# ETL Testing Overview

# Objectives

Session 1 :-

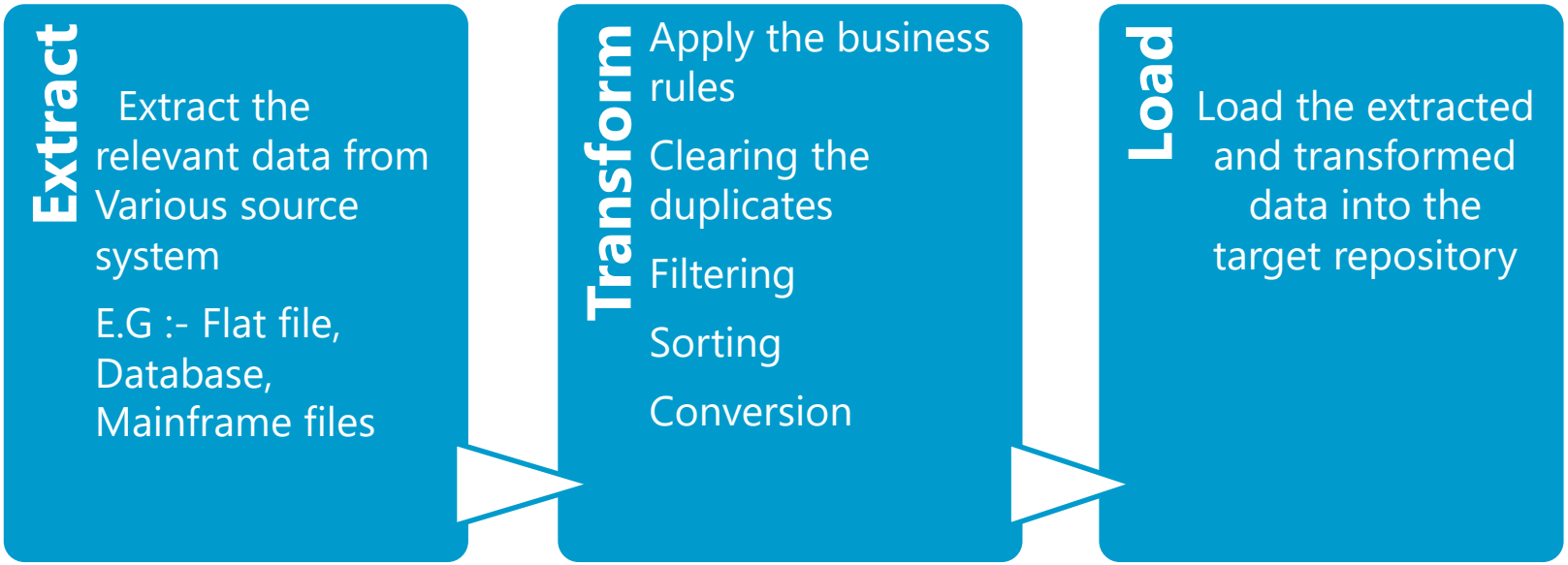
- What is process flow for ETL Testing?
- What is Data warehouse?
- ETL Tools, DBs, Reporting Tools in Market
- What are Principles of ETL Testing?
- Learning of various methods to perform ETL testing

Session 2 :-

ETL validations using talend

ETL automation Tool – One Click

# ETL Stands For Extract ,Transform and Load



# Data warehouse

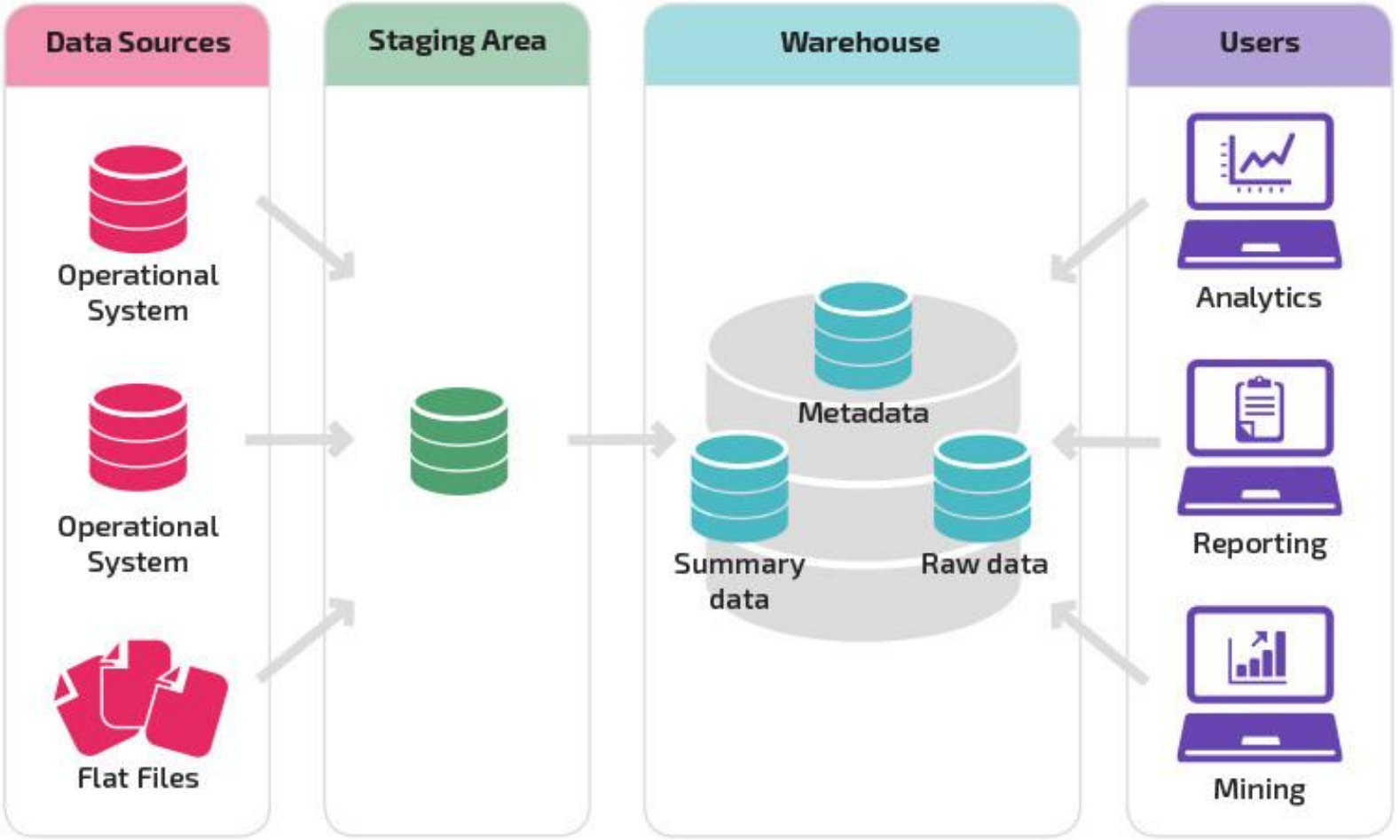
- Data warehouse is the database used for reporting and data analysis. It is central repository of data which is created by integrating data from multiple sources.
- Data warehouse stores current as well as historic data and are used for creating reports.
- Typical  ETL based data warehouse uses
  - Staging Layer
  - ODS Layer
  - Data Mart

Staging layer or Staging database stores raw data extracted from each of source data system.

ODS (Operational Data Store) layer is designed where data can be scrubbed , resolved for redundancy  and checked for compliance with  the corresponding business rules.

Data mart is subset of data warehouse that focuses more on subject areas. For E.g like sales, Finance and Department data

## Data ware House architecture



## Difference between Data warehouse and Database testing

There is popular misunderstanding that database testing and data warehouse is similar while the fact is that both hold different direction in testing

Database testing is done using smaller scale of data normally with OLTP (Online Transaction Processing) types of database while data warehouse testing is done with large volume with data involving OLAP (Online Analytical Processing) database.

In database testing normally data is consistently injected from uniform sources while in data warehouse testing most of the data comes from different kind of sources which are sequentially inconsistent

We generally perform CURD (Create, Update, Read and Delete) operation in database testing while in data warehouse testing we use read only (Select) operation

# ETL Testing Process Flow

- Business requirement understanding
- Query Resolution from business team & stakeholders
- Test Estimation
- Test planning based on the inputs from test estimation and business requirement
- Designing test cases and test scenarios from all the available inputs
- Once all the test cases are ready with required test data and are approved, testing team proceed to perform pre-execution check
- Lastly execution is performed till exit criteria are met
- Upon successful completion summary report is prepared and closure process is done.

# ETL Testing is categorized into four different Types

1. New Data Warehouse Testing

2. Migration Testing

3. Change Request

4. Report Testing

## 1. New Data Warehouse Testing

- New DW is built and verified from scratch.
- Data input is taken from customer requirements and different data sources and new data warehouse is build and verified with the help of ETL tools.

## 2. Migration Testing

- In this type of project customer will have an existing DW and ETL performing the job but they are looking to bag new tool in order to improve efficiency.

## 3. Change Request

- In this type of project new data is added from different sources to an existing DW.
- Also, there might be a condition where customer needs to change their existing business rule or they might integrate the new rule.

## 4. Report Testing

- Reports are the end result of any Data Warehouse and the basic propose for which DW is build.
- Report must be tested by validating layout, data in the report and calculation.

# Basic ETL validations

***Source layer validations :-***

Source Type : - File

1) Verify the source file name
2) Verify the flat file coma separated (CSV) or pipe delimited
3) Verify the number of attributes present in detail records
4) Validate detail record attributes data type and length as per source mapping document
5) Verify total detail records count
6) Verify header record values

   For E.g :- File name, created date
7) Verify the trailer record values

   trailer which contains summary information for all the items
8) Verify the quality of data
9) Check for the duplicate records in the file

# Basic ETL validations

***Source layer validations :-***

Source Type : - Database

1. Verify the table name
2. Verify all the columns should present in the table which need to pull from source to stage table
3. Verify the column data type and length
4. Verify the quality of data
5. Verify the date format if the column data type is date
6. Verify the target table column data length should not be less than source table data length
7. Check for the duplicate records in the source table

# Basic ETL validations

***ETL layer validations :- Staging Layer validation***

Source Type : - Flatfile

1. Verify the location of path where staging files are saved.
2. Verify the staging file name.
3. Verify stage file type i.e pipe delimited or comma separated.
4. Verify workflow log and session logfile is saved in the form of a file at the location that are defined in the session's properties.
5. Verify the parameter file path.
6. Verify the variables in parameter file.
7. Check for the rejected records and rejection reason in the rejected record file.

If flat file is having Detail and trailer records

If detail records row count matches with trailer row value count then start the staging process

else

send failure email and stop the process

# Basic ETL validations

## History layer validation:-

Data completeness testing :- To verify the all the expected data loaded in target from the source.

Duplicate Check Testing :- Duplicate test is to ensure that there are no duplicate values for unique columns.

Data transformation testing :- Testing data transformation is done as in many cases it cannot be achieved by writing one source SQL query and comparing the output with the target. Multiple SQL queries may need to be run for each row to verify the transformation rules.

Data quality test :- Data quality test includes number check, date check, data check, null check etc

Incremental ETL testing :- This testing is done to check the data integrity of old and new data with the addition of new data. Incremental testing verifies that the inserts and updates are getting processed as expected during incremental ETL process.

Verify the data retention logic. How many years of history needs to save in the database

Verify SCD type load in the history table :-

Type 1 - Overwriting the old value

Type 2 - Creating a new additional record

Type 3 - Adding a new column

# Basic ETL validations

## Data mart layer validation

1) Verify the records in dimension table

2) Verify the record in fact table

3) Verify star schema or snow flake schema  used to build data mart.

4) Verify the data quality and Data validation

3) Validate constraint testing  :- Check the below constraints applicable to the particular table

a) NOT NULL
b) UNIQUE
c) Primary Key
d) Foreign key
e) Check
f) Default
g) NULL

# Basic ETL validations

## Report layer validation

1) Verify the number of records displayed are correct and no records are missing.

2) Verify Data Integrity. i.e. correct data is displayed on the report.

3) Verify if all the calculations on reports are correct (for eg, Total, Average, Max, Min)

4) If report have Summary and Details Tabs, verify if the data is matching and so all the calculations.

5) Verify if data satisfy the filter conditions ( for. Eg. Date Range, IDs etc..)  correctly

6) Verify if all the graphs are drawn correctly with data.

7) Verify how much time report takes to generate and its not exceeding time given in requirement.

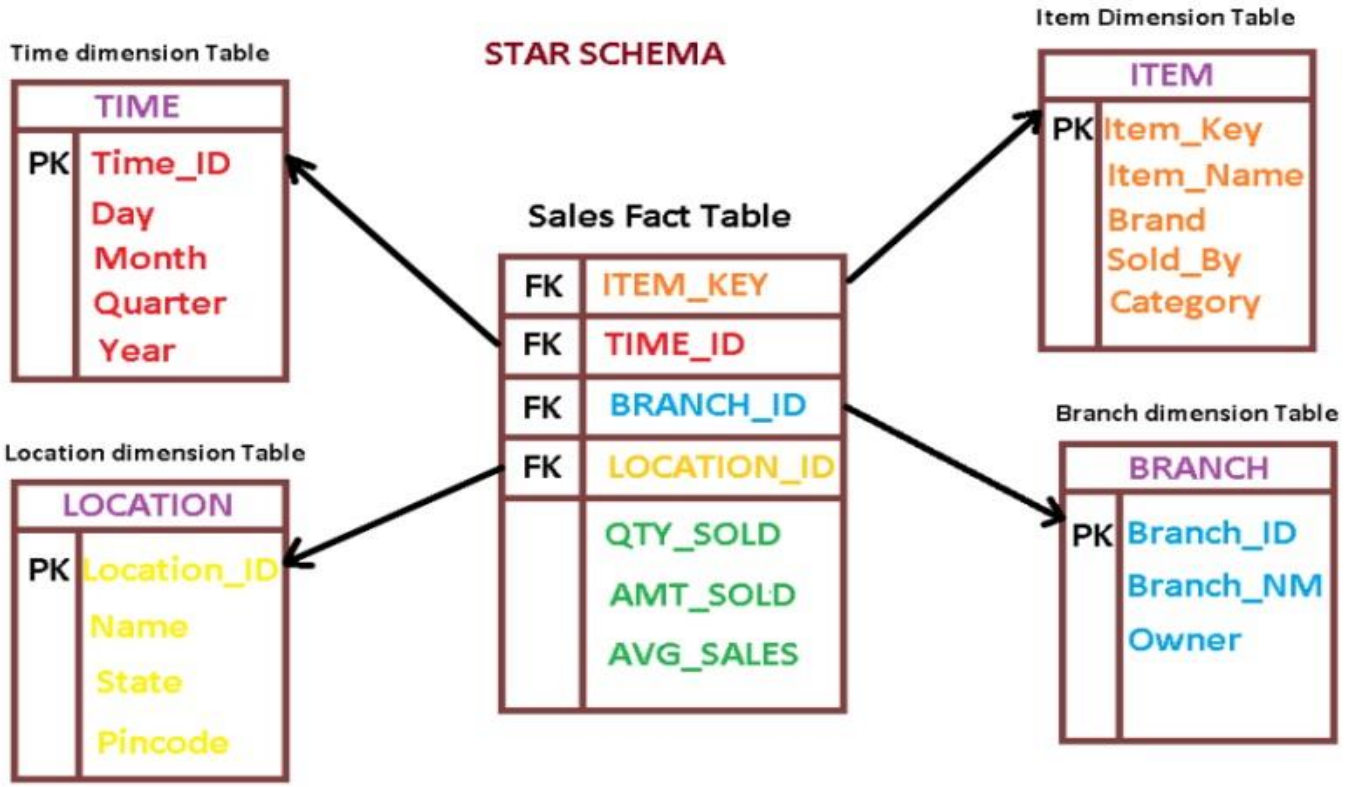
8) Verify how much time it takes to download.

9) Verify if it takes reasonable time to export in supported format.

# Basic ETL validations

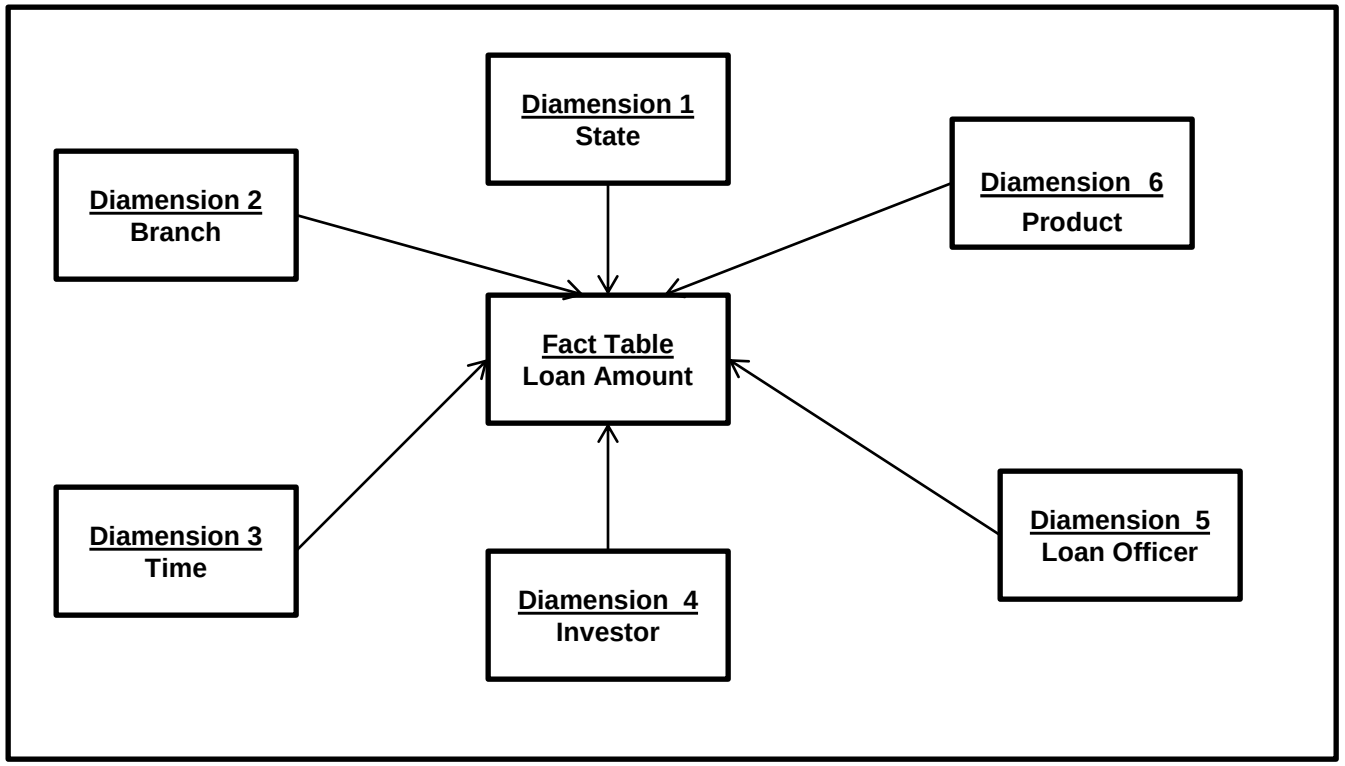
Dimension Table:- Dimension tables provides descriptive information for all the measurements recorded in fact table

Fact table :- It contains all the primary keys of the dimension and associated facts or measures(is a property on which calculations can be made) like quantity sold, amount sold and average sales.

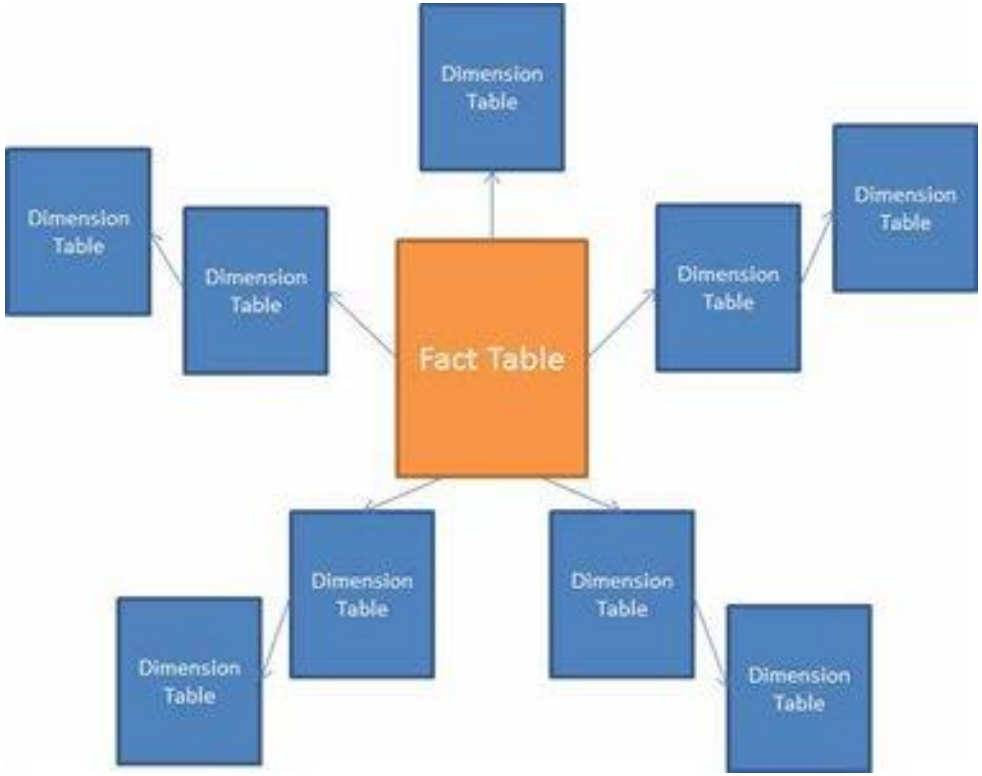


# ETL Testing

Star schema



# Snow flake schema



# ETL Tools

**Informatica PowerCenter :-** 


Informatica PowerCenter is a mature, feature-rich enterprise data integration platform for ETL workloads.

**AWS Glue :-** 


AWS Glue is a fully managed ETL service from Amazon Web Services that is intended for big data and analytic workloads.

**Alooma :-** alooma

Alooma is an ETL data migration tool for data warehouses in the cloud.

2019 Google acquired Alooma and restricted future signups only to Google Cloud Platform users

# ETL Tools

**Talend :-**


Talend Data Integration is an open-source ETL data integration solution. The Talend platform is compatible with data sources both on-premises and in the cloud, and includes hundreds of pre-built integrations.

Talend has received an average rating of 4.0 out of 5 stars on G2, based on 47 reviews. In addition, Talend has been named a “Leader” in the 2019 Gartner Magic Quadrant for Data Integration

# ETL RDBMS DBs

## Oracle :-

Oracle developed by Oracle Corporation is the most popular relational database system (RDBMS). Not only Oracle is a RDBMS, but also provides functionality for Cloud, Document Store, Graph DBMS, Key-value storage, BLOG, and PDF Storages. Recently. Oracle just announced autonomous feature that allows database to be intelligent and self-managed.
The current version of Oracle Database is 18c

## MySQL :-

MySQL is world's most popular database that is open source and free. MySQL was acquired by Oracle as a part of Sun Microsystems acquisition in 2009.

Key properties of MySQL: -

- MySQL is a database management system.
- MySQL databases are relational.
- MySQL software is Open Source.
- The MySQL Database Server is very fast, reliable, scalable, and easy to use.
- MySQL Server works in client/server or embedded systems.

# ETL DBs

## SQL Server :-

- SQL Server database developed by Microsoft is one of the most popular databases in the world.
- SQL Server is also a part of Microsoft's Azure cloud as Azure SQL Server
- The current version of SQL Server is SQL Server 2019.

## PostgreSQL :-

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads.

## IBM DB2 :-

IBM Db2 database is a relational database that delivers advanced data management and analytics capabilities for your transactional and warehousing workloads. This operational database is designed to deliver high performance, actionable insights, data availability, and reliability, and it is supported across Linux, Unix, and Windows operating systems.

# ETL DBs

## Teradata : -

Teradata is one of the popular Relational Database Management System. It is mainly suitable for building large scale data warehousing applications. Teradata achieves this by the concept of parallelism. It is developed by the company called Teradata.

# Reporting Tools

## Power BI for Office 365 –

*sporting hundreds of data visualizations, built-in AI capabilities, Excel integration, and custom data connectors.*

## Tableau –

*Tableau reporting tools are interactive with visual insights designed to combine, shape, clean and operationalize data flows.*

## IBM Cognos Analytics :-

- Cognos Analytics is an AI-fueled business intelligence platform that supports the entire analytics cycle, from discovery to operationalization.
- Visualize, analyze and share actionable insights about your data with anyone in your organization.
- Deploy where and when you need it with support for multicloud environments – public, private, on premise, and on IBM Cloud Pak™ for Data

# Principles of ETL Testing

1. Understand a source-target mapping document
2. Determine Different Testing Methods
3. Validate For Duplicate Check Testing
4. Understand Incremental, Full and Historical Load Testing
5. Understand Report Testing using Drill Down / Drill Through

# 1. Understand  Source-Target Mapping Document

The ETL mapping document contains the source, target and business rules information.

**A typical mapping document should contain the following information:**

1) Mapping indicator(Values A:ADD, D:Delete, C:Change)
2) Change description (Used to indicate mapping changes)
3) Key Indicator( Indicates whether the field is Primary key or not)
4) Source Table/File Name
5) Source Field Name
6) Source Field Data Type
7) Source Field Length
8) Source Field Description(The description will be used as a meta data for end user)
9) Business Rule
10) Target Table Name
11) Target Field Name
12) Target Data Type
13) Target Field Length
14) Comment

# Here is how the mapping will look like –

| Mapping Indicator | Change Description | Key Indicator | Source Table/File Name | Source Field name | Source Field Length | Source Data Type | Source Field Description | Business Rules | Target Table Name | Target Field Name | Target Data Type | Target Filed Length | Comment |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | | - | Emp.csv | First Name | 50 | Text | Emp First | Direct mapping | Emp_Dim | Emp_Fname | Varchar | 250 | |
| A | | - | Emp.csv | Last Name | 50 | Text | Emp Last | Direct mapping | Emp_Dim | Emp_Lname | Varchar | 250 | |
| A | | Primary | Emp.csv | Emp No | 10 | Varchar | Emp id | Direct mapping | Emp_Dim | Emp_ID | Varchar | 10 | |
| A | | - | Emp.csv | Department | 50 | Text | Emp Department Name | IF Emp Dep = "Sales" then load as "S"<br>Emp Dep = "Paking" then load as "P"<br>Emp Dep = "Transport" then load as "T" | Emp_Dim | Emp_Dept | Varchar | 250 | |

# 2. Determine Different Testing Methods

### 2.1) Constraint Testing:

In the phase of constraint testing, we identify whether the data is mapped from source to target or not using below scenarios –

***1. Not Null –***

The NOT NULL constraint enforces a field to always contain a value. This means that you cannot insert a new record, or update a record without adding a value to this field.

E.g.

```
CREATE TABLE Persons (
    P_ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Address varchar(255),
    City varchar(255)
);
```

```
INSERT INTO DBO.PERSONS (P_ID,LastName,FirstName,Address,City)
VALUES
(1000,NULL,NULL,'WHITEFILED','Bangalore')
```

# 2. Determine Different Testing Methods

**Update Statement :-**

```
UPDATE DBO.PERSONS
SET LastName =NULL
WHERE P_ID =1
```

**Select statement :-**

```
SELECT LastName, FirstName, Address FROM Persons
WHERE Address IS NULL

SELECT LastName, FirstName, Address FROM Persons
WHERE Address IS NOT NULL
```

## *2. Unique -*

- The UNIQUE constraint uniquely identifies each record in a database table.
- The UNIQUE and PRIMARY KEY constraints both provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint defined on it.
- Note that you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

E.g.

```
CREATE TABLE Personsunique
(
P_Id int UNIQUE,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

## *2. Unique -*

Select statement to identify Unique values:-

```
select P_ID,COUNT(*) FROM Personsunique
GROUP BY P_ID
having count(*) > 1
```

Unique key allow to have single NULL value

```
INSERT INTO [dbo].[Personsunique]
           ([P_Id]
           ,[LastName]
           ,[FirstName]
           ,[Address]
           ,[City])
     VALUES
           (NULL,'RAM','KOMAL','Irving,IL,62051','Texas')
```

## *2. Unique -*

```
INSERT INTO [dbo].[Personsunique]
           ([P_Id]
           ,[LastName]
           ,[FirstName]
           ,[Address]
           ,[City])
     VALUES
           (5,'RAM','KOMAL','Irving,IL,62051','Texas’)

INSERT INTO [dbo].[Personsunique]
           ([P_Id]
           ,[LastName]
           ,[FirstName]
           ,[Address]
           ,[City])
     VALUES
           (NULL,'MADHAV','Guru','Irving,IL,62051','BANGALORE’)
```

### *3. Primary Key -*

- The PRIMARY KEY constraint uniquely identifies each record in a database table.
- Primary keys must contain unique values.
- A primary key column cannot contain NULL values.
- Most tables should have a primary key, and each table can have only ONE primary key.

E.g.

```
CREATE TABLE PersonsPrimary
(
P_Id int NOT NULL,
FirstName varchar(255),
City varchar(255),
PRIMARY KEY (P_Id)
)
```

***3. Primary Key -***

1) primary key should be unique values

```
select P_ID,COUNT(*) FROM PersonsPrimary
GROUP BY P_ID
HAVING COUNT(*) > 1
```

2) Primary key should not have null values

```
  select * from PersonsPrimary
  where P_ID IS NULL
```

3) SELECT * FROM PersonsPrimary

4) UPDATE PERSONSPRIMARY

```
  SET P_ID =NULL
  WHERE P_ID =1
```

***3. Primary Key -***

```
5) INSERT INTO [dbo].[PersonsPrimary]
        ([P_Id]
        ,[FirstName]
        ,[City])
    VALUES
        (NULL,'RAJESH','Bangalore')
```

## *4. Foreign key -*

A FOREIGN KEY in one table points to a PRIMARY KEY in another table.

The "People" table:

| Id | FirstName | Last_Name | CONTACT_PHONE |
|---|---|---|---|
| 100 | Amit | Joseph | 883746922929 |
| 200 | Bhavin | Alstorm | 633837793729 |
| 300 | Divya | Bolden | 6782318921892 |

The “Credit_Card" table:

| Card_Id | Card_Number | Peo_Id |
|---|---|---|
| 1 | 52368 | 100 |
| 2 | 44678 | 200 |
| 3 | 12551 | 300 |

The “ID column in the "People" table is the PRIMARY KEY in the "Persons" table.

The "Peo_Id" column in the “Credit_card" table is a FOREIGN KEY in the "Orders" table.

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

```
CREATE TABLE Orders
(
O_Id int NOT NULL PRIMARY KEY,
OrderNo int NOT NULL,
P_Id int FOREIGN KEY REFERENCES Persons(P_Id)
)
```

## *5. Default –*

- The DEFAULT constraint is used to insert a default value into a column.

- The default value will be added to all new records, if no other value is specified.

E.g.

```
CREATE TABLE Personsdef
(
P_Id int NOT NULL,
FirstName varchar(255),
City varchar(255) DEFAULT 'London'
)

INSERT INTO [dbo].[Personsdef]
           ([P_Id]
           ,[FirstName]
           )
     VALUES
           (2,'Martin')
```

## 2.2) **Source to Target Count Testing**:

E.g..
SELECT COUNT(*) FROM table_name;

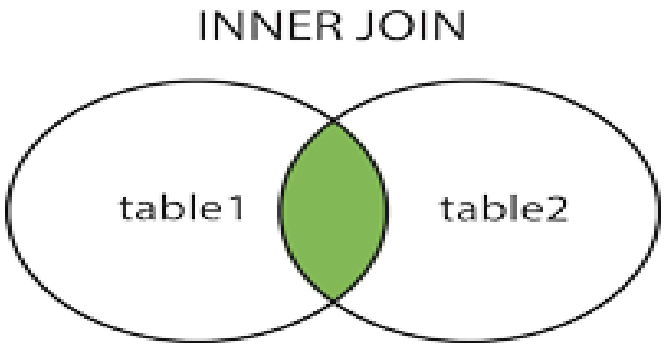
Source Database :- TRAVEL
Select count(*) from People
RowCount =1418

Target Database :- TRAVEL_E
Select count(*) from people
RowCount =1513

## 2.4) Using different Joins –

➢INNER JOIN:

- Returns all rows when there is at least one match in BOTH tables
- The INNER JOIN keyword selects records that have matching values in both tables




```
Select
P.ID,P.FIRST_NAME,P.LAST_NAME,P.EMAIL,C.CARD_NUMBER,C.EXPIRATION_DA
TE
FROM people p
INNER JOIN CREDIT_CARDS C
ON P.ID =C.PEO_ID
```

## 2.4) Joins –

- INNER JOIN on more than 2 tables :-

Tables :- People
ITINERARIES
HOTEL_BOOKINGS

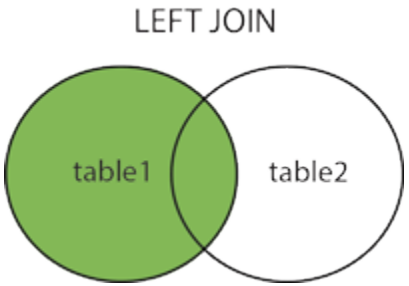
SQL :-

```
Select P.ID,P.FIRST_NAME,P.LAST_NAME,P.EMAIL
FROM ((people p
INNER JOIN ITINERARIES IT ON P.ID =IT.PEO_ID)
INNER JOIN HOTEL_BOOKINGS HB ON IT.ID = HB.ITN_ID)
```

## 2.4) Using different Joins –

- LEFT OUTER JOIN: Return all rows from the left table, and the matched rows from the right table

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

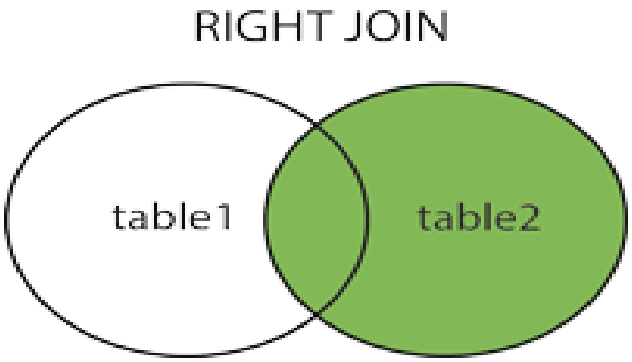



```
Select P.ID,P.FIRST_NAME,P.LAST_NAME,P.EMAIL,C.CARD_NUMBER,C.EXPIRATION_DATE
FROM people p
LEFT JOIN CREDIT_CARDS C
ON P.ID =C.PEO_ID
```

## 2.4) Using different Joins –

- RIGHT OUTER JOIN: Return all rows from the right table, and the matched rows from the left table

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.




```
Select P.ID,P.FIRST_NAME,P.LAST_NAME,P.EMAIL,C.CARD_NUMBER,C.EXPIRATION_DATE
FROM PEOPLE P
RIGHT JOIN CREDIT_CARDS C
ON C.PEO_ID = P.ID
```

In the credit Card table id > 2800 records are not present in People table

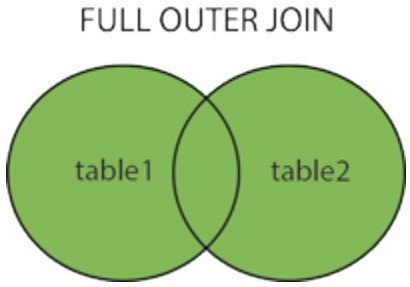
## 2.4) Using different Joins –

- FULL OUTER JOIN: Return all rows when there is a match in ONE of the tables

- The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.

**SQL :-**

```
Select P.ID,P.FIRST_NAME,P.LAST_NAME,P.EMAIL,C.CARD_NUMBER,C.EXPIRATION_DATE
FROM PEOPLE P
FULL OUTER JOIN CREDIT_CARDS C
ON C.PEO_ID = P.ID
```



## UNION Operator

- The UNION operator is used to combine the result-set of two or more SELECT statements.
- Each SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement must also be in the same order

Syntax :-

SELECT *column_name(s)* FROM *table1*
UNION
SELECT *column_name(s)* FROM *table2*;

Select FIRST_NAME,LAST_NAME from people

UNION

SELECT FIRST_NAME,LAST_NAME FROM PEOPLE_PET

ORDER BY FIRST_NAME,LAST_NAME

Union selects only distinct values from the two tables

## UNION ALL Operator

- Use UNION ALL to select duplicate values
- The UNION operator is used to combine the result-set of two or more SELECT statements.
- Each SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement must also be in the same order

Syntax :-

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

```
Select FIRST_NAME,LAST_NAME from people
UNION ALL
SELECT FIRST_NAME,LAST_NAME FROM PEOPLE_PET
ORDER BY FIRST_NAME,LAST_NAME
```

## 2.5) Minus Queries –

DESCRIPTION:

-The SQL MINUS operator is used to return all rows in the first SELECT statement that are not returned in the second SELECT statement.

-Each SELECT statement within the MINUS query must have the same number of fields in the result sets with similar data types.

Oracle SYNTAX-

```
SELECT expression1, expression2, ... expression_n
FROM tables
MINUS
SELECT expression1, expression2, ... expression_n
FROM tables;
```

e.g.

Table :- PET1

| PET1 | | | | |
|---|---|---|---|---|
| ID | DESIGNATION | FIRST_NAME | LAST_NAME | EMPNO |
| 1 | c1 | John | mahadevan | 10 |
| 3 | c2 | RAM | TOMMY | 10 |
| 2 | c3 | JOHN | MAHESH | 10 |

Table :- PET2

| PET2 | | | | |
|---|---|---|---|---|
| ID | DESIGNATION | FIRST_NAME | LAST_NAME | EMPNO |
| 1 | c1 | John | mahadevan | 10 |
| 2 | c4 | ismail | raj | 10 |

```
SELECT ID,DESIGNATION,FIRST_NAME,LAST_NAME,EMPNO from PET1
EXCEPT
SELECT ID,DESIGNATION,FIRST_NAME,LAST_NAME,EMPNO from PET2
```

Result Set :-

| ID | DESIGNATION | FIRST_NAME | LAST_NAME | EMPNO |
|---|---|---|---|---|
| 2 | c3 | JOHN | MAHESH | 10 |
| 3 | c2 | RAM | TOMMY | 10 |

# 3. Validate For Duplicate Check Testing

In this phase of ETL Testing, a Tester can face duplicate value very frequently so, at that time the tester follows database queries why because huge amount of data is present in source and Target tables.

```
SELECT ID,FIRST_NAME,LAST_NAME,COUNT(*) COUNT
FROM PET3
GROUP BY ID,FIRST_NAME,LAST_NAME
HAVING COUNT(*) > 1
```

Note:

1) There are mistakes in Primary Key or no Primary Key is allotted then the duplicates may arise.
2) Sometimes, a developer can do mistakes while transferring the data from source to target at that time duplicates may arise.
3) Due to Environment Mistakes also duplicates arise .

# 4. Understand  Incremental, Full and Historical Load Testing

- Full load is like starting fresh....The first time when you load data into DW using any ETL tool is Full Load.
- Delta/Incremental Load is loading only that data which is modified after Full load. It may be hourly/Daily/Weekly/Monthly load.

- **Slowly Changing Dimension (SCD)** is a dimension that stores and manages both current and historical data over time in a data warehouse.

- There are 3 types of SCD as below :

a) **SCD Type 1**
b) **SCD Type 2**
c) **SCD Type 3**

# SCD Type 1:

This method overwrites the old data in the dimension table with the new data.

E.g.

As an example, I have the customer table with the below data.

| customer_id | customer_name | Location |
|---|---|---|
| 1 | Smitha | London |

Here the customer name is miss spelt. It should be Smith instead of Smitha. If you use type1 method, it just simply overwrites the data. The data in the updated table will be.

| customer_id | customer_name | Location |
|---|---|---|
| 1 | Smith | London |

When to use –

SCD type 1 methodology is used when there is no need to store historical data in the dimension table.

It is used to correct data errors in the dimension.

Advantages –

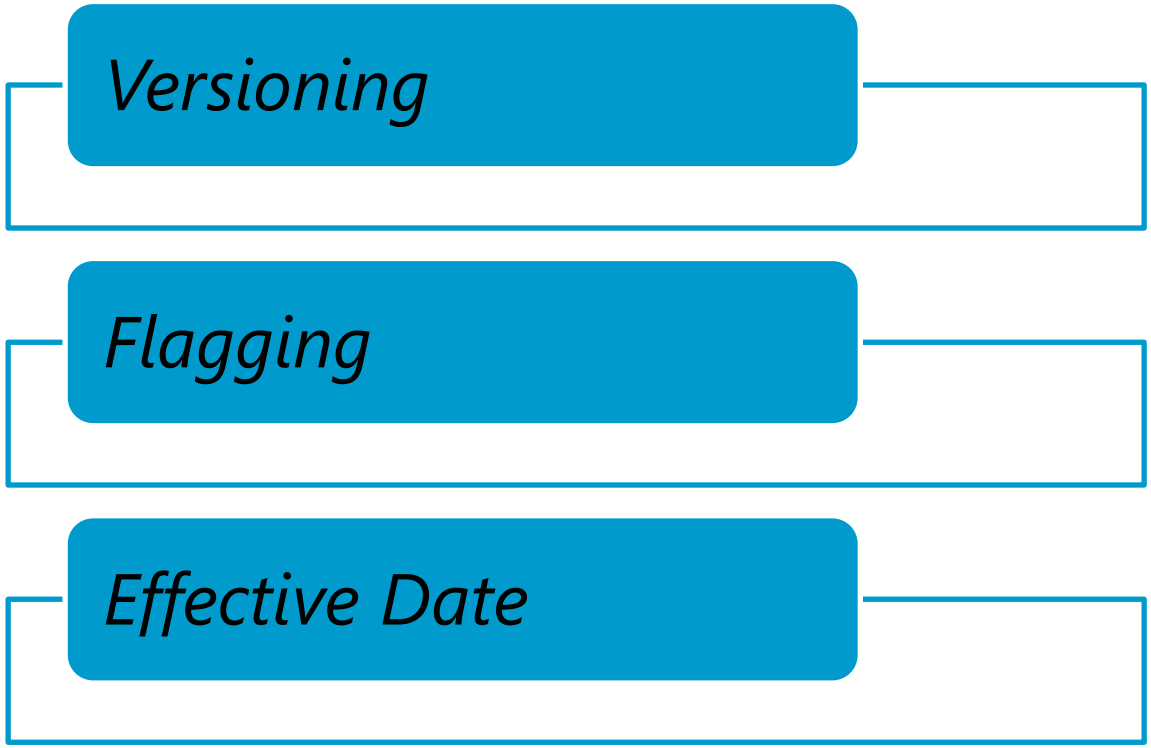
Since there is no need to track old values, easy to handle and less space occupied.

Disadvantages –

It is not possible to track history.

## SCD Type 2:

SCD type 2 stores the entire history the data in the dimension table. With type 2 we can store unlimited history in the dimension table. In type 2, you can store the data in three different ways. They are:



## SCD Type 2 Versioning:

In versioning method, a sequence number is used to represent the change.

The latest sequence number always represents the current row and the previous sequence number will represent old data.

As an example, let's use the same example of customer who changes the location.

Initially the customer is in London location and the data in dimension table will look as.

| surrogate_key | customer_id | customer_name | Location | Version |
|---|---|---|---|---|
| 1 | 1 | Smith | London | 1 |

The customer moves from London to Seattle and the version number will be incremented.

The dimension table will look as –

| surrogate_key | customer_id | customer_name | Location | Version |
|---|---|---|---|---|
| 1 | 1 | Smith | London | 1 |
| 2 | 1 | Smith | Seattle | 2 |

Now again if the customer is moved to another location, a new record will be inserted into the dimension table with the next version number.

**SCD Type 2 Flagging**:

In flagging method, a flag column is created in the dimension table. The current record will have the flag value as 1 and the previous records will have the flag as 0.

Now for the first time, the customer dimension will look as-

| surrogate_key | customer_id | customer_name | Location | flag |
|---|---|---|---|---|
| 1 | 1 | Smith | London | 1 |

Now when the customer moves to a new location, the old records will be updated with flag value as 0 and the latest record will have the flag value as 1.

| surrogate_key | customer_id | customer_name | Location | Flag |
|---|---|---|---|---|
| 1 | 1 | Smith | London | 0 |
| 2 | 1 | Smith | Seattle | 1 |

**SCD Type 2 Effective Date**:

In Effective Date method, the period of the change is tracked using the start_date and end_date columns in the dimension table.

| surrogate_key | customer_id | customer_name | Location | Start_date | End_date |
|---|---|---|---|---|---|
| 1 | 1 | Smith | London | 01-Mar-2013 | 20-Feb-2014 |
| 2 | 1 | Smith | Seattle | 21-Feb-2014 | NULL |

The NULL in the End_Date indicates the current version of the data and the remaining records indicate the past data.

**SCD Type 3**:

In type 3 method, only the current status and previous status of the row is maintained in the table. To track these changes two separate columns are created in the table.

| surrogate_key | customer_id | customer_name | Current_Location | Previous_location |
|---|---|---|---|---|
| 1 | 1 | Smith | London | NULL |

Let say, the customer moves from London to Seattle and the updated table will look as

| surrogate_key | customer_id | customer_name | Current_Location | previous_location |
|---|---|---|---|---|
| 1 | 1 | Smith | Seattle | London |

Now again if the customer moves from Seattle to New York, then the updated table will be

| surrogate_key | customer_id | customer_name | Current_Location | previous_location |
|---|---|---|---|---|
| 1 | 1 | Smith | New York | Seattle |

The type 3 method will have limited history and it depends on the number of columns you create.

## *Thus, ETL Testing mainly consists of below –*

- Validation for job/sequence ETL completion
- Log files validation
- Data validations for -
  - ✓ Source table to Target table
  - ✓ Source table to Target file
  - ✓ Source file to Target table
- Lookup logic validation
- Validations for data extraction, data cleansing, data consolidation, merging, splitting, aggregations, data loads
- Source & Target data comparison
- Validation of business logic using SQL scripting
- Data validation on reports