Abstract - Design of an efficient data warehouse that can provide a comprehensive platform for big data storage and an optimized query processing has been receiving major attention by various organizations. It is essential for the data modeler to have good knowledge and understanding about the design of an appropriate data warehouse based on the structure of the data and how the information is stored for query retrieval or presentation. In this paper, four different methods are presented that extend the design of the star schema and snowflake schema to better support the data warehouse design. The methods were implemented and their advantages and disadvantages based on the designs and query processing performances were analyzed and compared to each other. A designer could use each method individually or in combinations to provide more suitable and efficient data warehouse structure for implementation and query processing.

Keywords: Data Warehousing, Star Schema, Snowflake Schema, SQL, Data Mining, Normalization, Optimization

1 Introduction

Over the years, Business Intelligent (BI) for Data Warehousing has received major attention from various organizations. It has become very important in storing and retrieving data for decision making. Since the introduction of the data warehouse, the process of building a warehouse as a centralized data collection and repository for Decision Support Systems (DSS) has evolved through phases of research development and is becoming more widely accepted by diverse business organizations. The creation of a serviceable data warehouse requires substantial effort and experience in the intricate design as well as sophisticated implementation procedures. In a data warehouse, the database expands immensely. This combined with the often ad-hoc and complex queries to the database dictates that an essential task in the design process of the warehouse must be to consider ways to reduce the response time and thus minimize the cost of the query effort [1], [2], [3].

In the design of a data warehouse, one of the fundamental structures utilized is the star schema, which has become the design of choice because of its compact structure. The structure utilizes a central table and one level of surrounding tables that results in a more optimized query performance and ease of use compared to the structures of prior data warehouse designs. One structural dilemma that can occur in the star schema involves the handling of multi-values and duplicate values [4]. The affected tables are commonly normalized to a higher level resulting in another structure referred to as a snowflake schema [5]. Although the snowflake structure can handle the issues of multi-values and duplicate values, the higher normalization of the design also increases the complexity of the design structure by adding more levels of tables. The expansion of the number of tables dramatically raises the number of joins required for queries thus prolonging the query retrieval time. The end result is a decrease in the efficiency of the query response time for the DSS.

This paper will offer four design methods with two creative techniques to overcome the star schemas limitations in supporting multi-values and duplicate values with displays of time performance for each method. The techniques maintain the values in the star structure and thereby avoid the need to proceed to the snowflake schema. The preservation of the star schema in the data warehouse design retains the simpler design structure and allows a variety of queries to be efficiently processed. First, the current utilization of Method 1 and Method 2, which are the star and snowflake, will be reviewed. This is followed by a description of Method 3 using a known number of multi-valued attributes and Method 4 using an unknown number of multi-valued attributes. Both Methods 3 and 4 utilize an extended star schema. After the descriptions, the application of efficient data mining using different queries with simple, complex and aggregated structures is examined and the time performances will be analyzed for enhanced decision-making and design strategies.

2 Data Warehouse Design

For a data warehouse, relevant information is gathered from an existing operational database in the form of records and stored in the data warehouse for future query processing, analysis, and evaluation. The collections of records are stored
in two types of tables, one fact table and many dimensional tables [6], [7]. The operational data, grouped into related records as historical data, are distributed into their related dimensional tables according to their types. The fact table holds the keys from each dimensional table. The configuration of the fact table holding a key from each dimensional table to associate all the related dimensional tables together results in a structure called the star schema. A data warehouse may contain many star schemas with each one supporting different dimensionalities through association of related records in the fact table. An example of a star schema is given in Figure 1 that includes Customer, Location, Product, and Calendar to support a fact table for Sales. As shown in Figure 1, the Method 1 schema supports a simple design that allows the data from the operational database to be Extracted, Transformed and Loaded (ETL) to the star structure.

The normalization of the star schema from Figure 1 converts the related dimensional tables to a higher normal form table(s) named as Method 2 for design of the structure. The higher normalized star schema structure results in more than one level of tables associated to the fact table. The new structure in Method 2 is referred to as a snowflake schema [8], [9]. This schema is more organized than the star schema in that it reduces the duplicates, but it also results in more dimensional tables from decomposition of original table(s), which in turn requires more complex query joins and lengthens the query response. Although the multi-values and duplicate values may be handled by normalizing a star schema into a snowflake schema, the advantages of the star design for design simplicity and query optimization are sacrificed. The proposed techniques in this paper offer alternative methods of handling multi-values and data normalization while preserving the star schema. Both methods are extensions to the star design and reduce the need to utilize the snowflake structure.

3 Extended Column-Wise Star Schema

In Method 3, a new schema to preserve the structural efficiency of the star schema and to allow multi-value data to be used in the same dimensional tables is provided. The operational tables and their data structures allow the designer to have a good working knowledge of the multi-value fields or duplicate attribute names that are going to be used for modeling the data warehouse. This method will take multiple values for the same attributes from the operational database and move them to a star schema in a column-wise order in the dimensional table(s).

Application of higher normalization to the star schema in Method 1 resulted in the snowflake structure in Method 2. In Method 3, rather than transforming the star schema to a snowflake by utilizing higher normalization, the design is kept in the star schema structure by employing the multiple values, such as storage locations, district managers or regions, with a special arrangement in the location table which is similar to normalization.
In Method 3, duplicate attributes are distributed column-wise in the same table, instead of normalizing them by moving them to another table or entity. This is also done with other entities, such as product and calendar. Since the designer of the data warehouse knows in advance or ahead of time about the number of the columns for attribute names (such as storage locations, product, and calendar) from the operational database, this knowledge will facilitate the design process. The dimensional tables with multiple attribute values can be created as Method 3 with their new columns to preserve the star schema structure similar to Method 1. Figure 3 shows the star schema with multiple attributes (such as CLDR_DT1, 2, 3, 4, 5, PREV_YR_DT1, 2, 3, 4, 5, ...) for the data warehouse design.

The disadvantage of this method is that some of the records in the tables (such as calendar) may have less number of values for sales amount. This will result in null or no value in that column. On the other hand, the advantage of this method is that the result is a compact design having fewer tables by allowing multiple attribute names and values to be present in one table (such as the product table) as opposed to two or more tables using higher normalization. This method also supports the star schema structure for simplicity and efficiency of the design which results in more optimized query retrieval with less number of joins in the dimensional tables. In general, when the number of multi-value attributes is known from the operational database system, Method 3 is the most suitable design as a technique to maintain the star schema in the data warehouse design.

4 Extended Row-Wise Star Schema

The operation databases are dynamic. Due to organization growth, operational databases expand daily through the addition of new information. This information could include new products or locations to the provided example as a result of expansion. Filtering of these types of multi-value data for the historical data will result in higher normalization and a structure similar to the snowflake schemas in Method 2. When the number of duplicate attributes or possible multiple values for the tuples of the data warehouse from different tuples of the operational databases are not known, Method 4 can be utilized in the data warehouse design to preserve the star schema structure by avoiding the snowflake schema and decomposition of the dimensional tables with duplicates into two or more tables.

In Method 4, the data warehouse can be designed using the historical data from the location table and new location by categories or managers allowing for future expansion of historical data to be handled through distribution of multiple locations in a tuple of the dimensional tables rather than the attributes of the location dimensional table. As shown in the following schema of the location dimensional table, the district manager and region attribute names will be repeated only once. This method will allow for future expansion of a new manager data and district by adding the new historical tuple to the location dimensional table. In Method 4, any multi-valued attribute is added as new record with a Row-Id to the table (existing table) instead of being normalize by moving to another table. To recognize all the records with the same RGN_ID, a new attribute with the name LOC_ROW_ID is used to represent the location records with duplicate values that are added to the data warehouse table. An example of Method 4 is shown in Figure 4.

The advantage of this method is that future expansion of new products and their locations in the operational database can be handled easily in the data warehouse tables.
Storing multi-values in the data warehouse star schema provides a more optimized structure than the normalized schema design of snowflakes. As shown in Figure 4, the sales, scan quantities and scan weight for each product and location can be stored separately. The disadvantage of this method is that it will have some duplicate values. More detailed advantages and disadvantages of the different methods will be discussed further in Section 7.

5 Software Design and Data Mining

The software used for design and implementation of the data warehouse system was Oracle and the interface was Java with embedded SQL using the Linux environment to calculate the data mining query response time and performance. For the data warehouse, the four described methods were created in the Oracle database system and the records were added to each database tables by making sure the dimensional tables supported similar collections of the record structures and numbers as other methods. The performance evaluation was carried out by running queries of various degrees of complication using Java to record the query run times. Three sets of queries were designed to provide each method with the same data mining and information retrieval criteria for the designed warehouse systems. These queries were considered as simple, complex and aggregated. The sample queries are listed below.

The performance for a simple query involved the core tables only, wherein the complexity of the snowflake design would not come into consideration. The following is the sample of simple query for the Method 1.


The complex query involves a "Where" condition for all the tables and the hierarchy. This associates all the joins and tables in the respective designs. The following is a sample of the complex query for Method 1.


An aggregated query is a step more complex query as it performs an aggregation operation along with involving all the joins and tables in the designs. The following is a sample of an aggregated query for Method 1.


The queries for the other methods were developed in the same manner to provide the same results. The time complexity for each method was tallied and tabulated.

Figure 4. Method 4 - Row-wise distribution
6 Performance Analysis for Methods

For performance, each database was populated with the necessary records for queries to calculate the performance by the query response time. The records for each method were arranged to represent the same size and structure. Each query was processed 5 times and the average was calculated.

In the star schema, the records for each of the tables were sales (698), locations (5), products (5), customers (8) and calendar records (28). As shown in Figure 5, the result of processing for the simple query averages was 17.6 milliseconds. On running the complex "Where" clause query, an increased performance impact averaging 23.2 milliseconds was found. The aggregated query showed a lesser increase in query runtime averaging 23.8 milliseconds.

In the snowflake schema for the same records, the query run time for the simple query averages 18.2 milliseconds which was slightly more than in the star schema. On running the complex "Where" clause, the time difference increased further to 25.2 milliseconds, with the maximum difference seen while running the aggregated query at almost 3.8 milliseconds more than star schema.

For the extended star schema with column-wise in Method 3 and the extended star schema with row-wise in Method 4, there are lower processing times for the complex and aggregated queries as shown in Figure 5.

7 Technical Details of Methodologies

In this section, the different data warehousing methods are briefly described including advantages and disadvantages.

Method 1: Star Schema
In the star schema, the fact table is surrounded by four dimensions to form a star schema. The simplest model is also the most effective and preferred choice for design and implementation of data warehouses.

JOIN description for Method 1: In this design, the dimensions are joined to the fact table using a primary-foreign key relationship. For example, the location dimensional table and the sales fact table are joined at the lowest granular level LOC_ID.

Advantages of Method 1
1. The design of the star schema has an ease in implementation. At its simplest, the star schema is presented by a fact table surrounded by dimensions. Multiple star schemas are possible by the concept of composite dimensions which results in fact constellations.
2. Query performance is generally best for this design due to the fewer number of joins and tables within the query.

Disadvantages of Method 1
1. Redundancy is a drawback. Due to denormalization, a lot of duplicates are observed in the data. While this has become less of a drawback with hardware memory becoming cheaper, this can affect query performance when the size of the dimensions is very large.
2. Data loads are more time consuming to process because of the denormalized nature of the data.

Method 2: Snowflake Schema
A snowflake schema is more complex in the implementation as it normalizes the dimension tables. It resolves the issues associated with the star schema, where duplicated and multi-values are removed because of normalization and data load times are much faster. However, because of its complexity, the query response time is affected.

JOIN description for Method 2: In this design, the dimensional tables are normalized into more than 1 table and the dimension table with the lowest grain generally is joined to the fact table. The normalized dimension is connected to the tables at the various hierarchy levels. For example, the lowest granular location dimensional table LOCATION and the sales fact table are joined at the LOC_ID column level, while dimensional tables LOCATION is joined to the next granular level dimensional table LOCATION_1 at DSTR_MGR_ID level and similarly.

Advantages of Method 2
1. The redundancy problem associated with the star schema is resolved with the implementation of normalization.
2. Data load issues and loading time are much faster because of the high degree of normalization.

Disadvantages of Method 2
1. Query retrieval is impacted because of the increased number of tables and joins involved.
2. Because of the complexity of the design, maintenance becomes more complicated in the case of a large complex data warehouse.

Method 3: Extended Column-Wise Star Schema
A proposed alternate design is the extended column-wise star schema. In this design, the star distribution is preserved by expanding the replicating data in columns one after the other to maintain the star schema model. Thus, the star schema model is maintained. However, the design can only be implemented when there are known number of multi-valued attributes, which requires the data modeler to have a very good knowledge of the number of multi-valued attributes for each column. A change in the fact table is required, as the fact table will only contain values at the highest aggregation across all dimensions, which reduces the data in the fact table significantly. The measures at the individual dimensional level are passed on to the distinct dimensions.
JOIN description for Method 3: In this design, the joins between the dimensions and the fact table are at the highest grain. For example, the location dimensional table and the sales fact table are joined at the RGN_ID level.

Advantages of Method 3
1. Much faster query response as the star schema is maintained.
2. Overcomes the duplication issue associated with star schema without normalization.

Disadvantages of Method 3
1. The model works only when the number of multi-valued attributes for each column is known.
2. Data load is still an issue and can be error prone if not done correctly as per data model and requires data transformation.
3. A thorough knowledge of the data model is required placing a lot of reliance on data.
4. Data redundancy will occur as not all columns have data for multi-valued attributed columns.

Method 4: Extended Row-Wise Star Schema
A proposed second alternate design is the extended row-wise star schema. In this design, the star distribution is preserved by expanding the replicating data in rows vertically to maintain the star schema model. The dimensional keys start from the highest grain and with a corresponding row id as a composite key to define the relationship between the dimensions and fact. It can be utilized when the data modeler does not know the number of repeating attributes in each column. As the total number of attributes are unknown, for each column every new attribute is added row-wise by increasing the defined row count. A dimensional row id is added. This model also results in changes to the fact table, wherein the primary key are now comprised of all the highest grains dimensions and their row ids.

JOIN description for Method 4: The joins are not the simple join between a column in two tables, but a composite key comprising of a higher granular level (highest hierarchy) column and a dimensional row id. These result in a multi-column join between the dimensions and the fact table. The fact table keys are comprised of the higher granularity plus dimensional row ids as a composite foreign key. For example, the location dimensional table and the sales fact table are joined at the composite key level comprised of the RGN_ID and LOC_ROW_ID columns.

Advantages of Method 4
1. This design can be implemented when the number of multi-valued attributes is unknown.
2. Good query response times for complex queries.

Disadvantages of Method 4
1. Because of the composite key join, simpler queries may take longer to execute.
2. Data loads are still an issue as the data needs to be transformed to suit the row id – composite key implementation.
3. Data redundancy will occur as not all columns may have data for multi-row id attributes.

8 Conclusion
One of the most critical steps in the developmental process of a data warehouse is the provision of a design that will produce optimal functioning of the data retrieval needs for the Decision Support System. An important structure in the design of a data warehouse is the star schema which provides a design structure that allows a variety of queries to be efficiently processed. Based on the performance analysis and the results observed, the star schema is the best performing design for data warehousing. In data warehousing, where data can run up to petabytes, the ability to do complex analysis and fast query retrieval is generally desired. The star schema design is best suited to do so. However, the star schema has its own drawbacks which are not limited to duplication, high data load time and huge dimensional tables. These issues are solved by the snowflake design model, which normalizes the dimensions to overcome the drawbacks associated with the star schema. This normalization in turn introduces more complexity in the model in the form of new tables and joins, which in turn affects query performance. By maintaining the multiple values within the star structure, the data warehouse benefits from the preservation of the star schema and using a column-wise or row-wise design which results in efficiency of query processing.

The consideration of different methods in the design of a data warehouse, such as the ones proposed in this paper, will result in more optimized warehouses for the storage and retrieval of data through data mining by Decision Support Systems.


9 References


