Benjamin Huus, Raja Yeswanth, Sreenilaya Cherukumalli

Final Report

Data Warehousing

# An overview of Data Warehousing using Star/Snowflake Schemas

Abstract

The architecture of the database schema plays a pivotal role in determining the efficiency and scalability of data retrieval and storage. Among the various schema designs, the snowflake schema stands out due to its highly normalized structure, which optimizes the organization of data into multiple related tables. This method of organizing data not only minimizes redundancy but also enhances data integrity and storage efficiency. By breaking down data into its most granular components, a snowflake schema supports more complex queries and provides a more detailed and precise representation of business data. This report delves into the benefits of implementing a snowflake schema in data warehouses, highlighting its impact on data management, query performance, and overall system maintainability. Through a comprehensive analysis, we aim to demonstrate how the snowflake schema can serve as a robust framework for businesses seeking to harness their data assets to drive informed decision-making and achieve operational excellence.

Star Schema

A star schema is a type of database schema that is optimized for data warehousing and online analytical processing (OLAP) applications. It is called a start schema because its structure resembles that of a star, with a central fact table connected to several dimensional tables. The star schema is designed to simplify querying and improve the performance of complex queries by organizing data in a way that makes it easy to understand and navigate.

The fact table is the central table in the star schema. It contains quantitative data related to the business process being evaluated. Each row in the fact table typically represents a unique event or transaction. Common attributes in the fact table include numeric values such as sales amount, quantity sold, and revenue. The table also contains foreign keys that reference primary keys in the dimension tables, establishing relationships between the fact table and the dimension tables.

Dimension tables surround the fact table and provide descriptive context to the facts. Each dimension table contains attributes that describe aspects of the business process, such as time, product, customer, or geography. Dimension tables are traditionally denormalized, meaning they store redundant data to make it easier and faster to perform queries. Each dimension table has a primary key that uniquely identifies each row, and these primary keys are used as foreign keys in the fact table.

Below is an example of a sales table:

A diagram of a data flow

Description automatically generated

Figure 1 - Star Schema

In this example, each sales transaction is stored within the Sales table, which is being used as the fact table. You can see various foreign keys within the Sales table such as ProductID, CustomerID, TimeID, and GeographyID which are used to connect the dimension tables. The dimensional tables in this example are Product, Customer, Time and Geography.

There are many reasons why star schemas are effective for data warehousing. A star schema is fast for querying primarily due to its simplified structure, which is designed to optimize data retrieval and improve query performance. One key reason for the improved query performance is that the dimension tables are typically denormalized. This means that they store redundant data to minimize the number of joins required during queries. This reduces the complexity of queries and improves performance. Denormalized tables allow for fewer tables to be joined, which reduces the time needed to execute queries. A star schema is also a very straight forward data model. With a central fact table and surrounding dimension tables, this makes it easy for users to understand and navigate. There are clear relationships between the fact table and the dimension tables to simplify query construction, which leads to faster and more efficient query processing. Star schemas are optimized to work for Online Analytical Processing (OLAP), which involves complex queries that aggregate and analyze large amounts of data. The schema’s structure supports efficient aggregation and summarization, which are common operations in OLAP queries. Indexing is easily done within a star schema and is created on the primary keys of dimension tables and the foreign keys in the fact table. These indexes improve the speed of joins and data retrieval. Fact tables often have clustered indexes on the primary key, which enhances the performance of range queries and aggregations. A common methodology in a star schema is to pre-computer and store aggregate data in the fact table. This reduces the need for on-the-fly calculations during query execution, speeding up the retrieval process. Common aggregations like sums, averages, and counts can be quickly retrieved from the fact table without recalculating them from raw data. Since the star schema minimizes the need for multiple joins by denormalizing dimension tables, the query execution plan becomes simpler and faster. Fewer joins mean less computational overhead and faster query execution times. Fact tables in star schemas are often partitioned based on time or other logical divisions, which allow for efficient data retrieval and query performance. Partitioning helps in managing large datasets by reducing the amount of data scanned during query execution. Overall, the star schema’s design principles and optimization techniques contribute to its efficiency and speed in querying large datasets in data warehousing environments.

Snowflake Schema

A snowflake schema is a type of database schema used in data warehousing and OLAP applications. It is an extension of the star schema where the dimension tables are normalized, splitting them into additional tables. This results in a structure that resembles a snowflake, with the fact table at the center connected to several normalized dimension tables. The normalization of dimension tables aims to reduce the data redundancy and improve data integrity.

Similar to the star schema, the fact table in a snowflake schema contains quantitative data related to business processes. The fact table includes foreign keys that reference primary keys in the dimension tables. The key difference between the star and snowflake schema is that in the snowflake schema, the dimension tables are normalized. These tables are decomposed into multiple related tables. The normalization process involves splitting the original dimension tables into smaller tables to eliminate redundancy and ensure data integrity.

Below is an example of a snowflake schema:

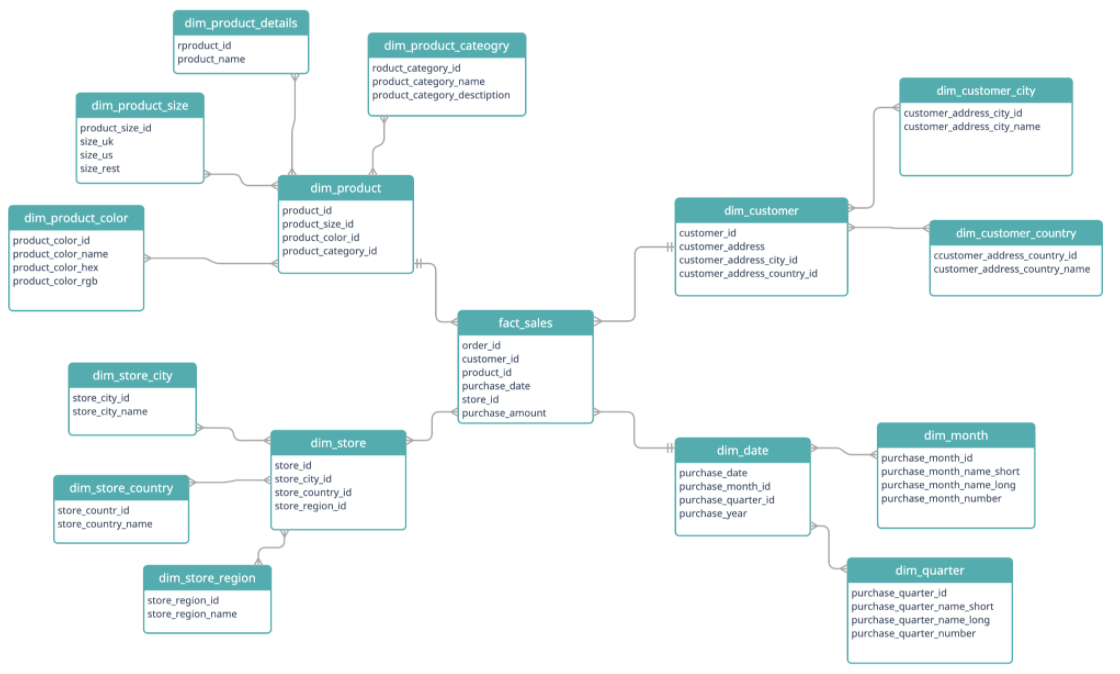


Figure 2 - Snowflake Schema

As you can see in this example, the fact table is again a Sales table, or the fact\_sales. This then breaks into four dimensional tables: dim\_product, dim\_customer, dim\_store, and dim\_date. Each of these dimensional tables are connected to the fact table by foreign keys: product\_id, customer\_id, store\_id, and purchase\_date. Unlike the star schema, the dimensional tables are normalized. This means that each of the dimension tables are then broken down further into smaller dimensional tables. The resulting data model resembles a snowflake, hence the name.

The snowflake schema offers several advantages over the star schema, particularly in areas of data redundancy, integrity, and storage efficiency. The snowflake schema normalizes dimension tables, which means that data is stored in a more structured manner, reducing redundancy. Each piece of information is stored only once, which saves storage space and avoids the need for duplicate entries. By normalizing the dimensional tables, the snowflake schema ensures data consistency and integrity. Any updates to a piece of information are made in one place, reducing the risk of data anomalies and ensuring that all references to that data are consistent. Since the data is not duplicated, the snowflake schema can be more storage efficient. This can be particularly beneficial in large data warehouses where storage costs are a concern. Normalized tables make it easier to update and maintain the database. Changes to dimension attributes need to be made in only one place, reducing the effort required to keep the database current and accurate. A snowflake schema is more complex and hierarchical which can make it easier to organize and categorize data logically. The snowflake schema can be more scalable for very large databases with many dimensions and attributes. Its normalized structure can handle a greater number of attributes without becoming unwieldy. Normalization of the tables helps in eliminating insertion, update, and deletion anomalies. By having a single source of truth for each piece of data, the schema helps maintain data accuracy and consistency.

Although there are many advantages of the snowflake schema, it comes at the cost of increased complexity and potentially slower query performance due to the need for more joins. The choice between a star schema and a snowflake schema depends on the specific requirements and constraints of the data warehousing project, including the importance of query performance versus data integrity and storage efficiency.

SnowFlake Implementation

For this experiment, the dataset used to create our Star/Snowflake schema originated from the following github project: <https://github.com/3amory99/Gravity-Books-Sales-End-to-End-Project>

For this implementation, the fact table was: fact\_order\_line. The fact\_order\_line table consists of transactional information for the fictional bookstore Gravity Books. Some of the transactional information includes order\_id, book\_id, price, order\_date, etc. Because it is a fact table, there are multiple foreign keys to connect the dimesion tables. These tier one dimensional tables are:

1. Cust\_order – the customer orders dimensional table
2. Book – each of the book properties is located in this dimension table
3. Customer – Information for each customer is stored in this table
4. Shipping\_method – shipping information
5. Address – Address information
6. Order\_status – A simple normalized table that defines the status of orders

Below is a snippet of the SQL used to create the snowflake fact table:

A computer screen shot of a computer code

Description automatically generated

Figure 3 - Snowflake Fact Table

Next, we have the dimensional tables. One example of a dimensional table is the Book table. This dimensional table holds information such as the book title, number of pages, publication date, etc. Because we are operating in the mindset of a snowflake schema, we’ll want to normalize this dimension table to help improve the storage efficiency. The book table is normalized by creating the deeper dimensional tables book\_language and publisher. These tables are connected on the foreign keys language\_id and publisher\_id.

Below is a code snippet of the book table generation:

A computer code with blue text

Description automatically generated

Figure 4 - Book Dimension Table

Finally, the leaf node dimension tables, book\_language and publisher make our schema a snowflake. Below is an code example of the publisher dimension table creation:

A close up of a computer screen

Description automatically generated

Figure 5 - Publisher Leaf Dimension Table

To see all the code for this implementation of the snowflake schema as well as the code used to generate the data, please visit: <https://github.com/bhuus/snowflake_schema>

The final ERD diagram for this schema is a bit complicated due to the nature of a snowflake schema, but below is a depiction derived from pgadmin:

A screenshot of a computer screen

Description automatically generated

Figure 6 - Snowflake Schema ERD

The green table is the central fact table, and the blue tables are the branching dimension tables.

Conclusion

In conclusion, both star and snowflake schemas offer distinct advantages and are suited to different scenarios in data warehousing. The star schema, with its denormalized structure, excels in simplicity and query performance, making it ideal for environments where speed and ease of use are paramount. Its straightforward design facilitates rapid data retrieval and efficient execution of complex queries, making it a preferred choice for many OLAP applications. The snowflake schema’s normalized structure enhances data integrity, reduces redundancy, and optimizes storage efficiency, making it suitable for large-scale data warehouses where maintaining consistency and managing storage costs are critical. While the snowflake schema’s complexity can introduce additional maintenance overhead and potentially slower query performance due to multiple joins, its logical organization supports detailed analysis and scalability. Ultimately, the choice between a star and snowflake schema should be guided by specific project requirements, balancing the need for query performance, data integrity, and storage consideration to achieve the optimal data warehousing solution.