , mealcard amount)**
         2. Track and analyse the end of week meal amount balance throughout the semester.
         3. If I use Transaction Fact table , the table will look like

**MealCardPayment(StudentKey, PaymentKey,Mealcardamount)** here the meal card amount is each time transaction amount. And this gonna be little more complicated if we need to do weekly analysis so we use periodic snapshot

**MealCardPayment(StudentKey, PaymentKey,MealcardamountEOW)**  This will keep the end of week balance only.

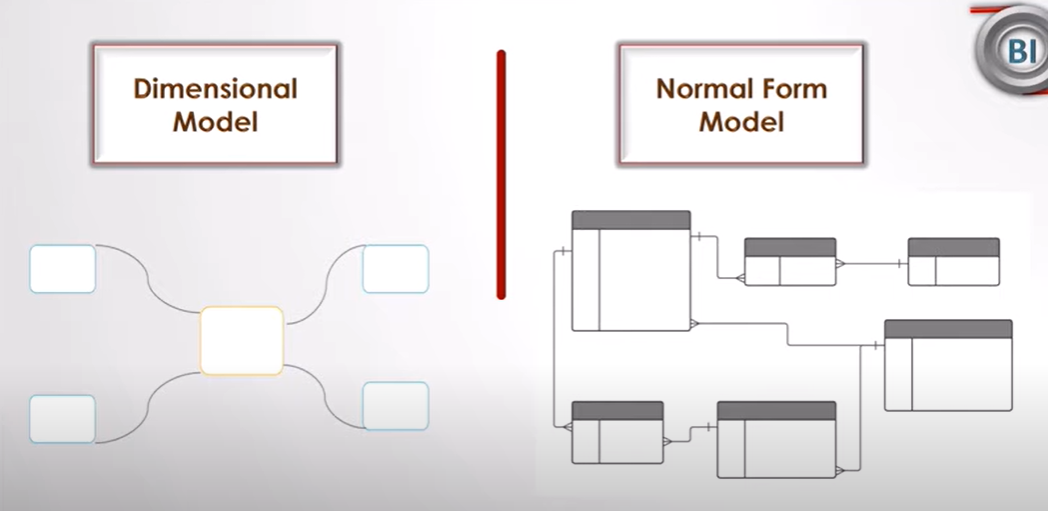
* + - 1. facts that are not related to regular transaction
         1. Here we work with semi additive fact, these facts are not transaction.
         2. Like % achieved, water level. Temperature etc
  1. Accumulating Snapshot
     1. Track a progress of business process at various stages
     2. Here we want to measure elapsed time to complete certain stage of a process.
     3. Include both completed and in progress stages , to analyse the progress
     4. We can measure other facts also as they grow with time elapsed in each phase of process.
     5. Here we have one to many relationships between fact and dimensional table

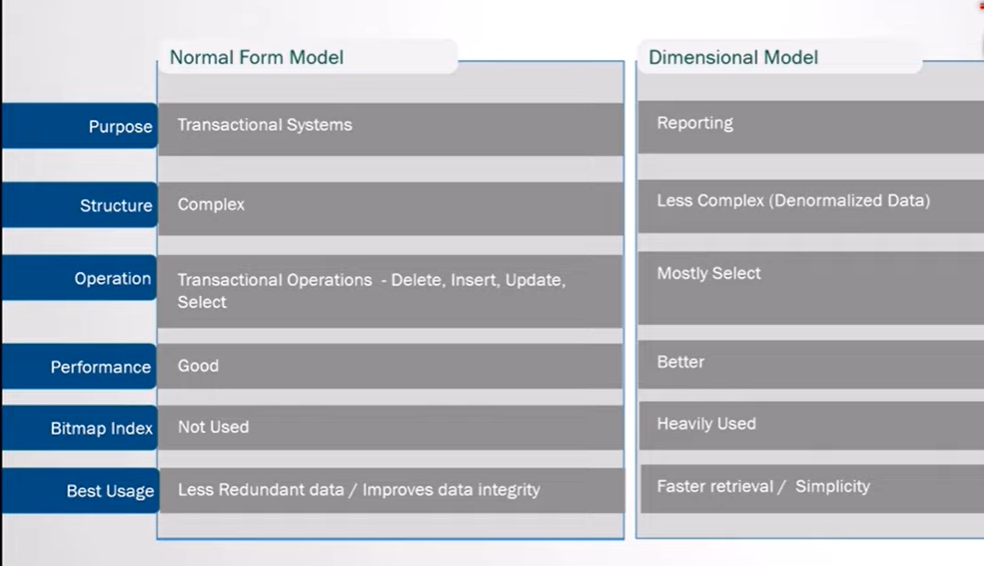
Submit date key , primary approval date key, secondary approval date key . all are pointing to same date dimension table.

* 1. Factless
     1. Record occurrence of a transaction without a measurement.
        1. Here we want to track the transaction , but we can not measure this transaction
        2. In other words we want to track any event related to some transaction
        3. Track if student has registered for webinar or not
        4. Track if patient are taking there medicine or not
        5. Medicine\_consumption\_Fact ( patient key , Medicine Key, taken\_dateKey)
        6. Here we do not have any fact to measure.
     2. Record coverage or eligibility relationship

# Dimensional Modelling part 2

* Fact – anything quantitative. Measuring
* Dimension- anything descriptive and level based (university🡪 college🡪department🡪hostel🡪batch year ), not number  
   we find dimension using , What(Product),When(time) , Where(Store), From(People)
* Advantage
  + Performance
    - Scalable
      * We can add new dimensional table, fact table
      * We can create new dimension also
    - Faster query
      * Data is not normalized, means duplicate rows will be present
    - Common Schema used Star schema, Snow flake schema
  + Simplicity
* Every Business process will have there own dimensional modelling tables like HR, SALE
* Easy to understand.



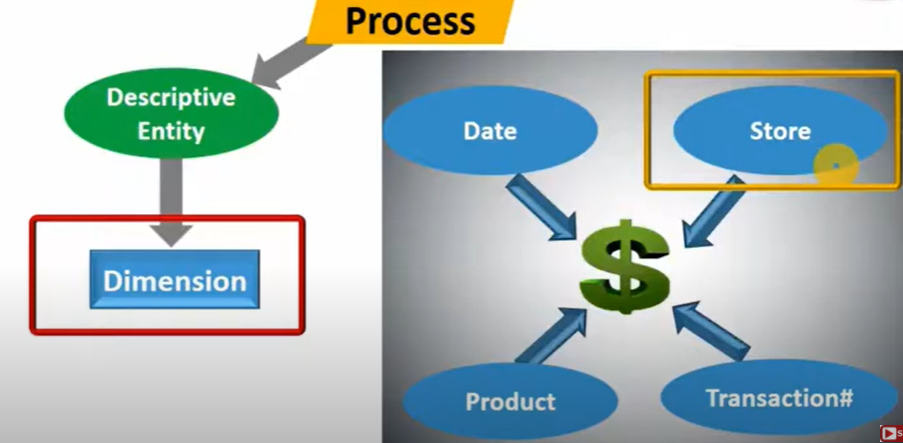


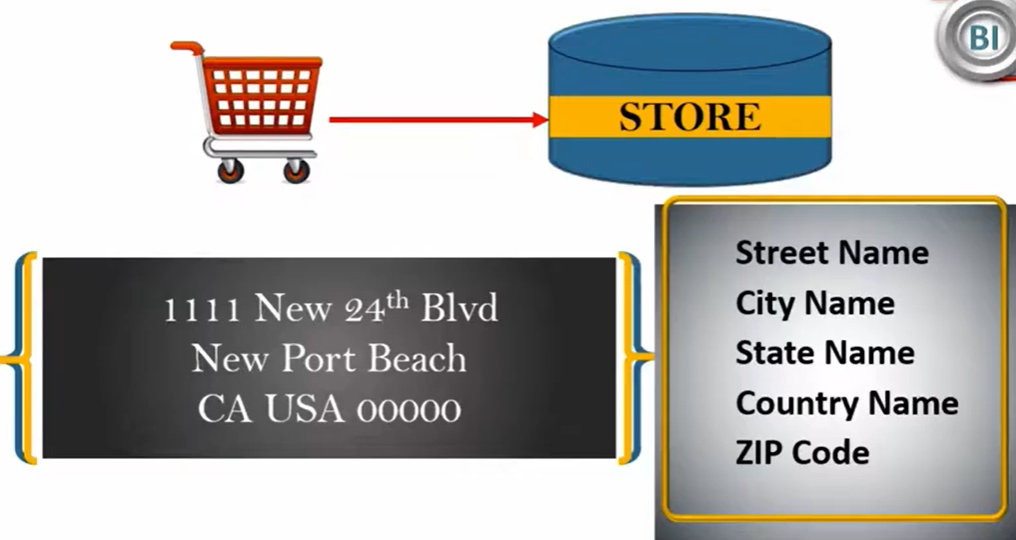
Sale

[time, Store, product, Customer Name **,quantity, invoice amount, unit sold. Invoice no**]

## Structure of Dimensional Table

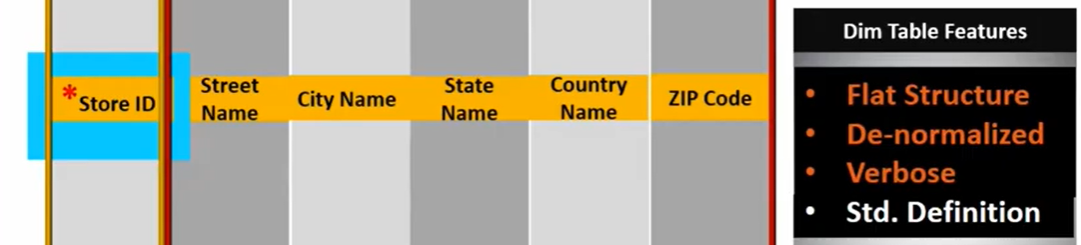
### Star Schema

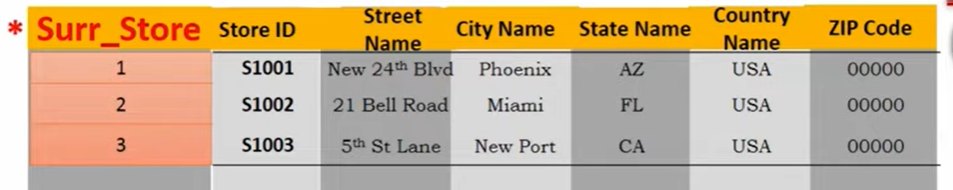




Dimension table will have Key and other descriptive column

Dimension table PK(Surrogate Key) will act as foreign key for fact table



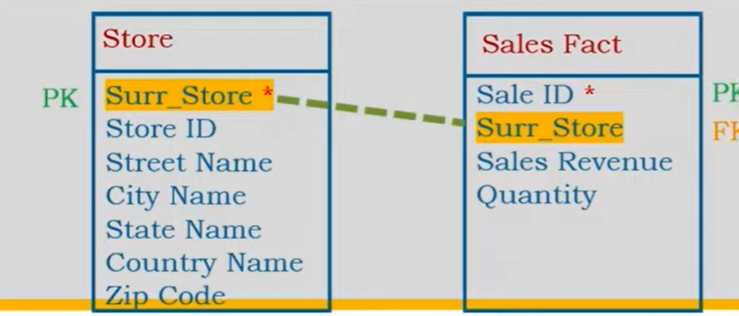


**Natural Key**

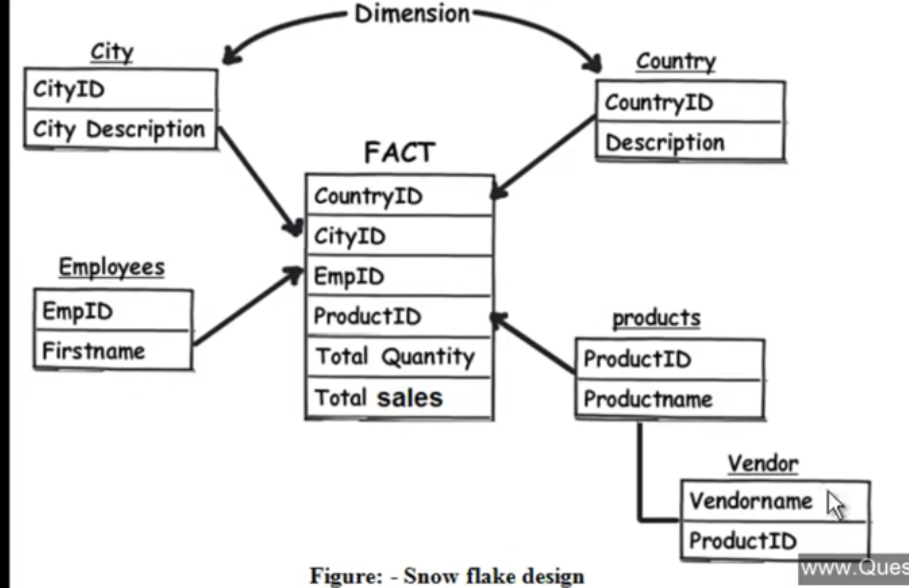
**Surrogate Key**

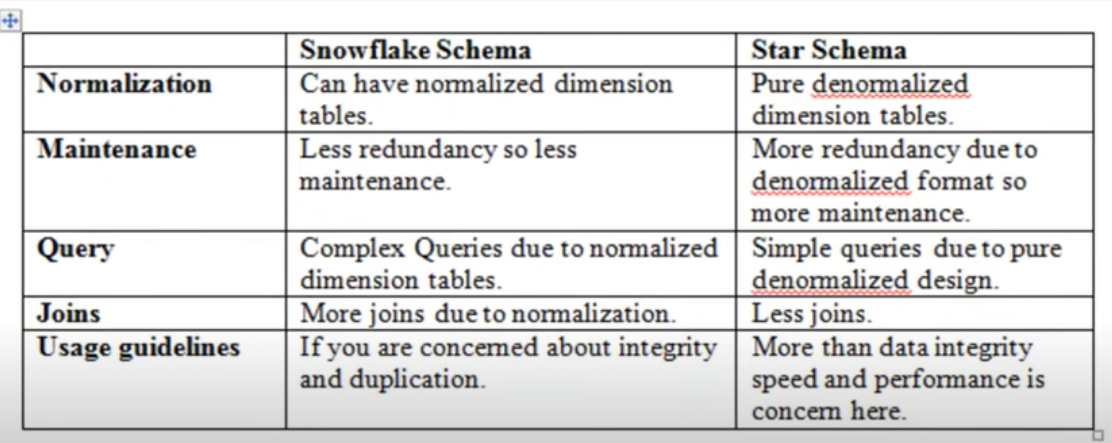
Both dimensional and fact table holds surrogate key.

For fact table mostly we tale combination of multiple surrogate key of dimensional table



### Snow Flake Schema





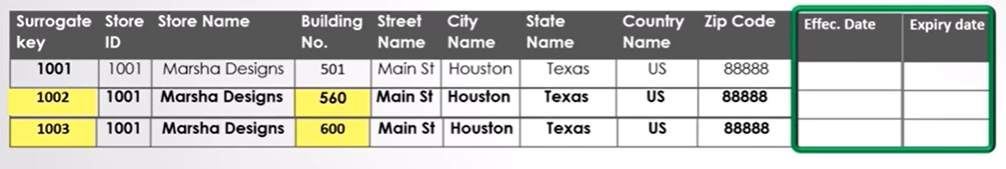
## Type of dimension

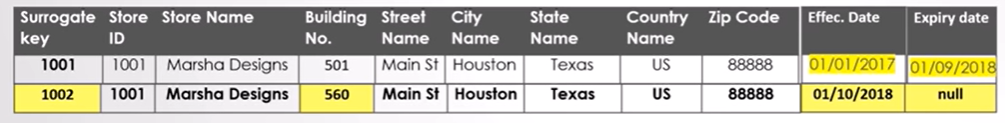
1. **Slowly Changing Dimension**

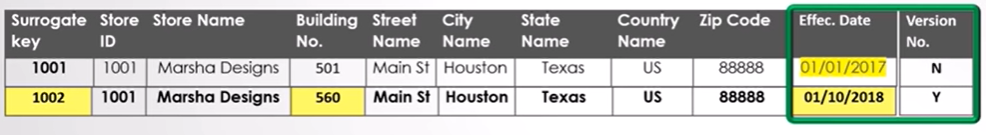
DW is time variant and Non volatile. So how are we going to save time changing dimesion like address , phone no.

How do we want to store current data, historical data , Both

* 1. Type 1 Overwrite the value
     1. It will replace older value with new one , so it will not store historical data
     2. Use it for spelling correction
  2. Type 2 preserve history using flag
     1. Which one is current record which one is older



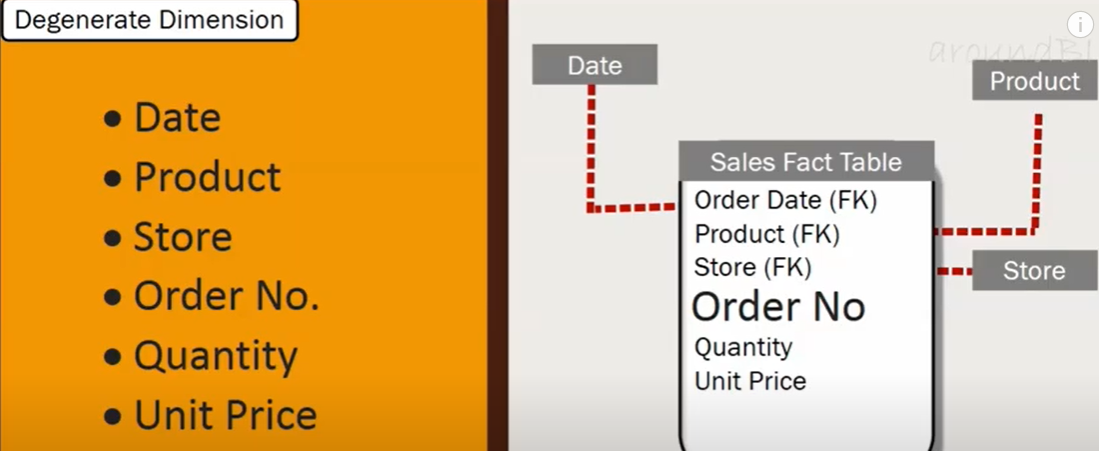




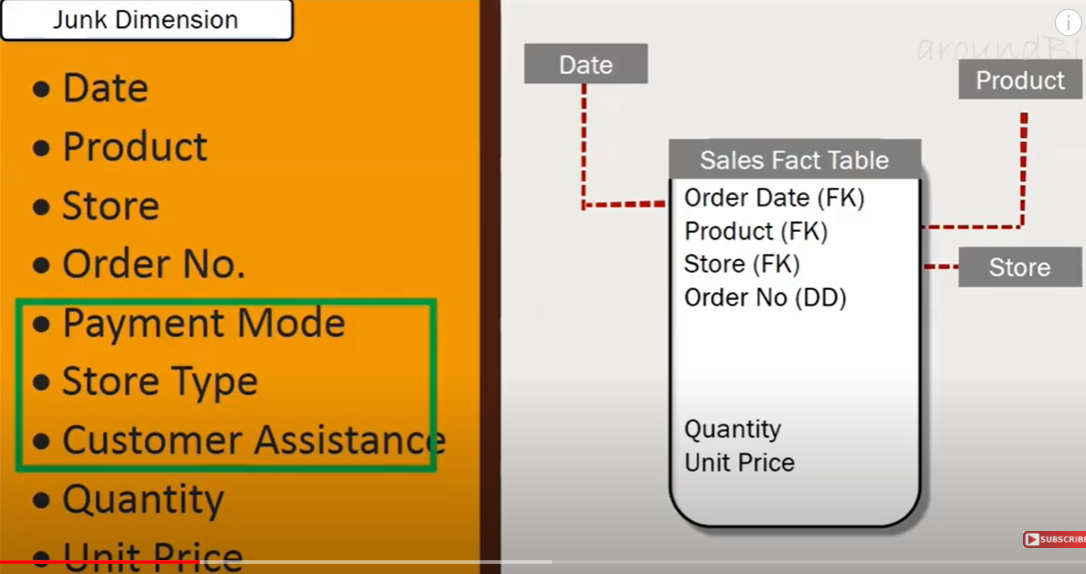
* 1. Type 3 Preserve Partial History

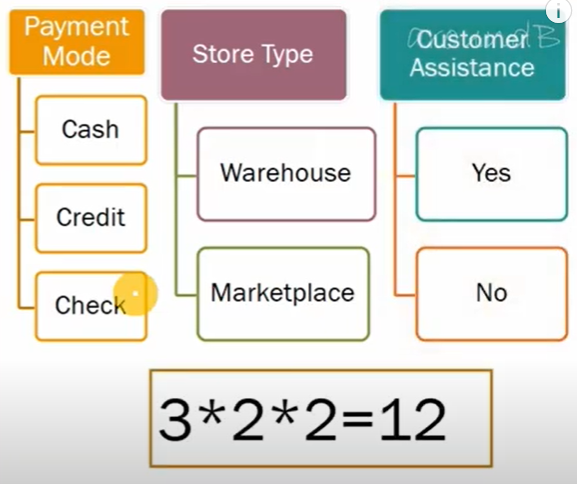


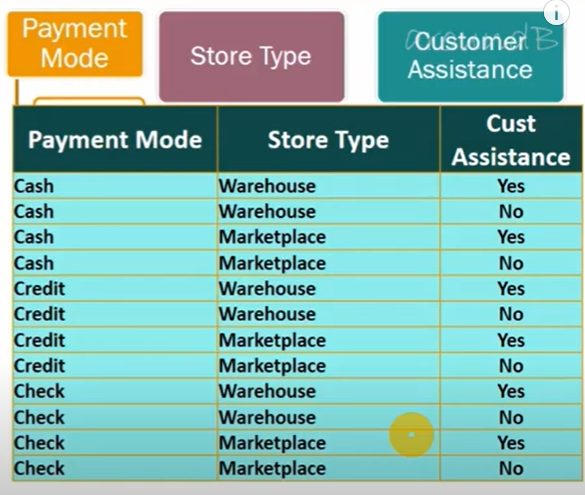
1. Degenerate Dimension
   1. It does not have its own dimensional table.
   2. This dimension key will be present in fact table. But it will not help to join with any dimensional table
   3. Ex- invoice no associated with fact table of sale, order no, policy no, GSTno
   4. It has one to one relationship with fact table
   5. Invoice no is a descriptive feature but it get stored in fact table
   6. [time, Store, product, Customer Name **,quantity, invoice amount, unit sold. Invoice no**]
   7. Here time will have dimensional table[year, month, day, hours, mm, sec]
   8. Store [country,city, store no, street, house no]
   9. Customer[first name, last name, phone no , ssn no]
   10. But invoice no will not have any other descriptive information

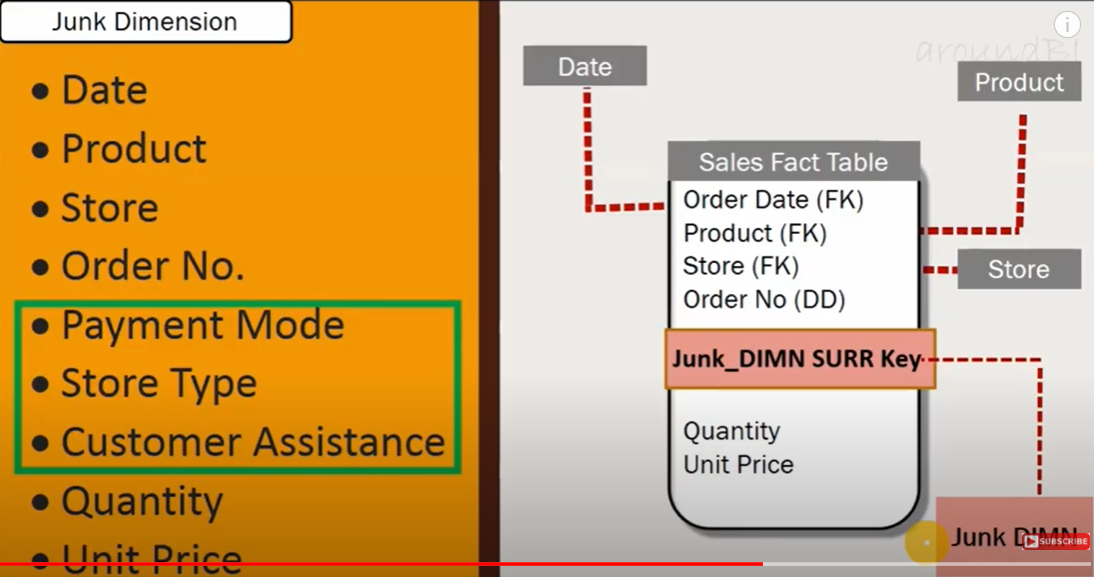


1. Junk
   1. Here we works with low cardinality column/categorical column/ more duplicate row for a column
   2. We will create a cross product of each junk column and assign them some PK.
   3. Then we will use those keys in our fact table
   4. Remember in a column , there should not be too much categories, as this may trigger a very big junk dimension.

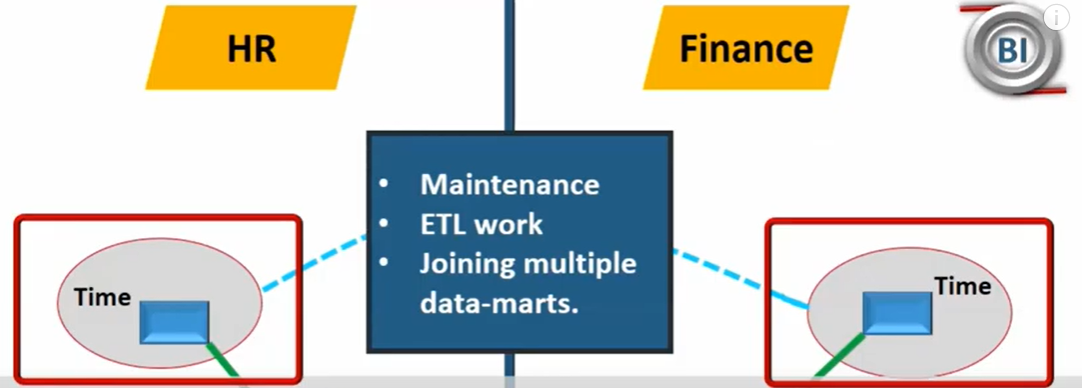


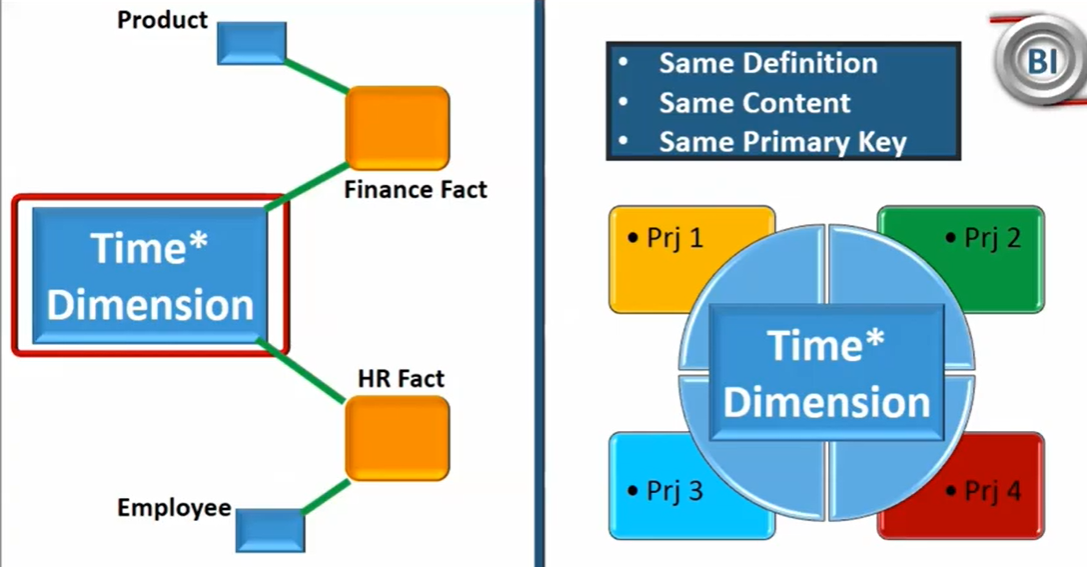






1. **Conformed Dimension**
   1. **Dimesion which has same meaning for every fact table**
   2. **Co**st associated with multiple dimensional table which are very identical
   3. Like time dimension table, now , two dimension for HR and Finance may use need same time table , but they may refer year as calender year , and financial year, which is differ in use, financial year start from April – march and calendar year is from jan to dec
   4. Ex- patient info, employee info
   5. Challanges

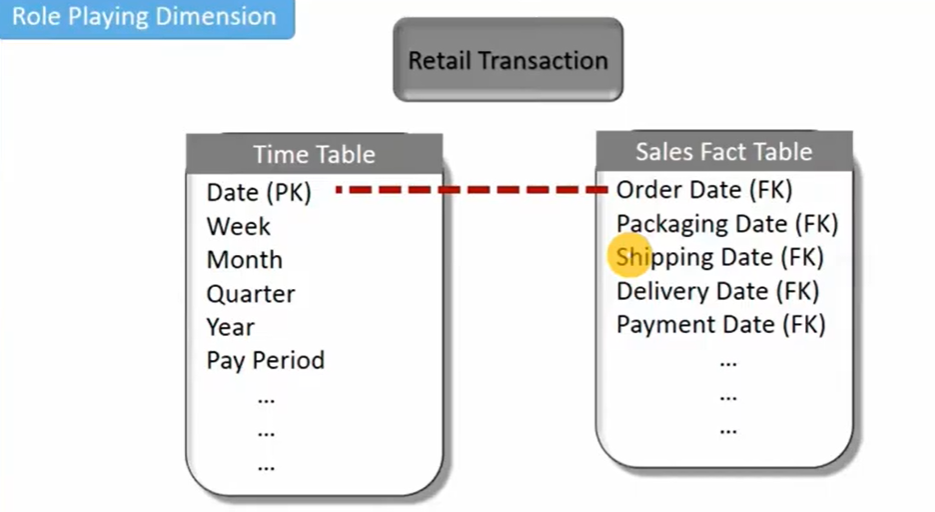


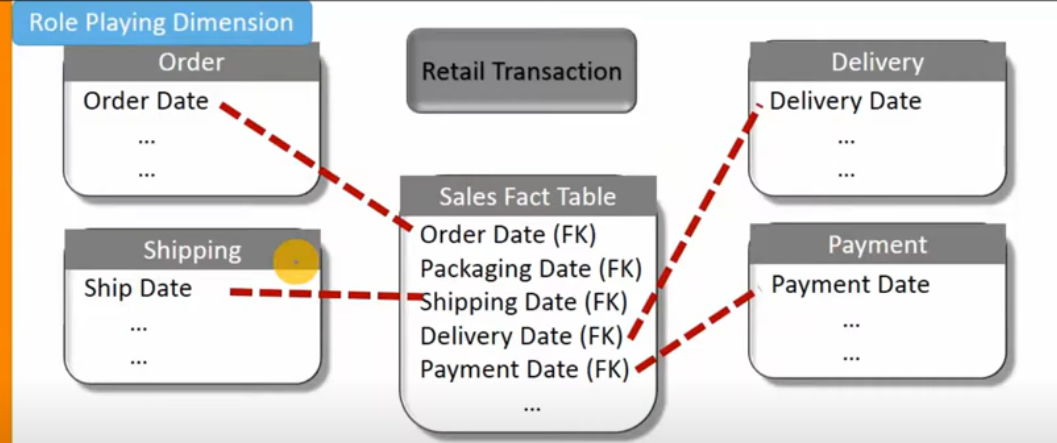


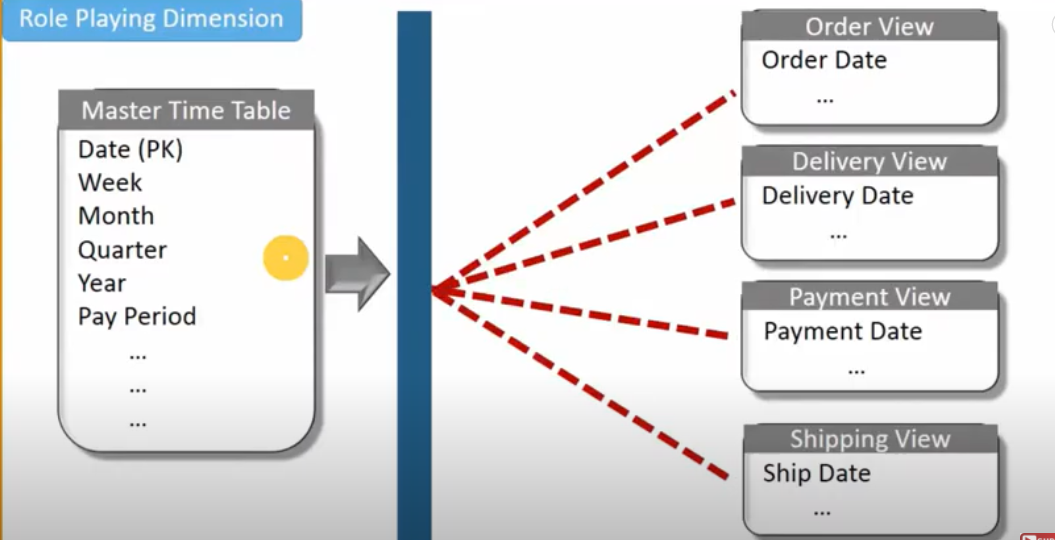
1. **Role Playing Dimension**

**If there is only on dimensional table for date, lets say order, delivery date is sae, now we put join on fact table, we will get wrong result**

**Ex-customer address/shipping address. We should have one master address dimension and multiple view which should get to fct table**







## Types of Fact

1. Additive Fact
   1. If it can be summed up or aggregated
2. Semi Additive fact
   1. Which is a snapshot
   2. Lets a product available in mart. So here we can not aggregate this , over time
   3. Here we can take average and check what is the average of product availability in mart
   4. After transforming we can take aggregate
3. Non Additive fact

**Like sale tax ,**

**Unit price**

**Percentage**

**Ratio**

## Structure of fact table

PK (Surrogate Key)

Composite Key of all FK (Some time we can use as PK)

Measure/Fact

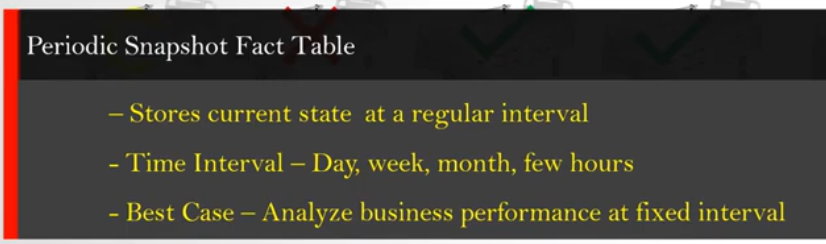
Descriptive column

Factor which help to define type of fact table

1. Grain – Granuality of data,   
   Date- it’s a day wise sale, month wise or yearly
2. It should define before configuring fact table
3. Agregate- combine
4. Transactional
   1. Sales Fact



* 1. Periodic Snapshot



* 1. Accumulating
     1. Shows different phase of a process

