Data Warehouse Assessment 1

1.

a) **Category of a product may change over a period of time. Historical category information (current category as well as all old categories) has to be stored. Which SCD type will be suitable to implement this requirement? What kind of structure changes are required in a dimension table to implement SCD type 2 and type 3.**

Ans:

Slowly Changing Dimensions (SCD): The Slowly Changing Dimension transformation coordinates the updating and inserting of records in data warehouse dimension tables.In a data warehouse environment, there may be a requirement to keep track of the change in dimension values are used to report historical data at any given point of time.so we can implement slowly changing dimension using various approaches.

there are 6 types of Slowly Changing Dimension that are commonly used, they are as follows:

- **Type 0 – Fixed Dimension**
    - No changes allowed, dimension never changes
- **Type 1 – No History**
    - Update record directly, there is no record of historical values, the only current state
- **Type 2 – Row Versioning**
    - Track changes as version records with current flag & active dates and other metadata
- **Type 3 – Previous Value column**
    - Track change to a specific attribute, add a column to show the previous value, which is updated as further changes occur
- **Type 4 – History Table**
    - Show current value in dimension table but track all changes in a separate table
- **Type 6 – Hybrid SCD**
    - Utilise techniques from SCD Types 1, 2 and 3 to track change

So now I have taken an Example to explain this SCD given below.

Example:

Full load SCD type 2:(Before change)

Emp_No	Emp_Name	Job	Start_Date	End_Date
1	**Smith**	**SE**	**14-11-2019**	**Null**
2	**Jhon**	**SSE**	**14-11-2019**	**Null**

SCD2:(After change)

Emp_No	Emp_Name	Job	Start_Date	End_Date	Flag
1	**Smith**	**SE**	**14-11-2019**	**28-11-2019**	**0**
2	**Jhon**	**SSE**	**14-11-2019**	**Null**	**0**
3	**Smith**	**DBA**	**28-11-2019**	**Null**	**1**

Full load SCD type 3: (Before change)

Emp_No	Emp_Name	Current_Job	Previous_Job
1	**Smith**	**SE**	**Null**
2	**Jhon**	**SSE**	**Null**

SCD3:(after change)

Emp_No	Emp_Name	Current_Job	Previous_Job
1	**Smith**	**DBA**	**SE**
2	**Jhon**	**SSE**	**Null**

b) What is surrogate key? Why it is required?

Ans:

Surrogate key: Data warehouse surrogate keys are sequentially generated meaningless numbers associated with each and every record in the data warehouse. These surrogate keys are used to join dimension and fact tables.

- Usually, database sequences are used to generate surrogate key so it is always a unique number
- Surrogate keys cannot be NULLs. A surrogate key is never populated with NULL values.
- It does not hold any meaning in the data warehouse, often called meaningless numbers. It is just sequentially generated INTEGER number for better lookup and faster joins.

Why it is required:

- surrogate keys are more efficient than business key so I would always recommend joining a fact table to a dimension table using a surrogate key .business key are a good way of avoiding duplicate records so in practice, I typically add a business key as an alternative key in a dimension as a further precaution.

- Basically, a surrogate key is an artificial key that is used as a substitute for the natural key (NK) defined in data warehouse tables. We can use natural key or business keys as a primary key for tables. However, it is not recommended because of the following reasons:

- **Natural keys (NK)** or **Business keys** are generally alphanumeric values that are not suitable for the index as traversing become slower. For example, prod123, prod231 etc.

- Business keys are often reused after some time. It will cause the problem as in data warehouse we maintain historic data as well as current data.

I have given an to explain example of Surrogate key:

Example1:

Student_SK	Student_ID	Student_NAME	Student_AGE
1	**16CA01**	**Avinash Porwal**	**25**
2	**16CA02**	**Rahul Sharma**	**24**
3	**16CA03**	**Sakshi Yadav**	**25**
4	**16CA04**	**Dhurv Gupta**	**27**

c) Stores are grouped in to multiple clusters. A store can be part of one or more clusters. Design tables to store this store-cluster mapping information.

Ans: A **table cluster** is a group of tables that share common columns and store related data in the same blocks.I am trying to explain this Concept by my given example:

Example:1

Stores1

ClusterOneId	ClusterSpace	ClusterValue	StoreID
1	**400**	**String**	**100**
2	**125**	**blank**	**200**
3	**654**	**lock**	**200**
4	**780**	**mycluster**	**100**
5	**320**	**clusterSync**	**500**

Fig:1.0(uncluster table1)

Stores2

StoreID	ClusterStatus	ClusterMemory	ClusterLoc
100	**available**	**25**	**Mumbai**
200	**delete**	**96**	**USA**

Fig:1.1(uncluster table 2)

Store1_Store2_StoreCluster

Store	ClusterStatus	ClusterMemory	ClusterLoc
100	**available**	**25**	**Mumbai**

Store	ClusterOneId	ClusterSpace	ClusterValue
100	1	400	String
100	4	780	mycluster

200	**delete**	**96**	**USA**

Store	ClusterOneId	ClusterSpace	ClusterValue
200	2	125	blank
200	3	654	lock

Fig:1.2(cluster table 3)

Note: More Overview of Table Clusters

When tables are clustered, a single data block can contain rows from multiple tables. For example, a block can store rows from both the employees and departments tables rather than from only a single table.

The cluster key is the column or columns that the clustered tables have in common. For example, the employees and departments tables share the department_id column. You specify the cluster key when creating the table cluster and when creating every table added to the table cluster.

The cluster key value is the value of the cluster key columns for a particular set of rows. All data that contains the same cluster key value, such as department_id=20, is physically stored together. Each cluster key value is stored only once in the cluster and the cluster index, no matter how many rows of different tables contain the value.

Example2:

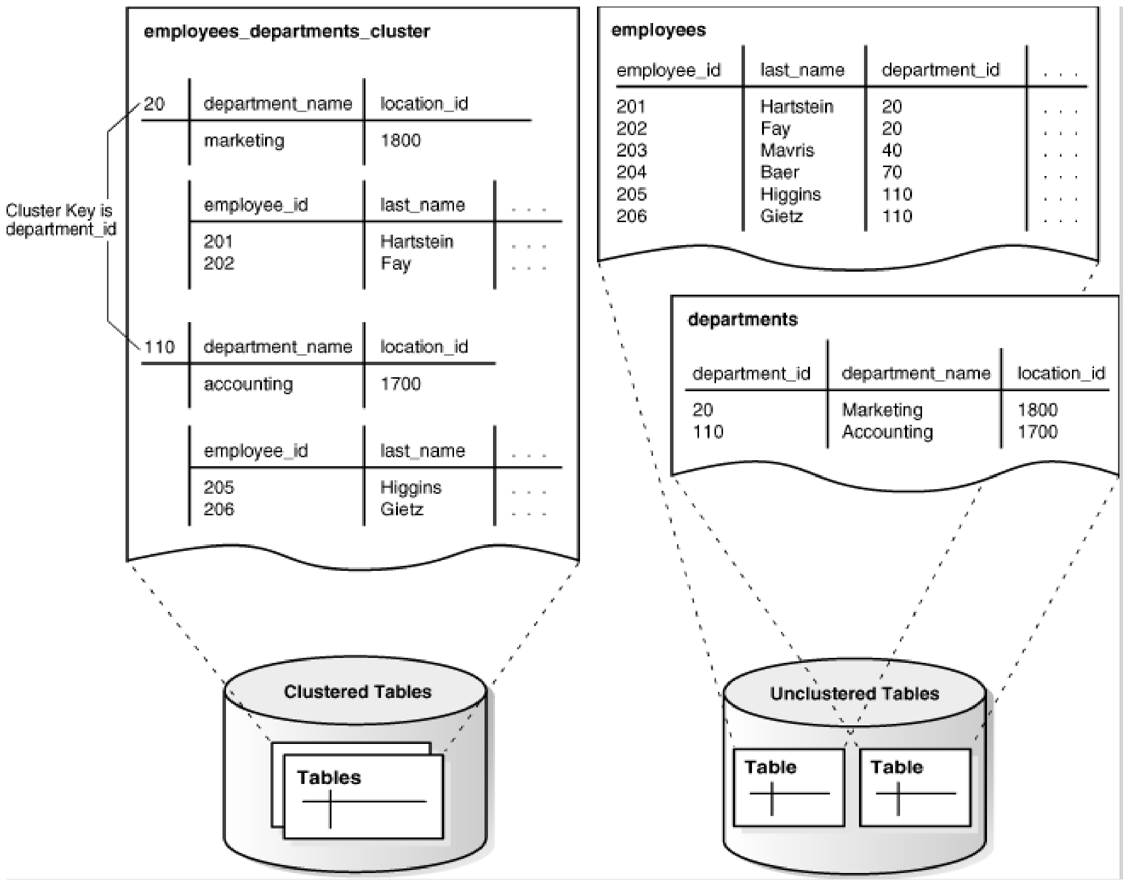


Fig:1.3(Cluster Table Data)

d) What is a semi-additive measure? Give an example.

Ans:

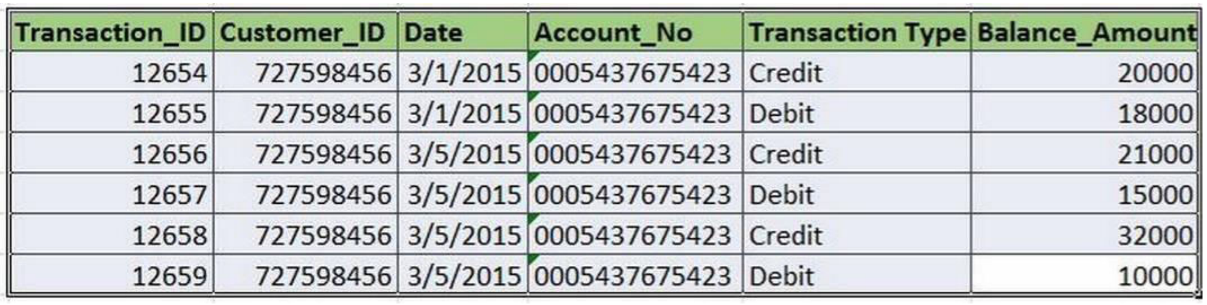
Some measure is semi-additive because they can be aggregated by some dimension, but not by others.

for example, an inventory closing balance measures is an example of a semi-additive measure. That's because the product inventory balance can be summed across the product dimension, but can not be meaningfully summed across the date dimension understand this better, consider the following inventory fact table.

Example1:

Product	March 1st	March 2nd	Total
Product A	10	15	25(15)
Product B	20	25	45(25)
Product C	30	35	65(35)
Product D	40	45	85(45)
Total by Product	100	120	220(120)

Example:2



Transaction_ID	Customer_ID	Date	Account_No	Transaction Type	Balance_Amount
12654	727598456	3/1/2015	0005437675423	Credit	20000
12655	727598456	3/1/2015	0005437675423	Debit	18000
12656	727598456	3/5/2015	0005437675423	Credit	21000
12657	727598456	3/5/2015	0005437675423	Debit	15000
12658	727598456	3/5/2015	0005437675423	Credit	32000
12659	727598456	3/5/2015	0005437675423	Debit	10000