Data Warehouse Schema

In a data warehouse, a schema is used to define the way to organize the system with all the database entities (fact tables, dimension tables) and their logical association.

Here are the different types of Schemas in DW:

1. Star Schema
2. SnowFlake Schema
3. Galaxy Schema
4. Star Cluster Schema

#1) Star Schema

This is the simplest and most effective schema in a data warehouse. A fact table in the center surrounded by multiple dimension tables resembles a star in the Star Schema model.

The fact table maintains one-to-many relations with all the dimension tables. Every row in a fact table is associated with its dimension table rows with a foreign key reference.

Due to the above reason, navigation among the tables in this model is easy for querying aggregated data. An end-user can easily understand this structure. Hence all the Business Intelligence (BI) tools greatly support the Star schema model.

While designing star schemas the dimension tables are purposefully de-normalized. They are wide with many attributes to store the contextual data for better analysis and reporting.

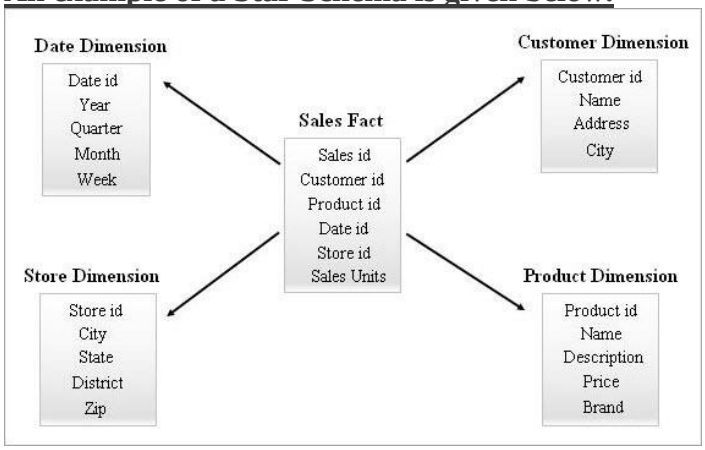
Benefits of Star Schema

- Queries use very simple joins while retrieving the data and thereby query performance is increased.
- It is simple to retrieve data for reporting, at any point of time for any period.

Disadvantages of Star Schema

- If there are many changes in the requirements, the existing star schema is not recommended to modify and reuse in the long run.
- Data redundancy is more as tables are not hierarchically divided.

An example of a Star Schema is given below.



Querying A Star Schema

An end-user can request a report using Business Intelligence tools. All such requests will be processed by creating a chain of "SELECT queries" internally. The performance of these queries will have an impact on the report execution time.

From the above Star schema example, if a business user wants to know how many Novels and DVDs have been sold in the state of Kerala in January in 2018, then you can apply the query as follows on Star schema tables:

```
SELECT    pdim.Name Product_Name,
          Sum (sfact.sales_units) Quanity_Sold
FROM      Product pdim,
          Sales sfact,
          Store sdim,
          Date ddim
WHERE sfact.product_id = pdim.product_id
          AND sfact.store_id = sdim.store_id
          AND sfact.date_id = ddim.date_id
          AND sdim.state = 'Kerala'
          AND ddim.month   = 1
          AND ddim.year    = 2018
          AND pdim.Name in ('Novels', 'DVDs')
GROUP BY pdim.Name
```

Results:

Product_Name	Quantity_Sold
Novels	12,702
DVDs	32,919

Hope you understood how easy it is to query a Star Schema.

#2) SnowFlake Schema

Star schema acts as an input to design a SnowFlake schema. Snow flaking is a process that completely normalizes all the dimension tables from a star schema.

The arrangement of a fact table in the center surrounded by multiple hierarchies of dimension tables looks like a SnowFlake in the SnowFlake schema model. Every fact table row is associated with its dimension table rows with a foreign key reference.

While designing SnowFlake schemas the dimension tables are purposefully normalized. Foreign keys will be added to each level of the dimension tables to link to its parent attribute. The complexity of the SnowFlake schema is directly proportional to the hierarchy levels of the dimension tables.

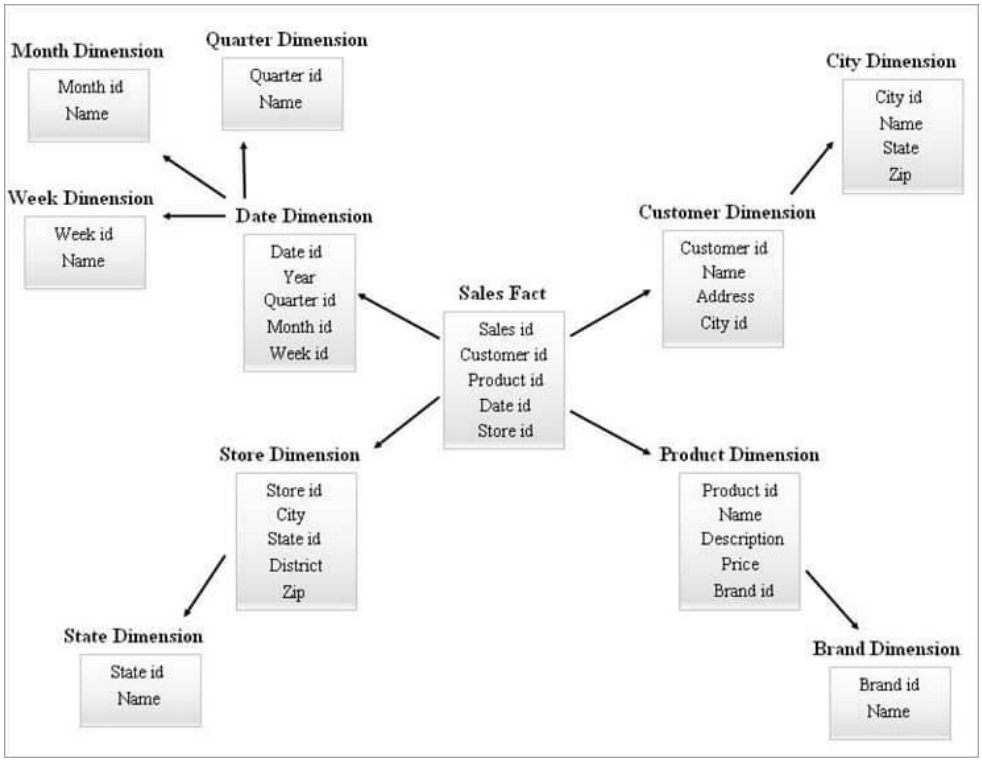
Benefits of SnowFlake Schema:

- Data redundancy is completely removed by creating new dimension tables.
- When compared with star schema, less storage space is used by the Snow Flaking dimension tables.
- It is easy to update (or) maintain the Snow Flaking tables.
-

Disadvantages of SnowFlake Schema:

- Due to normalized dimension tables, the ETL system has to load the number of tables.
- You may need complex joins to perform a query due to the number of tables added. Hence query performance will be degraded.

An example of a SnowFlake Schema is given below.



The Dimension Tables in the above SnowFlake Diagram are normalized as explained below:

- Date dimension is normalized into Quarterly, Monthly and Weekly tables by leaving foreign key ids in the Date table.
- The store dimension is normalized to comprise the table for State.
- The product dimension is normalized into Brand.
- In the Customer dimension, the attributes connected to the city are moved into the new City table by leaving a foreign key id in the Customer table.

In the same way, a single dimension can maintain multiple levels of hierarchy.

Different levels of hierarchies from the above diagram can be referred to as follows:

- Quarterly id, Monthly id, and Weekly ids are the new surrogate keys that are created for Date dimension hierarchies and those have been added as foreign keys in the Date dimension table.

- State id is the new surrogate key created for Store dimension hierarchy and it has been added as the foreign key in the Store dimension table.
- Brand id is the new surrogate key created for the Product dimension hierarchy and it has been added as the foreign key in the Product dimension table.
- City id is the new surrogate key created for Customer dimension hierarchy and it has been added as the foreign key in the Customer dimension table.

Querying a Snowflake Schema

We can generate the same kind of reports for end-users as that of star schema structures with SnowFlake schemas as well. But the queries are a bit complicated here.

From the above SnowFlake schema example, we are going to generate the same query that we have designed during the Star schema query example.

That is if a business user wants to know how many Novels and DVDs have been sold in the state of Kerala in January in 2018, you can apply the query as follows on SnowFlake schema tables.

```
SELECT    pdim.Name Product_Name,
          Sum (sfact.sales_units) Quanity_Sold
FROM      Sales sfact
INNER JOIN Product pdim ON sfact.product_id = pdim.product_id
INNER JOIN Store sdim ON sfact.store_id = sdim.store_id
INNER JOIN State stdim ON sdim.state_id = stdim.state_id
INNER JOIN Date ddim ON sfact.date_id = ddim.date_id
INNER JOIN Month mdim ON ddim.month_id = mdim.month_id
WHERE stdim.state = 'Kerala'
          AND mdim.month   = 1
          AND ddim.year    = 2018
          AND pdim.Name in ('Novels', 'DVDs')
GROUP BY pdim.Name
```

Results:

Product_Name	Quantity_Sold
Novels	12,702
DVDs	32,919

Points To Remember While Querying Star (or) SnowFlake Schema Tables

Any query can be designed with the below structure:

SELECT Clause:

- The attributes specified in the select clause are shown in the query results.
- The Select statement also uses groups to find the aggregated values and hence we must use group by clause in the where condition.

FROM Clause:

- All the essential fact tables and dimension tables have to be chosen as per the context.

WHERE Clause:

- Appropriate dimension attributes are mentioned in the where clause by joining with the fact table attributes. Surrogate keys from the dimension tables are joined with the respective foreign keys from the fact tables to fix the range of data to be queried. Please refer to the above-written star schema query example to understand this. You can also filter data in the from clause itself if in case you are using inner/outer joins there, as written in the SnowFlake schema example.
- Dimension attributes are also mentioned as constraints on data in the where clause.
- By filtering the data with all the above steps, appropriate data is returned for the reports.

As per the business needs, you can add (or) remove the facts, dimensions, attributes, and constraints to a star schema (or) SnowFlake schema query by following the above structure. You can also add sub-queries (or) merge different query results to generate data for any complex reports.

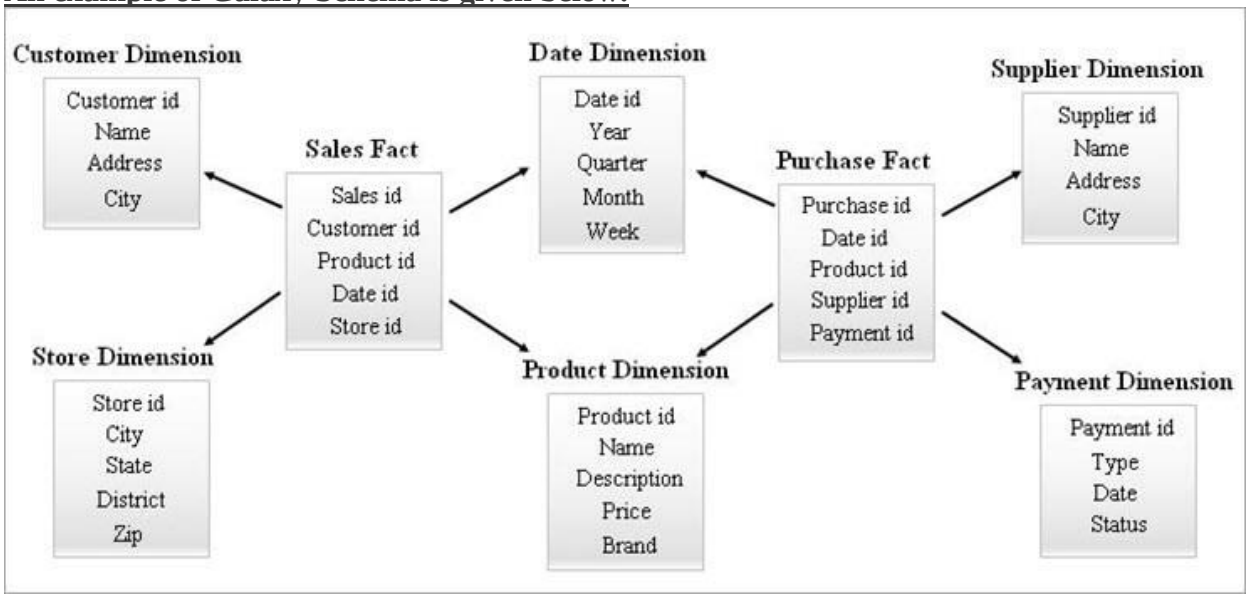
#3) Galaxy Schema

A galaxy schema is also known as Fact Constellation Schema. In this schema, multiple fact tables share the same dimension tables. The arrangement of fact tables and dimension tables looks like a collection of stars in the Galaxy schema model.

The shared dimensions in this model are known as Conformed dimensions.

This type of schema is used for sophisticated requirements and for aggregated fact tables that are more complex to be supported by the Star schema (or) SnowFlake schema. This schema is difficult to maintain due to its complexity.

An example of Galaxy Schema is given below.

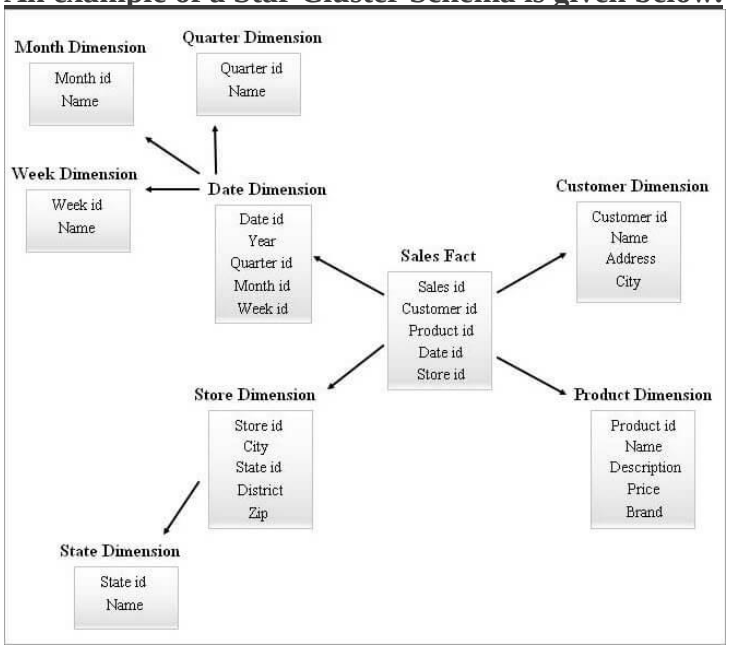


#4) Star Cluster Schema

A SnowFlake schema with many dimension tables may need more complex joins while querying. A star schema with fewer dimension tables may have more redundancy. Hence, a star cluster schema came into the picture by combining the features of the above two schemas.

Star schema is the base to design a star cluster schema and few essential dimension tables from the star schema are snowflaked and this, in turn, forms a more stable schema structure.

An example of a Star Cluster Schema is given below.



Which Is Better Snowflake Schema Or Star Schema?

The data warehouse platform and the BI tools used in your DW system will play a vital role in deciding the suitable schema to be designed. Star and SnowFlake are the most frequently used schemas in DW.

Star schema is preferred if BI tools allow business users to easily interact with the table structures with simple queries. The SnowFlake schema is preferred if BI tools are more complicated for the business users to interact directly with the table structures due to more joins and complex queries.

You can go ahead with the SnowFlake schema either if you want to save some storage space or if your DW system has optimized tools to design this schema.

Star Schema Vs Snowflake Schema

Given below are the key differences between Star schema and SnowFlake schema.

S.No	Star Schema	Snow Flake Schema
1	Data redundancy is more.	Data redundancy is less.
2	Storage space for dimension tables is more.	Storage space for dimension tables is comparatively less.
3	Contains de-normalized dimension tables.	Contains normalized dimension tables.
4	Single fact table is surrounded by multiple dimension tables.	Single fact table is surrounded by multiple hierarchies of dimension tables.
5	Queries use direct joins between fact and dimensions to fetch the data.	Queries use complex joins between fact and dimensions to fetch the data.
6	Query execution time is less.	Query execution time is more.
7	Anyone can easily understand and design the schema.	It is tough to understand and design the schema.
8	Uses top down approach.	Uses bottom up approach.

Conclusion

We hope you got a good understanding of different types of Data Warehouse Schemas, along with their benefits and disadvantages from this tutorial.

We also learned how Star Schema and SnowFlake Schema can be queried, and which schema is to choose between these two along with their differences.