Index

3. ETL Testing

    3.1 ETL Architecture

    3.2 Overview of ETL and importance of ETL TESTING?

    3.3 How DWH ETL Testing is different from the Application Testing.

1. Basics of DWH

1.1 Need of Data Warehouse in modern era.

Every Organization is in need of maintaining the data, this means maintaining data in organization can be different type such as Books, tapes, excels. As time goes on changing, the importance of the data and its maintenance and handling the data is also changing during modern era. Let's take an example of handling data when there are more than 1000 customers or employees in any organization and their data needs to be maintained and handle in future. All this has not been possible in practice until data warehousing came into reality.

In challenging times good decision-making becomes critical. The best decisions can be made when all the relevant data available is taken into consideration. The best possible source for that data is a well-designed data warehouse.

Some Organizations are really small which cannot afford a data warehouse. They can actually use data marts which interpret as data warehouse where their transactional data is very low. Data warehouse has become more efficient when handling data for larger organizations.

1.2 What is Data warehouse?

Data warehousing combines data from multiple, variable sources into one database which can be easily manipulated. This can be used for Analysis, Transformation and Reporting. Usually Larger companies use this data warehousing for analyzing trends over a period of time for viewing transactional data and plan accordingly.

Data warehouse is defined as subject oriented, Integrated, Time Variant and Non-Volatile collection of data for the management for decision making processing.

Subject Oriented: This is used to analyze particular subject area.

Integrated: This shows that integrates data from different sources.

Time variant: Historical data is usually maintained in a Data warehouse, i.e. retrieval can be for any period.

Non-Volatile: Once the data is placed in the data warehouse, it cannot be altered, which means we will never be able to alter the historical data.

1.3 Data Ware Housing Concepts

OLTP (Online Transaction Processing System)

OLTP is nothing but a database which actually stores the daily transactions which is called as the current data. Usually OLTP is used for more of the online applications where the application needs to update very frequently in order to maintain consistency in the data.

OLTP deals with the large number of data.

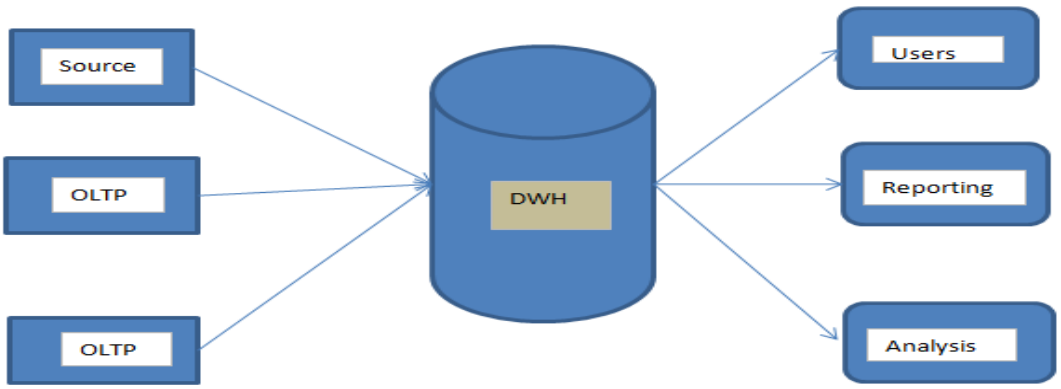
OLAP (Online Analytical Processing System

OLAP deals with analyzing the data for Decision making and planning. This actually deals with the aggregations and the data in OLAP will be from different data sources Compared to OLTP, OLAP deals with relatively small amount of data.

1.4 Difference between OLTP v/s OLAP

OLTP	OLAP
It is an online transactional system and manages database modification.	It is an online data retrieving and data analysis system.
Insert, Update and Delete information from the database.	Extract data for analyzing that helps in decision making.
OLTP and its transactions are the original source of data.	Different OLTPs database becomes the source of data for OLAP.
OLTP has short transactions.	OLAP has long transactions.
The processing time of a transaction is comparatively less in OLTP.	The processing time of a transaction is comparatively more in OLAP.
Simpler queries.	Complex queries.
Tables in OLTP database are normalized.	Tables in OLAP database are de-normalized.
OLTP database must maintain data integrity constraint.	OLAP database does not get frequently modified. Hence, data integrity is not affected.

1.5 Data Warehouse Architecture



1.6 Data Marts and its Types

Data Mart is nothing but the smallest version of the DWH. Data Mart deals with single subject area.
Data Mart focuses on one area hence they draw data from limited data source.

Example:
In Single Enterprise level which has multiple departments i.e. HR, Finance and Supply chain management those are nothing but Data Marts.

Types of Data Mart:

- Dependent DM
- Independent DM
- Hybrid DM

Dependent DM:

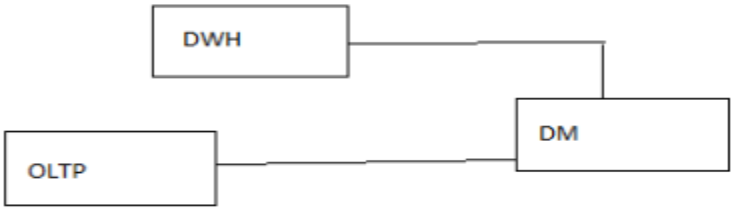
- Dependent data marts are created by drawing data directly from operational and DWH.

Independent DM:

- Independent data mart is created without the use of a central data warehouse.
- It has been created directly from source system and suitable for small organization.

Hybrid DM:

- In this type of data marts, data can feed from data warehouses or operational systems (OLTP).



Models:

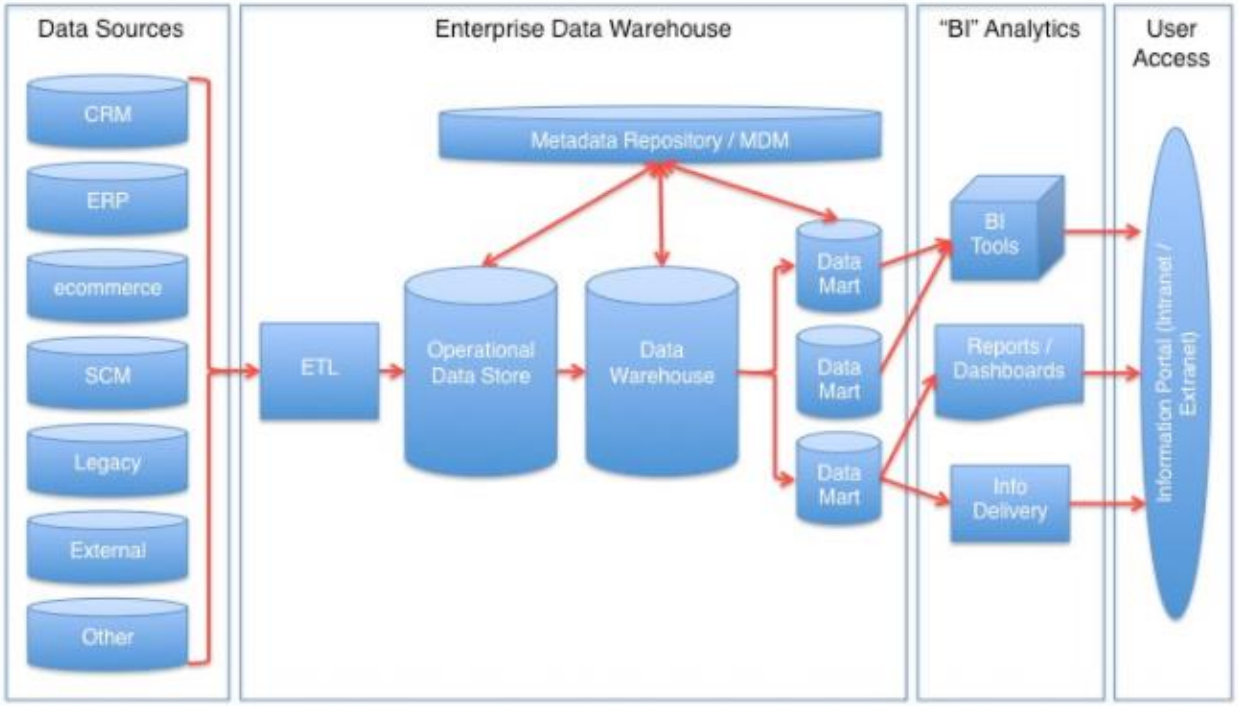
- **Top Down:** In top down approach first build DWH and then DM.
- **Bottom Up:** In Bottom up approach first build DM and then DWH.

Difference between DWH and DM:

Data Warehouse	Data Mart
Data warehouse is Multiple subject area.	Data mart is single subject area.
In data warehouse, Multiple data source	Data mart, limited data source.
Data warehouse is top-down model.	Data Mart is a bottom-up model.
To build a warehouse is difficult.	To build a data mart is easy.
In data warehouse, Fact constellation schema is used.	In Data Mart, Star schema and snowflake schema are used.
Data Warehouse is flexible.	Data Mart is not flexible.
Data Ware house has long life.	Data Mart has short life than warehouse.
In DWH, Data are contained in detail form.	In DM, Data are contained in summarized form.
Data Warehouse is vast in size.	Data Mart is smaller than warehouse

1.7 ODS (Operational Data Store) for Data Warehousing

An operational data store (ODS) is a central database that provides a understanding of the latest data from multiple transactional systems for operational reporting. It enables organizations to combine data in its original format from various sources into a single destination to make it available for business reporting.



Difference between ODS and DWH:

Operational Data Stores(ODS)	Data Warehouse(DWH)
ODS means for operational reporting and supports current or near real-time reporting requirements.	A data warehouse is intended for historical and trend analysis, usually reporting on a large volume of data.
An ODS consist of only a short window of data.	A data warehouse includes the entire history of data.
It is typically detailed data only.	It contains summarized and detailed data.
It is used for detailed decision making and operational reporting.	It is used for long term decision making and management reporting.
It is used at the operational level.	It is used at the managerial level.
It serves as conduct for data between operational and analytics system.	It serves as a repository for cleansed and consolidated data sets.
It is updated often as the transactions system generates new data.	It is usually updated in batch processing mode on a set schedule.

1.8 Difference between Database and Data Warehousing

Database	Data Warehouse
Database has the current data which has a chance of Updating day by day.	Data warehouse Stores the Historical data where the accuracy is maintained over period of time.
Contains the Day to day operations data.	Contains the Long term Operations data.
Database professionals, agents access this particular data.	Managers, Analysts access the data warehouse data
Data in the Database will always be in the form of Input.	Data in the Data warehouse will always be the output data where it is used for Analyzing trends.
Database is used for Transactions.	Data warehouse is used for Analytics.
Database has both Read/Write access.	Data warehouse has only the Read access.
Database contains only few number of records compared to a Data warehouse.	Data warehouse contains millions of records as the DWH gets refreshed with historic data.
Database is always Normalized.	Data warehouse is Demoralized.
The data view in the database will always be Relational.	The data view in the Data warehouse id always Multidimensional
Database contains the detailed data.	Data warehouse contains the consolidated data.

2. Data Warehousing Concepts

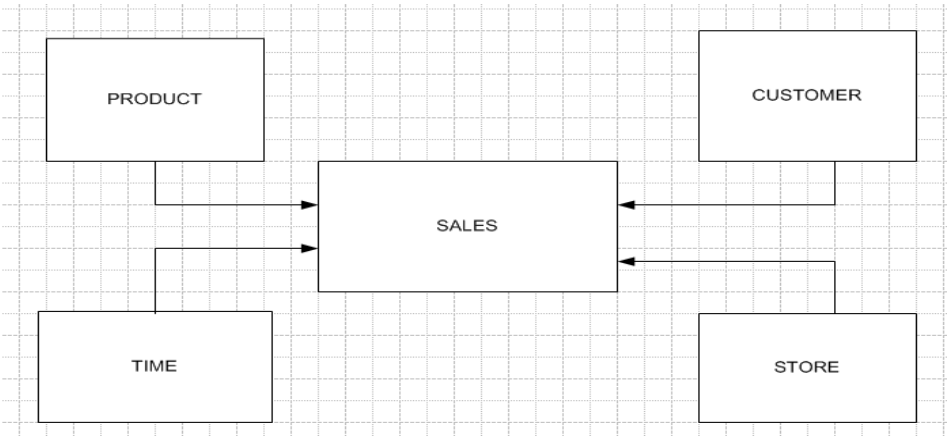
2.1 Data Models:

Data model tells how the logical structure of a database is modeled. Data Models are fundamental entities to introduce ideas in DBMS. Data models define how data is connected to each other and how it will be processed and stored inside the system.

2.1.1 Conceptual Data Model

This usually pictures the highest level of relationship between the entities.

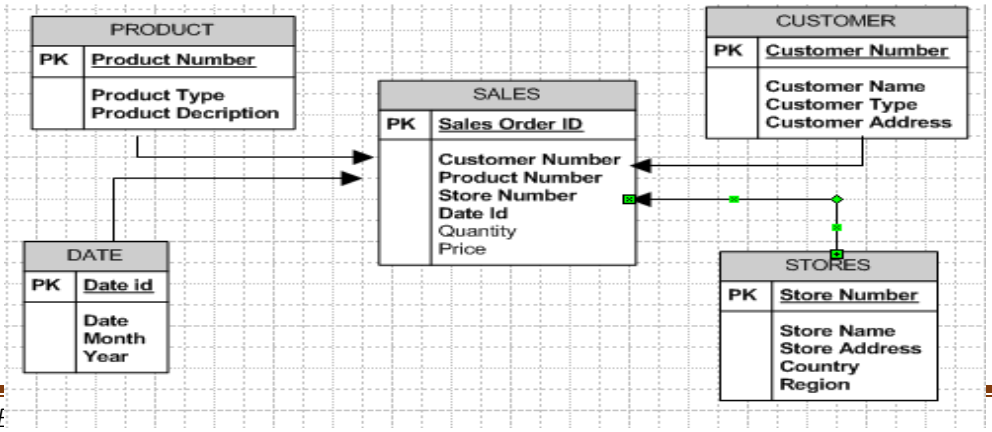
- Displays the important entities and the relationships among them.
- No attribute is specified.
- No primary key is specified.



2.2.2 Logical Data Model

Logical Data Model defines the data as much as possible, to show how they can be physically implemented in the database.

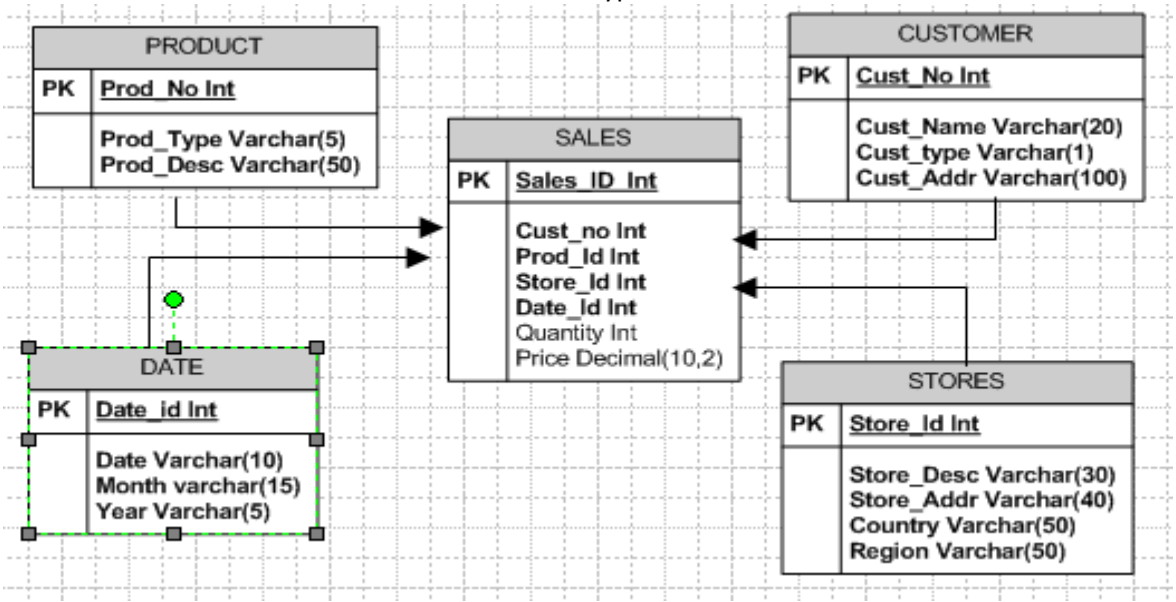
- Displays all the entities and the attributes and the relationships between them.
- Primary key for each entity is specified.
- Foreign keys for each entity if exists is specified.
- Normalization is performed.



2.1.3 Physical Data Model

Physical Model defines how the model physically exists in the system.

- Displays all the tables and columns.
- Displays foreign keys.
- Displays lookup tables.
- Change the relationships into foreign keys.
- Entity now becomes table.
- Attribute now becomes column Data types are also shown in this model.

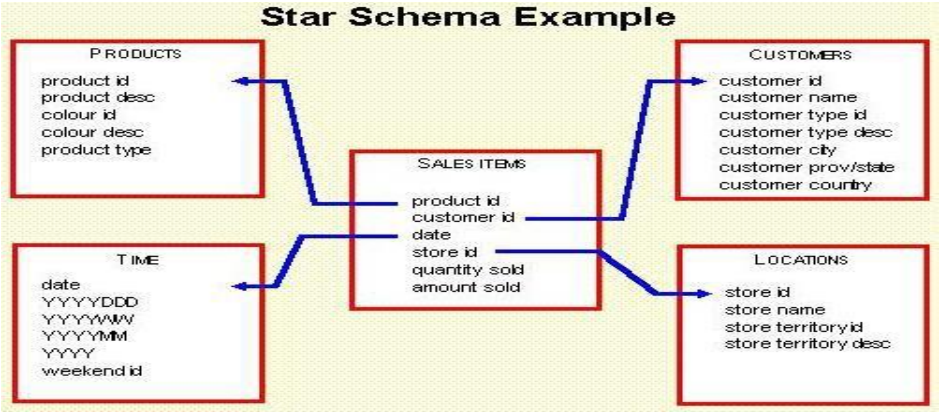


2.2 Data Schemas

A database schema defines its entities and the relationship among them. Database schema is a descriptive detail of the database, which can be implemented by means of schema diagrams. All these activities are done by database designer to principle architect in order to give some ease of understanding of detailed database.

2.2.1 Star Schema

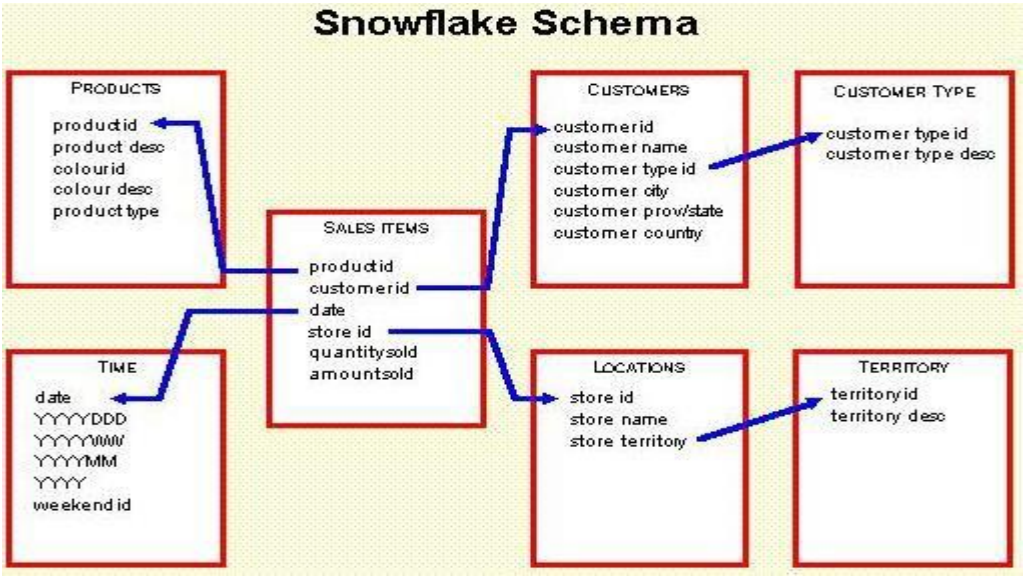
Star consists of the fact table in the middle and the dimensional table around it.



Fact table is usually a sum of all the dimensions. Dimension table is a single entity. A primary key in a dimension table is represented as a foreign key in a fact table.

2.2.2 Snowflake Schema:

This is an extension of the Star Schema, where each point of a start is divided into more granular level. Each Dimension table is still normalized into multiple lookup tables. The main advantage of the snowflake schema is to improve the query performance due to minimized disk storage and joining can also happen on smaller lookup tables.
Disadvantage is maintenance efforts to manage the number of lookup tables.

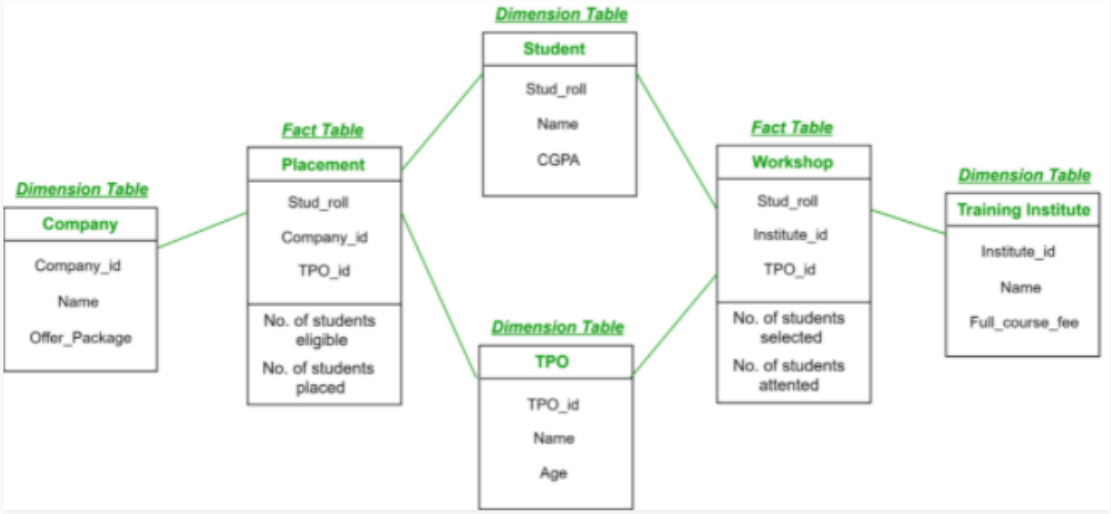


Difference between Star and Snowflake Schema

Star Schema	Snowflake Schema
In star schema, The fact tables and the dimension tables are contained.	In snowflake schema, The fact tables, dimension tables as well as sub dimension tables are contained.
Star schema is a top-down model.	Snowflake Schema is a bottom-up model.
Star schema uses more space.	Snowflake Schema uses less space.
It takes less time for the execution of queries.	Snowflake Schema takes more time than star schema for the execution of queries.
In star schema, Normalization is not used.	In Snowflake Schema, Both normalization and de-normalization are used.
It's design is very simple.	Snowflake Schema design is complex.
The query complexity of star schema is low.	The query complexity of snowflake schema is higher than star schema.
It's understanding is very simple.	it's understanding is difficult.
It has less number of foreign keys.	it has more number of foreign keys.
It has high data redundancy.	While it has low data redundancy.
Performance is high	Performance is low.

2.2.3 Fact constellation schema

Fact Constellation is a schema for representing multidimensional model. It is a collection of multiple fact tables having some common dimension tables. It can be viewed as a collection of several star schemas and hence, also known as Galaxy schema. It is one of the widely used schemas for Data warehouse designing and it is much more complex than star and snowflake schema.



In above Example:

Placement is a fact table having attributes: (Stud_roll, Company_id, TPO_id)
With facts: (Number of students eligible, Number of students placed).

Workshop is a fact table having attributes :(Stud roll, Institute_id, TPO_id)
With facts: (Number of students selected, Number of students attended the workshop).

Company is a dimension table having attributes: (Company_id, Name, Offer_package).

Student is a dimension table having attributes: (Student_roll, Name, CGPA).

TPO is a dimension table having attributes: (TPO_id, Name, Age).

Training Institute is a dimension table having attributes: (Institute_id, Name, Full_course_fee).

So, there are two fact tables namely, Placement and Workshop which are part of two different star schemas having dimension tables – Company, Student and TPO in Star schema with fact table Placement and dimension tables – Training Institute, Student and TPO in Star schema with fact table Workshop. Both the star schema have two dimension tables common and hence, forming a fact constellation or galaxy schema

2.3 Dimensions

A dimension table contains dimensions of a fact. They are joined to fact table via a foreign key. Dimension tables are de-normalized tables. The Dimension Attributes are the various columns in a dimension table Dimensions offers **descriptive characteristics** of the facts with the help of their attributes. The dimension can also contain one or more hierarchical relationships.

Table which describes dimension involved are called as dimension table.

Types of Dimensions

1. SCD
2. Confirmed dimension
3. Degenerated dimension
4. Junk dimension
5. Roll playing dimension
6. Static dimension
7. Shrunken dimension

1. SCD (Slowly changing dimension):

Dimension that change slowly over a period of time rather than changing regularly is group in SCD.

For ex: Location, Name, Address, City etc.

Slowly Changing Dimensions are often categorized into three types namely

- SCD Type1 (SCD 1)
- SCD Type2 (SCD 2)
- SCD Type3 (SCD 3)

Ex: Consider a customer with name SOHAN who is living in Sydney from 2018.This can be depicted in a table as below:

CUSTOMER NUMBER	CUTOMER NAME	YEAR	location
1	SOHAN	2018	SYDNEY

SCD Type 1(SCD 1)

Replaces the old entry with the new value.

The customer SOHAN has moved from Sydney to Perth in the year 2021

In this Type 1 SCD, the table will be changed as below:

CUSTOMER NUMBER	CUSTOMER NAME	YEAR	LOCATION
1	SOHAN	2021	PERTH

In this case, the previous entry which is treated as history is lost.

SCD Type 2 (SCD 2)

The New record is inserted into the same table, In this Type 2 SCD, the table will be changed as below:

CUSTOMER NUMBER	CUSTOMER NAME	YEAR	LOCATION	ETL_flag	START_DATE	END_DATE
1	SOHAN	2020	Sydney	N	1/2/2020	1/2/2021
1	SOHAN	2021	Perth	Y	1/2/2021	NULL

To identify the current record and history record the columns like **ETL_Flag or START_DATE and END_DATE** is used.

In this case, each record will be treated differently, which makes the table to grow fast and complicates the ETL process. This is mainly for tracking the historical changes.

SCD Type 3 (SCD 3)

New fields/Columns are added to the table to main the history.

In this Type 3 SCD, the table will be changed as below:

CUSTOMER NUMBER	CUSTOMER NAME	YEAR	LOCATION	OLD YEAR	OLD LOCATION
1	SOHAN	2021	Perth	2020	Sydney

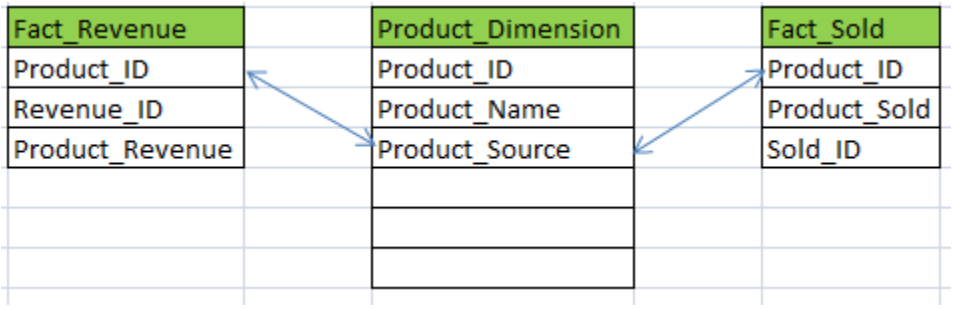
The previous record itself is modified such that neither the history is lost and even the new record is also displayed. But this can accommodate only one change. This Type3 SCD is rarely used, when the changes are prone to change only once.

2. Confirmed dimension

A dimension table used across the multiple facts.

It is a dimension that has same meaning to every fact table into the structure.

Example:



For the above table Product_ID is the dimension which has the same meaning in each of the fact table i.e. Fact_revenue and Fact_sold.

3. Degenerated dimension

A degenerate dimension which is derived from the fact table and doesn't have its dimensions.

A degenerate dimension is when the dimension attribute is stored as part of the fact table and not in a separate table. Product id comes from product dimension table. Invoice number is a standalone attribute and has no other attributes associated with it. An invoice number can be crucial since the business would want to know the quantity of the products.

Example:

Invoice_Number	Product_ID	Quantity	Amount
22233676	1234	4	1000
22233676	5678	3	1000
22233676	9876	4	1000
22233676	4325	2	1000
22233676	6543	1	1000

4. Junk dimension

It is a single table with a combination of different and unrelated attributes to avoid having a large number of foreign keys in the fact table. They are often created to manage the foreign keys created by rapidly changing dimensions.

Junk dimensions are used to reduce the number of dimensions in the dimensional model and reduce the number of columns in the fact table.

Example:

Car colors (red, black, blue, etc.) and body style (sedan, van, SUV, etc.) As you can see these are limited in number and, if created as single dimensions, the dimensions would be limited to a single attribute. In order to eliminate these small dimensions, we create a single "junk" dimension which cross joins all possible attributes into a single dimension which will be used in the fact table.

DIM_Car_Attributes
DCA_Key
DCA_Color
DCA_Body

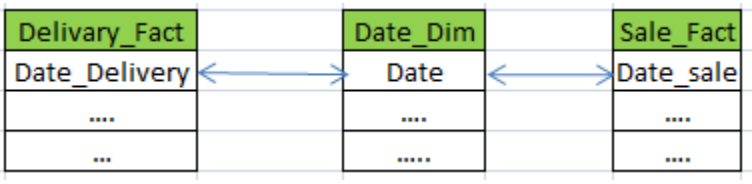
DCA_key	DCA_Color	DCA_Body
1	White	HatchBack
2	Black	HatchBack
3	Blue	HatchBack
4	Red	HatchBack
…	…	…
11	White	Sedan
12	Black	Sedan
13	Blue	Sedan
14	Red	Sedan

5. Roll playing dimension

Dimension which is often used for multiple purpose within the same database are called roll playing dimension.

Example: Date

A fact table may include foreign keys for both ship date and delivery date.



6. Static dimension

Static dimensions are nothing but the constant. It not **extracted** from the **real data source** but created in the data warehouse context.

Example:

Gender is constant i.e. Male and Female

7. Shrunken dimension

A shrunken dimension is a subset of another aspect. This is basically used in snow flake schema.

Example:

If the product dimension is a snowflake, then there is a separate table for the product category, a shrunken dimension.

2.4 **Facts:**

Facts are nothing but measures or we can simply called it as numbers.

The fact table is a central table in the data schemas. It is found in the centre of a star schema or snowflake schema and surrounded by a dimension table. It contains the facts of a particular business process, such as sales revenue by month. Facts are known as measurements or matrices. It captures a measurement or a metric.

2.4.1 Types of Facts

1. Additive Fact

2. Semi-Additive Fact

3. Non-Additive Fact

1 .Additive Fact:
Additive facts are facts that can be summed up through all of the dimensions in the fact table.

Example:
Let us use examples to illustrate Additive facts. The first example assumes that we are a retailer, and we have a fact_Sale table with the following columns:

Fact_Sale
Date
Store
Product
Sales_Amount

The purpose of this table is to record the sales amount for each product in each store on a daily basis. **Sales_Amount** is the fact. In this case, **Sales_Amount** is an additive fact, because you can sum up this fact along any of the three dimensions present in the fact table -- date, store, and product.

The sum of **Sales_Amount** for all 7 days in a week represents the total sales amount for that week.

2. Semi-Additive:
Semi-additive facts are facts that can be summed up for some of the dimensions in the fact table, but not the others.

3. Non-Additive:
Non-additive facts are facts that cannot be summed up for any of the dimensions present in the fact table.

Example: For Semi/Non-Additive
A bank with the following fact table:

Fact_Current_Balance
Date
Account
Current_Balance
Profit_Margin

The purpose of this table is to record the current balance for each account at the end of each day, as well as the profit margin for each account for each day.
Current_Balance and **Profit_Margin** are the facts. **Current_Balance** is a semi-additive fact, as it makes sense to add them up for all accounts (what's the total current balance for all accounts in the bank?).

But it does not make sense to add them up through time (adding up all current balances for a given account for each day of the month does not give us any useful information). **Profit_Margin** is a non-additive fact, for it does not make sense to add them up for the account level or the day

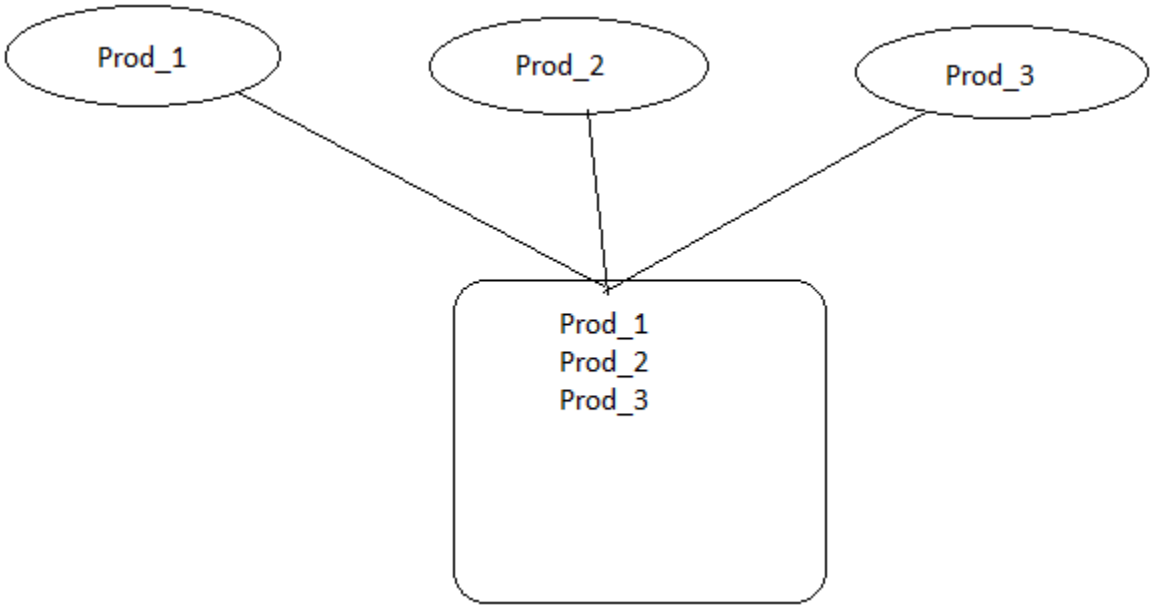
2.4.2 Types of fact Table.

1. Transaction Fact Table
2. Snapshot Fact Table
3. Accumulated fact Table
4. Fact less Fact Table

1. Transaction Fact Table:

The transaction fact table is a basic approach to operate the businesses. These fact tables represent an event that occurs at the primary point. A line exists in the fact table for the customer or product when the transaction occurs.

Example:



Many rows in a fact table connect to a product because they are involved in multiple transactions. Transaction data is often structured quickly in a one-dimensional framework.

2. Snapshot Fact Table:

This type of fact table describes the state of things in a particular instance of time, and usually includes more semi-additive and non-additive facts.

Example:

Daily balances fact can be summed up through the customers' dimension but not through the time dimension.

3. Accumulated fact Table

This type of fact table is used to show the activity of a process that has a well-defined beginning and end.

Example:

The processing of an order, an order moves through specific steps until it is fully processed. As steps towards fulfilling the order are completed, the associated row in the fact table is updated.

4. Fact less Fact Table:

We have also a transaction fact tables which contain no measures. We call it as fact less fact tables. These tables are used to capture the action of the business process.

Fact less Fact table is fact table that does not contain fact. They contain only dimensional keys and it captures event that happen only at information level.

Example:

Think about a record of student attendance in classes. In this case, the fact table would consist of 3 dimensions: the student dimension, the time dimension, and the class dimension. This fact less fact table would look like the following:

Fact_Attendance
Student_ID
Class_ID
Time_ID

The only measure that you can possibly attach to each combination is "1" to show the presence of that particular combination. However, adding a fact that always shows 1 is redundant because we can simply use the COUNT function in SQL to answer the same questions.

Fact less fact tables offers the most flexibility in data warehouse design. For example, one can easily answer the following questions with this fact less fact table:

- How many students attended a particular class on a particular day?
- How many classes on average does a student attend on a given day?

Without using a fact less fact table, we will need two separate fact tables to answer the above two questions. With the above fact less fact table, it becomes the only fact table that's needed.

2.5 Normalization and Demoralization

Normalization is a process of eliminating the redundant data and storing the related information in a table.
The key points of Normalization are as below:

- Eliminating Redundant Data
- Faster Update
- Improve Performance
- Performance in Indexes

Below are the different Normal Forms

First Normal Form:

If the table is said to be in the 1st Normal Form it should follow the below rules
Each cell must have one value. Eliminating Duplicate Columns
Create separate table for the group of the related data and each row should be identified by Primary Key.

Let us take an example here:

Name	Department	Phone Number	Salary	TAX
Name 1	Computers	123,545,886,887,890,000,000	4000	40
Name 2	Electronics	123,456,782,345,666,000,000,000	5000	50
Name 3	Civil	4567890	3000	30

In the above table we see that there are different phone numbers for the single Name and we have to remove these duplicates by uniquely identifying each of them and giving a unique identification by giving them the Primary Key.

Now the below table is redesigned such that it is in the First Normal Form.

ID	Name	Department	Phone Number	Salary	TAX
1	Name 1	Computers	12354588	4000	40
2	Name 1	Computers	6887890	4000	40
3	Name 1	Computers	123456	4000	40
4	Name 2	Electronics	12345678	4000	40
5	Name 2	Electronics	2345666	4000	40
6	Name 2	Electronics	789654333	4000	40
7	Name 3	Civil	4567890	3000	30

Second Normal Form:

If the table is said to be in the 2st Normal Form it should follow the below rules

- It should satisfy the 1st Normal Form.
- Separate the particular Columns, values are duplicated in each row should be placed in the separate Table.
- Create the relationship between the tables.

Here in the above table we see that the Name and the department columns are duplicated and in order to handle this we need to maintain the duplicates in the different table as below:

EMPID	Name	Department	Salary	TAX
1	Name 1	Computers	4000	40
2	Name 2	Electronics	4000	50
3	Name 3	Civil	3000	30

ID	EMPID	Phone Number
1	1	12354588
2	1	6887890
3	1	123456
4	2	12345678
5	2	2345666
6	2	789654333
7	3	4567890

Here in these tables above EMPID is treated as the primary Key for the First Table and the Foreign Key for the Second Table.

Third Normal Form:

If the table is said to be in the 3rd Normal Form it should follow the below rules

- It should satisfy the 2nd Normal Form.
- Separate the particular Columns that are not dependent on the primary key of the table.

EMPID	Name	Department	Salary
1	Name 1	Computers	4000
2	Name 2	Electronics	4000
3	Name 3	Civil	3000

ID	EMPID	Phone Number
1	1	12354588
2	1	6887890
3	1	123456
4	2	12345678
5	2	2345666
6	2	789654333
7	3	4567890

Salary	TAX
4000	40
3000	30

Fourth Normal Form:

If the table is said to be in the 4thNormal Form it should follow the below rules

- It should satisfy the 3rdNormal Form.
- The Non Key columns should be dependent on fully primary key instead of partial key, fits not separate the tables.

Here in the above as we have already put a primary key in the table during the First Normal Form the above tables are in Fourth Normal Form also.

De-Normalization:

De-normalization is a database optimization technique in which we add redundant (repetitive) data to one or more tables. This can help us avoid costly joins in a relational database. Note that de-normalization does not mean not doing normalization. It is an optimization technique that is applied after doing normalization.

Example:

We have two table students and branch after performing normalization. The student table has the attributes Roll_no, stud-name, age, and Branch_id.

Roll_No	Student_Name	Age	Branch_ID
11	Stevan	23	1
12	Mark	22	2
13	Nishem	25	1
14	Taylor	21	3
15	Kane	26	4

Additionally, the branch table is related to the student table with branch_id as the student table's foreign key.

Branch_ID	Branch_Name	Total_students
1	EC	150
2	CSE	100
3	Civil	60
4	Mech	80

A JOIN operation between these two tables is needed when we need to retrieve all student names as well as the branch name. Suppose we want to change the student name only, then it is great if the table is small. The issue here is that if the tables are big, joins on tables can take an excessively long time.

In this case, we'll update the database with de-normalization, redundancy, and extra effort to maximize the efficiency benefits of fewer joins. Therefore, we can add the branch name's data from the Branch table to the student table and optimizing the database.

3. ETL Testing

3.1 ETL Architecture

3.2 Overview of ETL and importance of ETL TESTING?

3.4 How DWH ETL Testing is different from the Application Testing.