Paper 123-26

Querying Star and Snowflake Schemas in SAS

Bart Heinsius, E.O.M. Data, Hilversum. The Netherlands

ABSTRACT

Star schemas, and sometimes snowflake schemas, are often used in Data Warehouses as the storage structure for dimensional data that is to be queried efficiently. In Data Warehouses built using SAS® Software, star and snowflake schemas can also be implemented.

Star and snowflake schemas in SAS can be queried using SAS SQL or the SAS DATA step. Tests have shown that using the DATA step can result in significant performance gains over SQL.

This paper will discuss the use of star and snowflake schemas in SAS and will look into the performance issues that arise. Then, a number of techniques are discussed that address these issues. They involve bitmap indexes and DATA step code generation from dedicated metadata. Then, the Star Class is introduced, which implements these techniques.

This paper should be of interest to Data Warehouse architects and builders and to OLAP application builders working with SAS Data Warehouses used for dimensional querying. An assumption is made that the reader is familiar with dimensional modeling and has a basic knowledge of BASE SAS software, SQL and SAS/EIS® software.

INTRODUCTION

Dimensional modeling techniques are often used for Online Analytical Processing (OLAP) applications, such as the Multidimensional Report object in SAS/EIS software. Dimensional modeling allows for the analysis of facts (analysis variables) over dimensions (class variables). Figure 1 shows an example of a dimensional model on car sales.

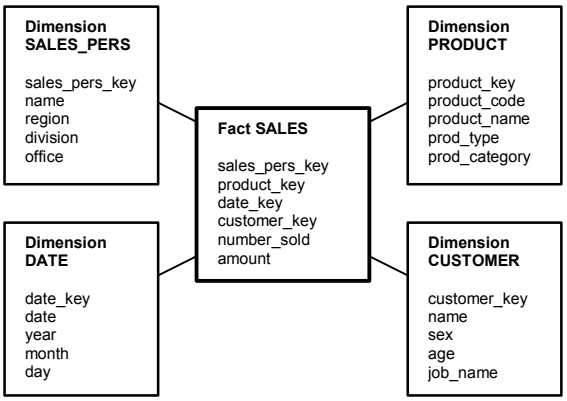

Figure 1. An example Star Schema.

The dimensional model depicted in Figure 1 is known as a Star Schema. A dimensional model is a Star Schema when it has all dimension tables joined directly to the fact table. The benefits of using a star schema for querying lies in the fact that all dimension tables are directly joined with the fact table and that the number of joins that need to be performed to obtain the query result is limited.

Another dimensional model that is sometimes used is the Snowflake Schema. Snowflake schemas are like star schemas, except that the constraint that every dimension table is directly joined to the fact table is dropped. Figure 2 shows an example.

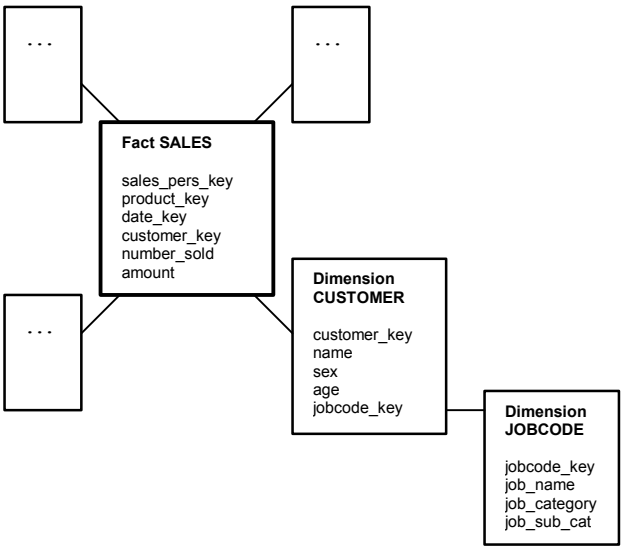

Figure 2. An example Snowflake Schema.

Snowflake schemas are much less used than star schemas. It is the author's opinion that, in certain situations, snowflake schemas are better suited than star schemas. In these situations, Data Warehouse architects often still choose star schemas because many relational database management systems (RDBMSs) have problems optimizing queries on snowflake schemas. As you will read further on in this paper, the SAS System can, with a little help, very well optimize a query on a snowflake schema too.

Using SAS Software, you can create your own reports from both star and snowflake schemas. The SQL Procedure provides an easy method of querying your dimensional data.

Standard SAS software provides the Multidimensional Data Provider (MDP) object. Through this object, SAS/EIS applications like the Multidimensional Report object can report dimensional data from several data sources, such as SAS data sets, multidimensional databases (MDDBs), views on external RDBMS files, and also from star schemas. Other applications like Enterprise Reporter™ software, Enterprise Guide® software, and the Open OLAP Server can also use the MDP.

The MDP is a perfect tool for OLAP applications. Because a dimensional model may contain a very large amount of data, the often-requested crossings can be aggregated in SAS data sets or in MDDBs, allowing optimal response times. As you drill down in your data, the MDP may choose the star schema as the most appropriate data source for a query.

Generally speaking, star schemas allow better performance as the query result gets smaller. If a query is submitted on a star schema for which all data in the schema is needed, performance will severely degrade. In these cases, the use of summary tables or MDDBs will perform much better. For optimal performance, you can use a combination of data sets, MDDBs, and a star schema.

DIMENSIONAL QUERYING

There are several strategies for resolving star or snowflake schema queries. The response times you achieve will depend on the query itself and on the strategy you choose. Some often-used strategies are:

- *Table scans*. Using this method, the entire fact table is scanned. Only those rows that comply to the where clause are output to the result table.

- *Using indexed table lookups*. With this method, the following steps are taken:

  1. Choose one dimension table that has a condition on it and subset it.
  2. Join the result of step 1 with the fact table through index lookup.
  3. Join the result of step 2 with any other dimension tables that have conditions on them using index lookup or hashing and output only those rows that comply to the conditions.

  This method requires simple B-tree indexes on the primary keys in the dimension tables and on the foreign keys in the fact table. It is the most-used strategy in dimensional querying.

- *Using composite index lookups*. In this method, a Cartesian product is made of all dimension tables that have conditions on them. The result containing all possible key combinations is used to extract the rows from the fact table, using the appropriate composite fact table index.

- *Using bitmap indexes*. When subsets are requested on low-cardinality columns, the use of bitmap indexes can enhance performance significantly over the use of B-tree indexes.

- *Using a combination of simple, composite, and/or bitmap indexes*. This method combines all available indexes to compose a list of observation numbers to fetch from the fact table.

Not all of the strategies listed above are available in every environment. For instance, BASE SAS Software does not provide bitmap indexes. Also, without tweaking your SQL code, SAS SQL will never use the method of composite index lookups because it will always try to avoid creating any Cartesian product.

DIMENSIONAL QUERY PERFORMANCE

The most important factor in dimensional query performance is the number of times the hard disk is accessed for resolving a query. The more accesses made, the slower the query will run. Therefore, optimization of dimensional query performance focuses on minimizing disk reads and writes. This can be translated to, to a certain extent, minimizing the number of rows fetched from the fact table.

As an example, if you wanted to know the total number of cars sold to female customers on January 3, 2000, you could write the following SQL query:

```
select sum(number_sold) as number_sold
from   fact_sales          a
,      dimension_date      b
,      dimension_customer  c
where  b.date = '03jan2000'd
and    c.sex = 'F'
and    a.date_key = b.date_key
and    a.customer_key = c.customer_key
;
```

If the SAS SQL optimizer processes this query as follows:

1. Obtain all CUSTOMER_KEYs through subsetting DIMENSION_CUSTOMER on SEX = 'F'
2. Join the result of step 1 with FACT_SALES
3. Join the result of step 2 with DIMENSION_DATE and subset the result on DATE = '03jan2000'd.

the query will not perform very well because, assuming half of your customers are male and the other half female, FACT_SALES would be joined with half of all CUSTOMER_KEYs, which would then be joined with DIMENSION_DATE to obtain only those sales dated '03jan2000'd

If the optimizer selects:

1. Obtain all DATE_KEYs through subsetting DIMENSION_DATE on DATE = '03jan2000'd
2. Join the result of step 1 with FACT_SALES
3. Join the result of step 2 with DIMENSION_CUSTOMER and subset the result on SEX = 'F'.

the query will perform much better. Assuming '03jan2000'd is only one of many dates on which a car was sold, FACT_SALES would only be joined with a few DATE_KEYs, which would then be joined with DIMENSION_CUSTOMER to obtain only those sales to customers with SEX = 'F'.

The amount of fact table disk read operations can sometimes be further decreased using one of the other strategies mentioned before. This depends on the type of query, the columns that have conditions on them, the conditions themselves, the available indexes, the table sort order, data set page size, and others.

CHOOSING THE BEST QUERY STRATEGY

Not all query strategies are available in every environment. Also, not all of them perform evenly well in every case. Without discussing all the pros and cons of each strategy, table scans, for instance, are often a 'last resort' if all other strategies fail to deliver good performance. Composite index lookups can be very fast, but when the Cartesian product that is created becomes large, this method can result in terrible response times.

Using BASE SAS software and PROC SQL, you are limited to the indexed table lookup, the composite index lookup, and the table scan. In all three cases, you must depend on the SAS SQL Optimizer and hope it makes the right decisions in resolving your query.

If you use BASE SAS Software and the DATA step, you can use the same strategies as you can with PROC SQL. The difference with PROC SQL is that for the DATA step, you have to program the decisions that the SAS SQL Optimizer would make for you in PROC SQL, yourself. If you would know on forehand what the best decisions were, this would be an advantage. You could write your DATA step so that it would perform best. In many cases, this would do a better job than PROC SQL. If you wouldn't know what the best decisions were, PROC SQL would probably do a better job.

Apart from the three query strategies above, if you use the DATA step SET statement with the POINT= option, you can read the fact table using random (direct) access by observation number. If you could assemble a list with the observation numbers you require from the fact table and you use the POINT= option to fetch them, that would give you the best performance, provided you can assemble the observation number list in a reasonable amount of time.

Using the DATA step, you could also create your own bitmap indexes and use those in assembling the observation number list.

BITMAP INDEXES

Where B-tree indexes are most appropriate for columns of high cardinality (many different values), bitmap indexes are most appropriate for columns of low cardinality. Typical examples of such columns are gender codes and yes/no indicators. The basic idea behind a bitmap index is to create a bit string where every single bit indicates if a certain row in a table has a certain value or not.

Some RDBMSs provide bitmap indexes. Base SAS Software does not. This does not mean that bitmap indexing techniques cannot be used within the SAS System. Using the DATA step, you can create and exploit your own bitmap indexes.

As an example of bitmap indexing, take the table depicted in Table 1. This table has five rows and two columns with low cardinality.

Table 1. Sample table.

Obsnum	Gender	Prospect
1	M	Y
2	F	Y
3	F	N
4	?	N
5	M	Y

Table 2 shows the bitmap indexes on columns Gender and Prospect. Every row in the bitmap indexes corresponds to a row in the sample table. The bits in each row in the bitmap indexes indicate whether the sample table row has the value specified in the bitmap index column header, where '1' indicates 'yes' and '0' indicates 'no'. So, from the first row in the bitmap indexes in Table 2 you can see that row 1 in the sample table has Gender 'M' and Prospect 'Y', and that row 2 has Gender 'F' and Prospect 'Y'.

Table 2. Bitmap indexes on Gender and Prospect.

	Gender			Prospect	
Obsnum	M	F	?	Y	N
1	1	0	0	1	0
2	0	1	0	1	0
3	0	1	0	0	1
4	0	0	1	0	1
5	1	0	0	1	0

Querying using a bitmap index works as follows. If, for instance, all rows are requested from the sample table that have gender = 'F', the Gender bitmap index can be used. By sequentially scanning the index and checking which rows have the 'F' bit set to one, all necessary sample table rows are obtained and can be fetched from the sample table.

If all rows are requested that have gender = 'F' and prospect = 'Y', the Gender and Prospect bitmap indexes can easily be combined. Each row for which the bitwise logical AND of the row value and the bit string '01010' returns TRUE, complies to the condition and can be fetched from the sample table. This is where the strength of bitmap indexes lie. Especially when a query's conditions include limits on several bitmap indexed columns, the separate bitmap indexes can be combined to return a relatively small subset of rows that need to be fetched from the sample table.

Physically, the bits in the rows in all bitmap indexes are combined and stored as numeric values. The bitmap indexes in Table 2 could be stored as follows:

Table 3. Physical storage of the bitmap index.

Obsnum	Value
1	18
2	10
3	9
4	5
5	18

The reason why sequentially scanning the bitmap index is much faster than sequentially scanning the sample table or than using a B-tree index is that the bitmap index is generally very small. From Paul Dorfman's SUGI25 paper [3], an 8 byte numeric SAS variable can store 56 bits under OS/390 and 52 bits under Windows NT. This means that a bitmap index on column Gender with three distinctive values 'F', 'M', and '?' on a 50,000 row table will be about 23 KB, compared to around 400 KB for a B-tree index.

Because several bitmap indexes can be combined in a single physical file (data set), the method of storing the indexes as shown in Table 2 may not always be efficient. An alternative is to store it in a transposed format. This format ensures that with a single disk read, you obtain as much index information as possible. This format is shown in Table 4.

Table 4. Transposed bitmap index.

	Obsnum -->	1	2	3	4	5
Gender	M	1	0	0	0	1
	F	0	1	1	0	0
	?	0	0	0	1	0
Prospect	Y	1	1	0	0	1
	N	0	0	1	1	0

THE STAR CLASS

Because the optimal method of resolving a dimensional query depends on so many factors, and since there are many different query strategies available, the Star Class was developed. The Star Class is a SAS/AF® software class that manages star and snowflake schemas and provides facilities for querying them efficiently. It creates 'intelligent' data steps for every separate query, using table scans, indexed table lookups, composite index lookups, bitmap indexes, or combinations of the above where appropriate, to deliver fast dimensional query performance. Figure 3 depicts the Star Class architecture.

Methods of the Star Class are exposed through a SAS macro interface, which makes them easily accessible outside the SAS/AF environment.

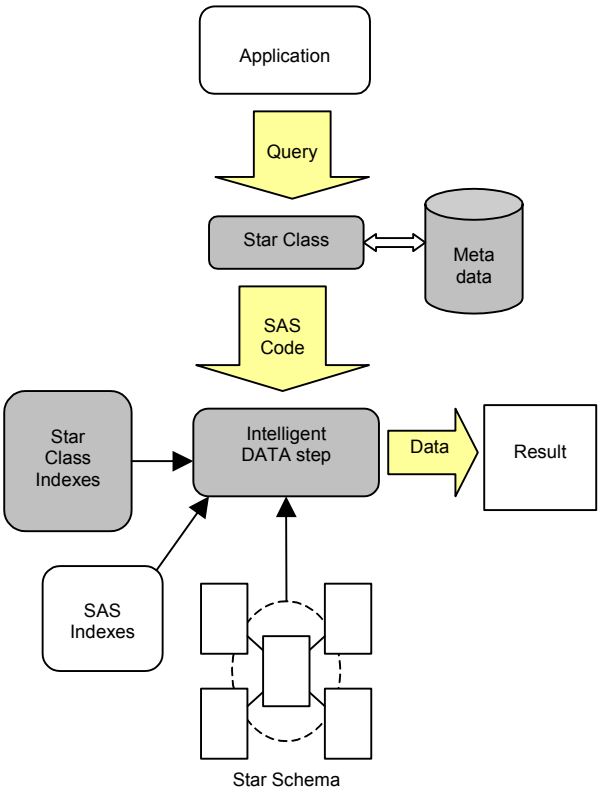

Figure 3. Star Class Architecture.

STAR CLASS METADATA

For the Star Class to be able to manage a star or a snowflake schema, it needs metadata on the schema. This metadata is acquired in three phases:

1. The Registration Phase
2. The Indexing Phase
3. The Analysis Phase

These phases are described below.

THE REGISTRATION PHASE

In the Registration Phase, the Star Class finds out the structure of the schema. It identifies the fact table, the dimension tables and primary and foreign keys. This is done, similar to the SAS/EIS Distributed Dimensional Metadata facility, through an SQL view on the schema, which is then registered.

For example, using the Car Sales Star Schema:

Contents of library:

```
fact_sales              (data)
dimension_sales_pers    (data)
dimension_customer      (data)
dimension_product       (data)
dimension_date          (data)
```

You create a view *star_schema_sales*:

```
proc sql;

 create view star_schema_sales as
  select *
  from   fact_sales            a
  ,      dimension_sales_pers  b
  ,      dimension_customer    c
  ,      dimension_product     d
  ,      dimension_date        e
  where  a.sales_pers_key = b.sales_pers_key
  and    a.customer_key   = c.customer_key
  and    a.product_key    = d.product_key
  and    a.date_key       = e.date_key
 ;

quit;
```

You start off the registration phase by submitting the following code from the program editor:

```
%starreg(schema = star_schema_sales)
```

You only execute this step when the structure of the star schema changes.

THE INDEXING PHASE

To enhance query performance, the Star Class creates and/or updates additional B-tree and/or bitmap indexes for your star or snowflake schema. To do this, you submit the following code:

```
%staridx(schema = star_schema_sales)
```

You must execute this step every time the data in you star schema has changed.

THE ANALYSIS PHASE

To enable the Star Class to generate 'intelligent' DATA steps, it needs to gather information about the contents of the data in your star or snowflake schema, such as cardinality, the distribution of values in the data, and the availability of indexes. You start this process through submitting the following code from the program editor:

```
%staranly(schema = star_schema_sales)
```

You must execute this step every time the data in you star schema has changed.

QUERYING SCHEMAS USING THE STAR CLASS

To query the star or snowflake schema using the Star Class, you submit the `%starqry` macro from the program editor. For example, to obtain a result data set containing the total number of cars sold to women on January 3, 2000, you submit the following code:

```
%starqry(schema  = star_schema_sales
        ,selvars = sum(number_sold)
        ,grpvars =
        ,where   = (date = ('03jan2000'd)
                    sex = ('F'))
        )
```

Code like this is easy to write and can also be generated easily by code generators.

BENCHMARKING

PROC SQL with simple primary and foreign key indexing is the most-used method for querying star or snowflake schemas in SAS. Therefore, to evaluate the performance of the Star Class, it is compared to PROC SQL. For this benchmark, a real life star schema was used with 3 million facts and 5 dimension. The

queries shown in the table were a random selection of often made end user queries.

Table 5. Benchmarking.

Query	PROC SQL	Star Class
1	2.97	2.29
2	1.09	1.21
3	1:18.92	0.59
4	3.28	5.12
5	1:28.42	42.30
6	30.75	0.67
7	1:03.90	19.78

From these results, you can see that sometimes, the Star Class is much faster than PROC SQL is. In other cases the Star Class and PROC SQL perform about equally and in other cases, the Star Class is slower than PROC SQL. This can happen when the strategy that PROC SQL uses by default, is the fastest strategy available. In such cases, the Star Class is a bit slower because of the overhead concerned with finding the best strategy.

USE OF THE STAR CLASS WITH SAS/EIS AND OTHER SAS MODULES

As mentioned before, starting SAS version 6.12, star schemas can be used as a data source in SAS/EIS. The Star Class provides the SCL code for two method overrides that allow de SAS/EIS Multidimensional Data Provider (MDP) to use the Star Class for both regular queries and for SAS/EIS Detail Data. Because of the rather inefficient way that standard SAS/EIS handles Detail Data, especially on data that resides on remote servers, the use of the Star Class greatly enhances its performance.

Since the Star Class can be used by the SAS/EIS MDP, it can be used in any SAS module that supports Distributed Dimensional Metadata. Examples of this are:

- The Open OLAP Server
- The SAS/SHARE Data Provider
- Enterprise Reporter software version 2.0 and higher
- Enterprise Guide software

CONCLUSION

Dimensional modeling is an often-used technique for building Data Warehouses. Star and Snowflake Schemas generally allow for fast query performance. This performance however, greatly depends on the query strategy that is used to resolve a query.

Since no two queries are the same and the performance of a dimensional query depends on many factors, determining which query strategy is the best for a particular query is not an easy task. The SQL Procedure does its best, but cannot always guarantee good response times.

The Star Class supports more query strategies than PROC SQL and can make use of simple, composite, and bitmap indexes, and combinations of those. In addition, it gathers information about the structure and contents of the dimensional model. This allows it to generate very efficient DATA step code for resolving queries, often resulting in great performance benefits.

The Star Class can be integrated in the SAS/EIS Multidimensional Data Provider, which enables it to be used in any application that can access multidimensional SAS data.

REFERENCES

[1] Kimball, R., ed. (1998), *The Data Warehouse Lifecycle Toolkit*, John Wiley & Sons, Inc.

[2] Shepard, M (1999), *Building Star Schema With SAS® Software*, SEUGI18 Proceedings.

[3] Dorfman, P.M. (2000), *Private Detectives In A Data Warehouse: Key-Indexing, Bitmapping, And Hashing*, SUGI25 Proceedings.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Bart Heinsius
E.O.M. Data
Oude Enghweg 16
1217 JC Hilversum
The Netherlands
Work Phone: +31 35 6220088
Fax: +31 35 6220106
Email: Bart.Heinsius@eomdata.nl
Web: http://www.eomdata.nl