Disclaimer: These slides are copyrighted and strictly for personal use only

- This document is reserved for people enrolled into the Ultimate Big Data Masters Program (Cloud Focused) by Sumit Sir

- **Please do not share this document,** it is intended for personal use and exam preparation only, thank you.

- If you've obtained these slides for free on a website that is not the course's website, please reach out to legal@trendytech.in . Thanks!

- All the Best and Happy Learning!

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

Data Modeling

Is a process of structuring the data internally in a system to meet the requirements in the best possible way.

It involves identifying the different types of data that will be stored and specifying the relationships between them. It also includes incorporating rules and constraints for data integrity and consistency.

The Goal of modelling data is to help achieve the end user needs in a most efficient manner.

Data can be modelled based on the needs of the end users (different perspectives of the user needs) :

1. **Type of the system to be modelled -**

Databases are a good fit for Online Transactional Processing / OLTP systems. Whereas Data warehouses are a best fit for Online Analytical Processing / OLAP systems. Systems are modelled based on the requirements.

2. **Type of Consumer -**

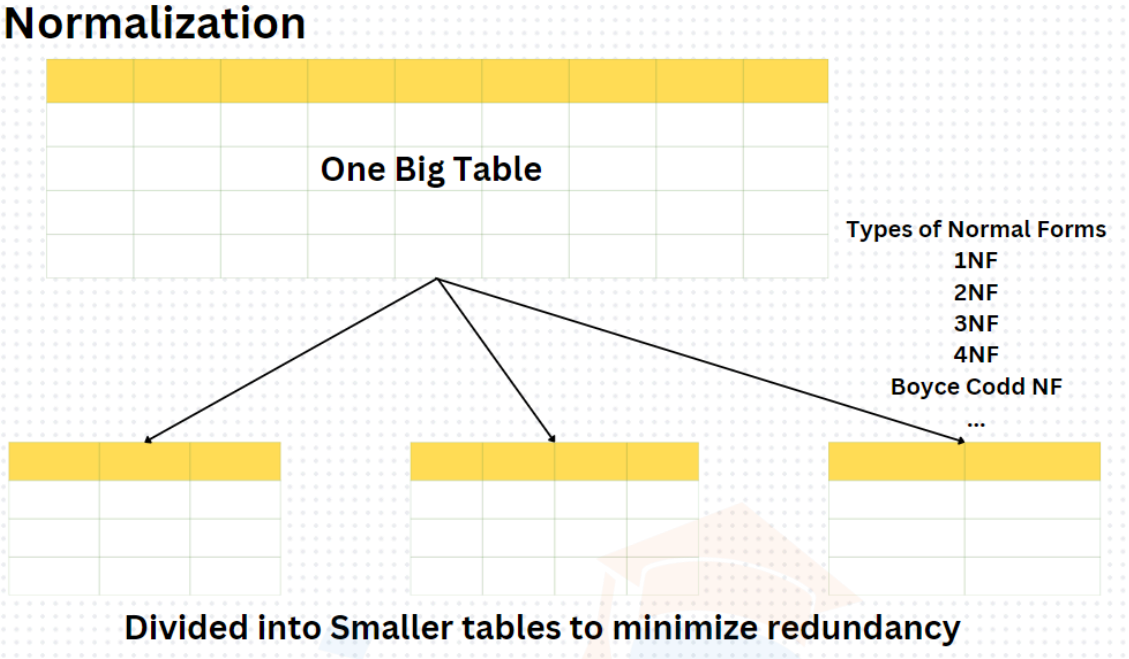
Data Analysts Vs Data Engineers require different complexity of tables to be designed for easy and optimized querying of data respectively.

- Data Analyst would require less complex structures that could enable them to access the data without having to write complicated queries.
- Data Engineers on the other hand would be able to write complex queries and handle complex structures to get optimized results.

NOT FOR DISTRIBUTION ©Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

Normalization



Normalization is a technique to reduce one large table into multiple small tables with the primary goal of minimizing redundancy.

The different normalization techniques used to divide the large table into smaller tables are :

1NF (First Normal Form)

Different Criterias to be met for the table to be in 1NF :

- Single cell should hold only one value (Atomicity)
- There should be a primary key to identify each row uniquely.
- No Duplicate Rows / Columns

Not in 1NF

Student ID	Courses
1	Math, English, Music
2	Science, History

In 1NF

Student ID	Courses
1	Math
1	Music
1	English
2	Science
2	History

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
© 2023 All Rights Reserved Worldwide

2NF (Second Normal Form)

Different Criterias to be met for the table to be in 2NF :

- Table should be in 1NF
- Non primary key attributes of the table should depend on the complete candidate key.

CAND_ID	SUBJECT_NO	SUBJECT_FEE
111	S1	1000
222	S2	1500
111	S4	2000
444	S3	1000
444	S1	1000
222	S5	2000

Not in 2NF

CAND_ID	SUBJECT_NO
111	S1
222	S2
111	S4
444	S3
444	S1
222	S5

In 2NF

SUBJECT_NO	SUBJECT_FEE
S1	1000
S2	1500
S3	1000
S4	2000
S5	2000

3NF (Third Normal Form)

Different Criterias to be met for the table to be in 3NF :

- Table should be in 2NF
- No transitive dependencies (The non-primary key columns should not have dependencies among themselves)

Grades Table

STUDENT_ID	COURSE	INSTRUCTOR	INSTRUCTOR_OFFICE
1	Math	Prof. A	Room 101
1	English	Prof. B	Room 102
2	Science	Prof. C	Room 103

Not in 3NF

STUDENT_ID	COURSE
1	Math
1	English
2	Science

COURSE	INSTRUCTOR
Math	Prof. A
English	Prof. B
Science	Prof. C

INSTRUCTOR	INSTRUCTOR_OFFICE
Prof. A	Room 101
Prof. B	Room 102
Prof. C	Room 103

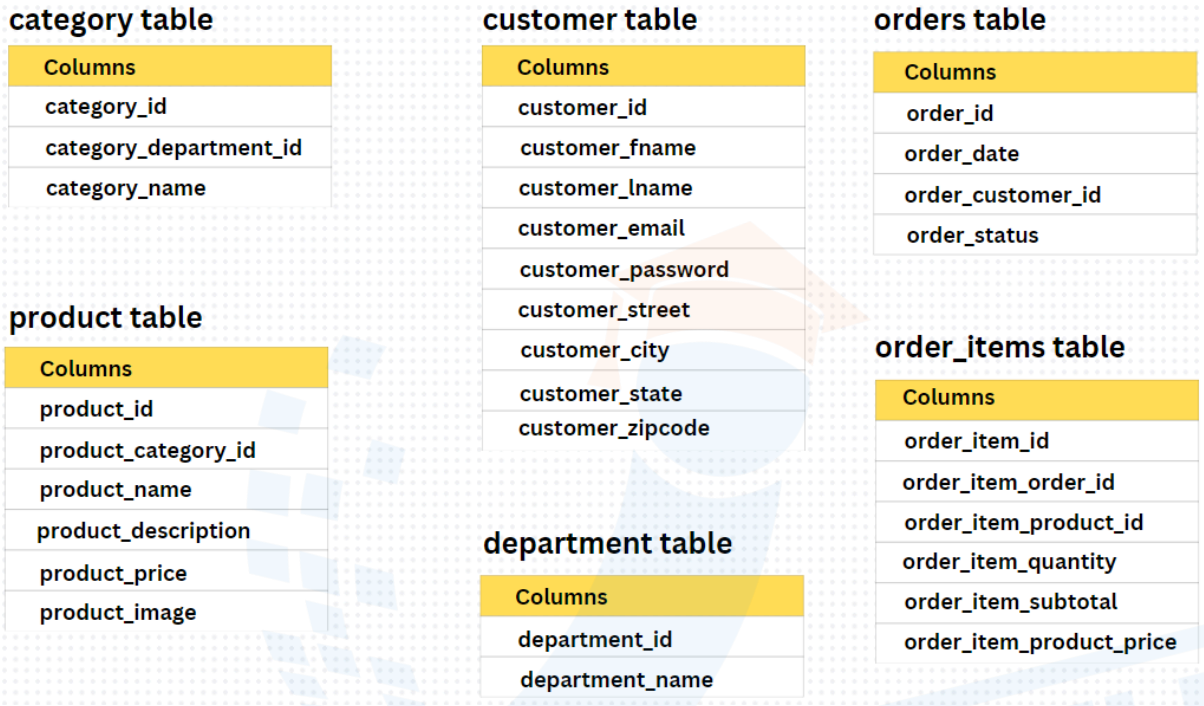
In 3NF

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

What is the need for dividing one large table consisting of all the data, into smaller tables?

The product, department and the category table are all related to a product and could have been as one single large table. However, it is divided into smaller tables as given in the diagram below.



Lets understand what is the need to have smaller tables as depicted in the above scenario :

- If we have one large table, then certain columns like Department and Category would repeat multiple times leading to data redundancy. In order to minimize data redundancy and optimize the structuring of data so that there would be no scope of inconsistent systems, the large table is divided into smaller tables.

Note:

What are the drawbacks of data redundancy?

- Storage requirements and thereby the cost will be more as the same data is repeated multiple times.
- Data Writes/Updates can be complicated (needs to be modified at multiple places because of redundancy) and could lead to inconsistent systems if the data is not updated at all the points of replication.

Therefore, normalized tables would be a best fit for OLTP systems.

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

Modeling a Datawarehouse (DWH)

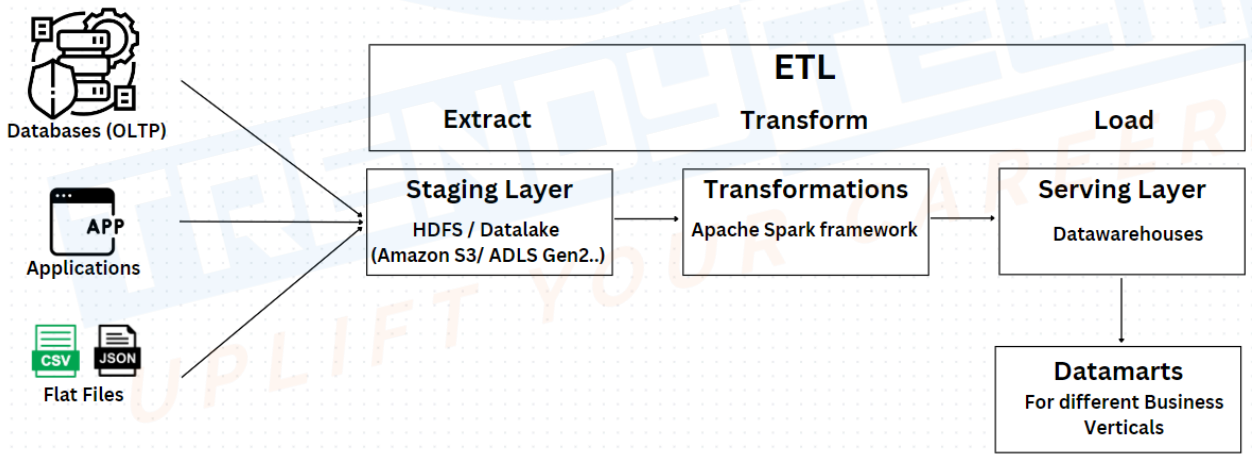
Can reporting be carried out with OLTP systems?

Yes, but not the optimal choice as OLTP systems are not meant for analysing large volumes of data. They are meant for handling transactional data. Here are the reasons why a OLTP system cannot be a good fit for reporting/ analysis purposes

- Since, Reporting would require a lot of data to get insights, it would involve a lot of join operations if it were to be an OLTP system (consists of Normalized small tables). Joins are costly operations.
- It could overload the OLTP system as it has to perform complex joins and analyze large volumes of data, thereby making it less performant even with the transactional tasks for which it is really meant for.

Datawarehouses DWH : (Online Analytical Processing - OLAP) are a best fit for such reporting or analysis purposes. **OLAP** systems are most performant and optimized for **Read** queries.

Sample Data Pipeline for Reporting Purpose



NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

Data Model in Data Warehouse

Dimensional Modelling - Is a design technique in databases intended to support end user queries in a DWH

Formal Definition - "Is a Design technique for databases intended to support end-user queries in a Datawarehouse" by Ralph Kimball

Process of modelling a business use-case into a series of facts and dimension tables for analysis.

Transactional DB Design Vs Reporting DB Design

Transactional DB Design :

- **Performance Focus :** Designed with a goal of fast data maintenance
- Quicker inserts and updates
- Small subsets of data is retrieved as a result of a query
- Data consistency is critical
- Follows laws of Normalization
- Focus here is to make the task easy for the customer who is performing writes / entering the data.

Reporting DB Design :

- Copy of the same transactional data is structured in a different way (in the form of fact and dimension tables)
- The intent of this design is to support the business queries and not meant for data maintenance.
- The resulting model reflects the kind of questions that the business wants to ask and not on the functions of the underlying operational system.
- Descriptive/supportive data like customer name, address, etc is separated from the actual factual or numeric data like order quantity, order amount, etc.
- **Performance Focus :** on retrieving the data quickly (faster reads)
- Larger datasets are retrieved as a result of a query.
- Insert and update speeds are not relevant.

NOT FOR DISTRIBUTION ©Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

Features of Dimensional Modelling

- Data Maintenance performance is secondary.
- Data is denormalized to support reporting.

  (Storage is much cheaper at this point in time and updates are not very frequent in this case.)

- Performance focus is on retrieving (read) the data quickly and less on data updates.

Fact and Dimension

Fact is a measurable metric.

Example : Order quantity, Order amount, Profit, etc.

Dimension is a supportive/additional property that enhances the fact.

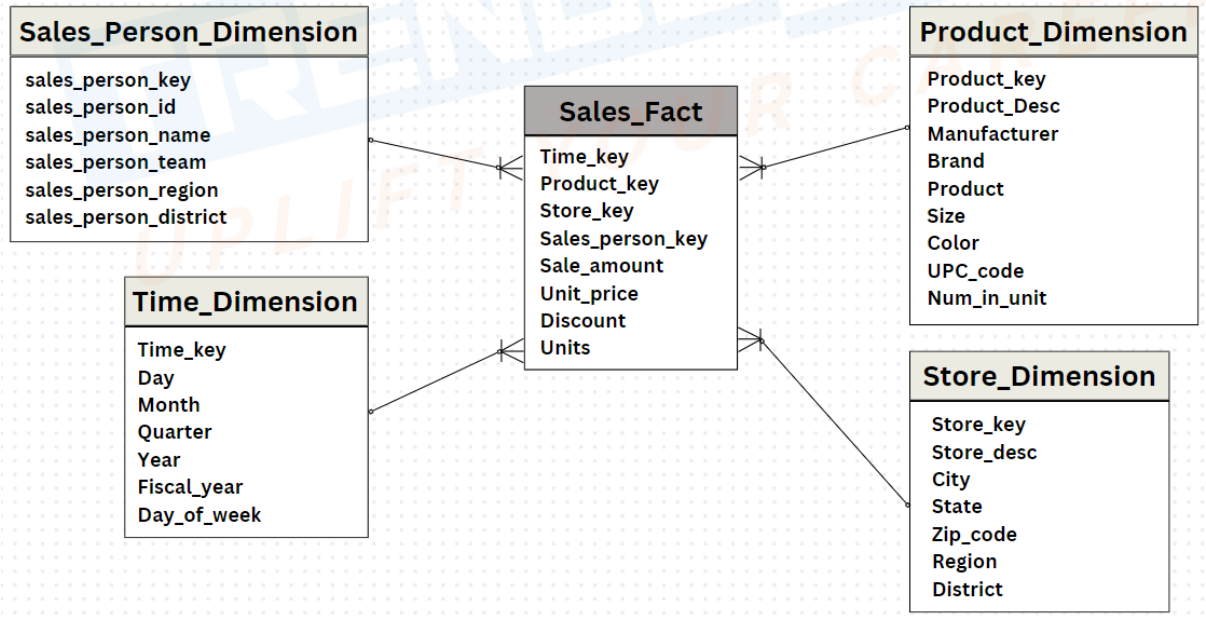
Example : Store, Customer, Product, etc.

In order to answer the business query, we can perform filter, sort, group,etc on dimensions like customer_number, store_number to get the final results.

Note:

- A decimal value mostly indicates a potential fact.
- A String value mostly indicates a potential dimension.

A real-time example of Fact and Dimension table - STAR SCHEMA



NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

Example Use-case :

- **A customer purchased an Iphone for $1000**

  In this case, **$1000** is a **FACT**

  The values of **WH questions** like - Who purchased it? In which store was it purchased? When was it purchased? Who was the sales person who made the sale? etc, will represent the **DIMENSIONS**

  Note:

  - The facts are generally very less compared to dimensions.
  - In case of Dimension modeling, a relationship exists only between the Fact and the Dimension table.
  - There is no connection between one Dimension table to another.
  - The dimension tables are denormalized tables.
  - In a Fact table, the no.of keys connecting to the different dimension tables are more than the actual fact values.
  - The Fact table will be the largest table consisting of huge volumes of data compared to any Dimension table. Fact table captures all the transactions and therefore keeps growing over time.

Surrogate Key

Surrogate keys are artificially generated keys (generally an integer) used by Datawarehouses to uniquely identify a row in a dimension table.

It is a good practice to not take the dimension keys from the source system but rather use the artificially generated surrogate keys as a primary key, because -

> There is a possibility that backend systems can change the data and therefore shouldn't depend on this changing data.

> Right approach is to have control over the key. (If the key is changed at the backend, it should not impact the tables and the join operations)

> Legacy keys (key from source system) can be stored in the Dimension table.

> The primary key should be different from the legacy and is a different column that is generated artificially.

> If there are multiple source systems, then there are chances of having the same keys or different key structures. This change in the key structure should not impact the tables and the operations.

NOT FOR DISTRIBUTION ©Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
© 2023 All Rights Reserved Worldwide

Note:

- Facts are the actual transactions that has taken place and will never change
- Whereas, dimensions can change slowly (Slowly Changing Dimensions SCD)
- Keys are very important to perform joins of tables. Since, the backend applications can change the data, it is important to have control over the keys.

Need of Surrogate Keys :

1. Required to maintain and implement the history of SCD (As the natural keys cannot be duplicated or repeated if it is a primary key)
2. Avoid conflicts among backend application keys if there are multiple source systems.
3. Insulates the datawarehouse from backend application changes (It would be difficult to perform joins if the keys change)

Snowflake Schema

In case of snowflake schema, the dimension tables are normalized (A single large Dimension table is divided into smaller tables)

Ex : Dimension-Product is divided into Dimension-Author | Dimension-Publisher | Dimension-Line

Snowflaking is a process of breaking one large dimension table into smaller related tables(relationship established using primary and foreign keys)

- When a Dimension relates to another dimension
- It would involve more joins and therefore causes a lot of performance issues.
- It is a good fit for OLTP systems.

Steps involved in Dimensional Modeling

1. Choose the business process. Ex : Model Sales

2. Declare the Grain / Granularity (is the level of detail that is required based on the requirement)

Ex : Consider the **orders table**

In this table, the grain can be declared at

a) **order** (In this case, the maximum details that can be accessed is only the total order amount and the no.of order_line_items)

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

b) **order_line_item** (In this case, you can more details about every single order_line_item as well)

3. Identify the Dimensions

4. Identify the Facts

Note : User stories are helpful in identifying which of the properties can be modelled as facts or as dimensions.

Slowly Changing Dimension SCD

SCD 0 - There are certain attributes of an entity that can never change like the Birth date, Birth place, etc. These attributes fall under SCD 0 category.

SCD 1 - In some scenarios, the value of an attribute is overwritten, the previous value is not retained. Such attributes where the history is not recorded, fall under SCD 1 category.

Advantage - It is easy to implement as the data needs to be just updated without having to record the previous value.

Disadvantage - The previous value of the attribute is lost while updating as the values are overwritten without capturing the history.

SCD 2 - This category maintains the complete history of the data changes. For every data update, a new row will be created. It is complex to implement SCD 2 as it involves addition and updation of multiple rows.

SCD 3 - This category maintains partial history. In this case, instead of a new row creation for updating the data, a new column is maintained in the same row to capture the current and the previous values of the updated attribute.

Easy to implement but has limited history, only one previous value is captured as history.

Note:

> SCD 1 implementation would be a good fit if the data is not critical, for which the history is required to be maintained.

> Based on the business requirement the respective SCD implementations can be adopted. It is always a good approach to maintain the complete history of the data as the business requirements can change over time and if the history is not captured, then it will not be possible to share the accurate data for analysis.

NOT FOR DISTRIBUTION ©Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
© 2023 All Rights Reserved Worldwide

Modeling and Optimizing Facts

- Can be tricky when compared to Dimension modeling because of the volume of data that gets accumulated over time.

In this scenario, one of the solutions is to choose the right Grain which will reduce the amount of data maintained.

- Fact table needs to be joined with Dimensions to achieve the desired results. This would involve **Joins** and joins is a costly operation as it is a **wide transformation** and involves data shuffling.

Therefore, it is always a good approach to pre-process(bucketing) facts and dimensions.

One Big Table (OBT)

One Big Table is a concept that has gained popularity in recent times. The idea is to store all the data in one single massive table.

One big table is a table consisting of all the attributes of an entity as columns. In this case, there is no fact and dimension table but rather a large table consisting of many columns.

Disadvantage :

- OBT will have a lot of redundancy
- Storage required would be more and thereby the storage cost will be higher.

Benefits :

- Joins can be completely eliminated.
- Simplified data model : It would be simpler to query a single table and no complex joins are required to get the desired data.
- Query performance will be much higher as it involves only a single table.
- Data modelling and Engineering efforts are cut down.
- Reduce development and maintenance efforts

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
© 2023 All Rights Reserved Worldwide

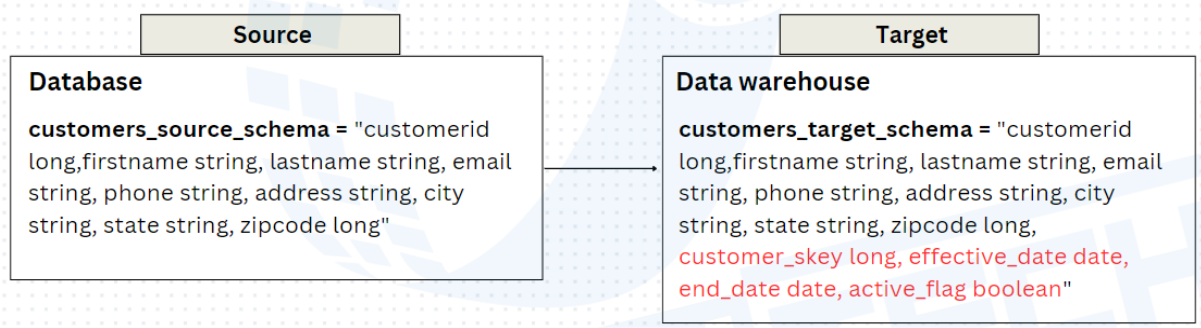
- Since the data is stored in a columns based file format, it would require relatively less storage than usual. Data updates would be complex in this format. However, such systems are mainly meant for reads and not write operations.

SCD Type 2 Implementation in Pyspark

SCD Type 2 retains the complete history of changes and is mostly used in the industry.

Example SCD Type 2 Implementations Steps

- Create a Directory Structure in HDFS
    1. Create a Main Folder called Demo which will contain subfolders - Source and Target
    2. Create a Source folder and add the customer.csv file to the source.
    3. Create a Target folder



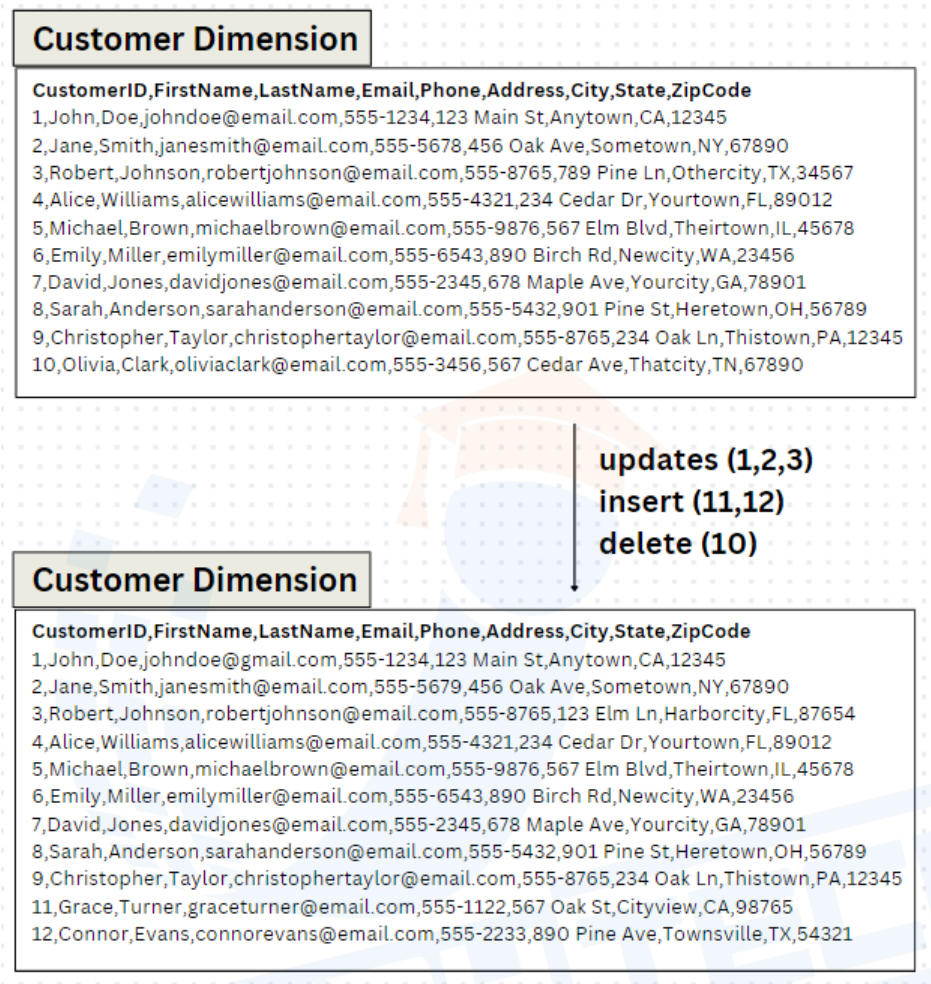
4. Perform the necessary transformations on the source file and add it to the Target. (new columns named customer_skey, effective_date, end_date, active_flag

```
enhanced_customers_source_df = spark.read \
.format("csv") \
.option("header",True) \
.schema(customers_source_schema) \
.load(source_url) \
.withColumn("customer_skey",row_number().over(window_def)) \
.withColumn("effective_date",date_format(current_date(),
DATE_FORMAT)) \
.withColumn("end_date",date_format(lit(future_date),
DATE_FORMAT)) \
.withColumn("active_flag", lit(True))
```

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

5. Now add the new_customers.csv to the source with the updated data. (This mimics the data changes that take place at the source location)



6. Now the source consists of the updated data and the target is still having the old data. In order to keep the target folder in sync with the source, **join** will be performed on the two tables.

7. Rename the columns for better differentiation between the source and target and add a new column for the **hash string.**

   Then perform a join on the source and target to check if the hash_md5 strings match. If they don't match, then some updates need to be done.

   ```
   customers_source_df_hash =

   column_renamer(get_hash(customers_source_df,
   slowly_changing_cols), suffix="_source", append=True)
   ```

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

After renaming the columns for Source :
| customerid_source | firstname_source | lastname_source | email_source | phone_source | address_source | city_source | state_source | zipcode_source | **hash_md5_source**
Likewise for Target :
| customerid_target | firstname_target | lastname_target | email_target | phone_target | address_target | city_target | state_target | zipcode_target | **hash_md5_target**

Key Points :

- Only the active records are required to be focussed on. The deleted records of the source data can be marked as inactive in the target
- If a certain record has nulls in the target but there are values present in the source for the same entry, this indicates that an **INSERT** should be performed.
- If a certain record has nulls in the source but there are values present in the target for the same entry, this indicates that a **DELETE** should be performed.
- **UPDATES -** All the keys that have the potential to slowly change need to be compared for any changes from source to target and needs to be updated accordingly in the target
- A better approach for updates, rather than comparing all the keys, is to use a hash approach (Take the hash of the keys which is a single bit string and match these strings. If there is a mis-match in the strings, then some of the values has to be updated)
- 4 Scenarios : **INSERT | DELETE | UPDATE | NO-CHANGE**

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide