Paper 304-2010

SAS® Scalable Performance Data Server®: Star Schema Support

Daniel M. Sargent, SAS Institute, Cary, NC

ABSTRACT

The release of SAS Scalable Performance Data Server 4.4 included SQL planner and execution optimizations for star schemas. This paper will cover important areas relevant to the SAS Scalable Performance Data Server star schema support:

1. basic concepts behind the star schema
2. overview of customer needs for star schemas
3. how the query is rewritten to enhance performance
4. limitation of the star schema support
5. performance matrix comparing different execution methods to star schema
6. Configuration and limitation

STAR SCHEMA

A star schema is a physical organization of data tables common to data warehouses or data marts. This section will cover important aspects of a Star Schema:

1. Types of tables in a Star Schema

Fact tables
Dimension tables

2. Relationship between fact and dimension tables
3. Star schema or De-normalization of table
4. Subsetting of tables in a star schema
    1. From the dimension tables to the fact table
    2. Rewriting the query to subset the fact table

THE TABLES OF A STAR SCHEMA

There are two types of tables in a star schema: fact table(s) and dimension tables. A star schema can have more than one fact table depending on the business rules and the queries that the star schema will support. The second type of table in a star schema is called a dimension table. In star schemas, there has to be least two, but most often more than two, dimension tables. This section will provide a better understanding of the two types of tables.

FACT TABLES

A fact table is the central table of a star schema which contains the historical transactional facts of the business. Fact tables contain facts about the business, information about when a transaction occurred, what product was sold, how many items were sold, how much they sold for, how they were sold (retail vs. Internet), where they were sold, who purchased the item, and what supplier provided the products to be sold. Each fact of the business in a star schema is identified by a unique combination of codes for each row contained in a set of columns of the fact table which is called the primary key. The codes in the columns of the primary key also exist uniquely in the dimension tables. This relationship between codes between the two types of tables is explained in more detail in the section covering the relationship between the fact and dimension tables.

The rows in a fact table are often summarized to reduce the physical size of the table. Although fine details of transactions are lost using summarization, the information is detailed enough to provide the information to operate the business.

For example, a grocery store does not always need to access the individual transactions for each watermelon sold on a specific day. However, the total number sold each day is important for inventory and

restocking purposes. The summarized row will contain information about the number of watermelons sold and the total price paid. If a grocery store sold 30 watermelons in a single day to 30 different customers and the fact table was to contain no summarized data compared to data summarized daily, the table would contain 30 rows compared to 1. Repeat this for each product in the fact table and the physical reduction in the size of the fact table can be significant.

An entire week for watermelon sales in a single store would require 7 rows of data in the fact table. We can use the same example of a store selling 30 watermelons a day to show the table-size reduction for a week. If 30 watermelons are sold each day of the week, the total number of watermelons sold each week will be 210. When these rows are summarized to a weekly level from detailed level, they would use only about 1/ 210 of the original space required. This is the main purpose for summarizing transactions in the fact table.

DIMENSION TABLES

The second type of table in a star schema is the dimension table. A star schema requires two or more. A dimension table most often does not have a relationship to any other dimension tables in the star schema. The few cases where dimension tables can be related often are where the relationship is a hierarchy. For example in the financial services industry, a star schema could have a dimension table for a household and also a dimension table for each cardholder. The relationship between the two tables is that a set of cardholders can be from the same household and the household key would be a part of the cardholder key. (Keys are discussed in the next section.) Dimension tables contain the full set of descriptor information for the codes that make up the primary keys. Descriptor information will include columns such as names, addresses, text descriptions, geographic information, and so forth. The primary key codes are contained in a column or a set of columns. The column or set of columns comprise a unique identifier for a given row in the dimension table. This set of unique identifiers is called the primary key of the dimension table.

Because the values that are contained in the primary keys of the star schema tables are often just a numeric value they provide little to no useful information for a user of the data. Granted a knowledgeable may know the data – to properly analyze the data they will need information stored in the dimension tables. The dimension tables is where information such as store location, state it is located in, region, ZIP code, size in square feet, phone number, year opened, population demographics for the store location, and so forth are stored. This important information (that is stored in the dimension tables) is linked to the fact table through a code value in one of the columns of the fact table primary key. Another example would be the UPC code: this code is a 12-digit number. Unless the user or business analysis knows the code and what product it is for, the code will provide very little helpful information.

A star schema will have a dimension table that contains all the possible UPC codes that the business sells. This dimension table will include columns with descriptor information about the code and categorical information that is understood by users and business analysis. The descriptor information of a dimension table often contains a field(s) with full text descriptions that describe the code. A good example would be UPC codes for different types of cereals; each UPC code would include the size of the box.

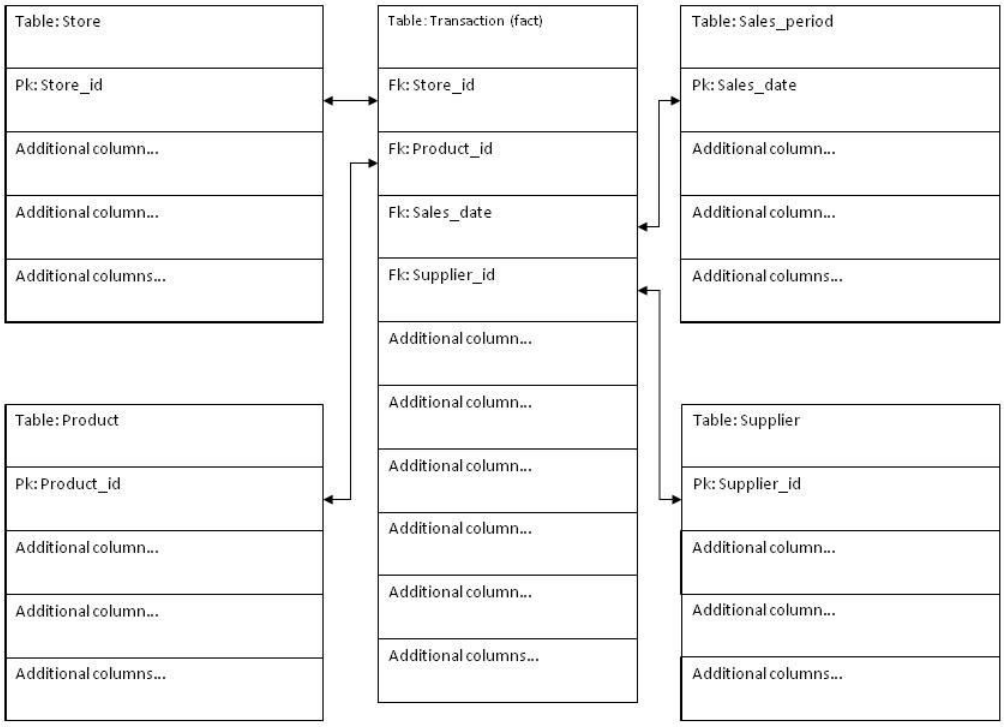
PRIMARY AND FOREIGN KEYS

The tables in a star schema contain a column or set of columns that are called table keys. There are two types of table keys: primary and foreign. The primary key of a table is the column or set of columns where the value for each row of the table is unique and is the identifier for that row. In a star schema, the primary key of the fact table is the combined set of primary key columns from the dimension tables. These columns that link the fact and dimension tables are called foreign keys.

The foreign keys of the fact table are the link between the fact and dimension tables. In a star schema, only the fact table contains foreign keys with a foreign key for each dimension table. Because the relationship of the foreign keys links to the primary key of the dimension tables, each foreign key value must match to a primary key value in the dimension table. This is called a constraint. The SAS Scalable Performance Data Server star-schema does not enforce the constraint values between the tables; it is up to the ETL process to assure the rows match.

For example, take a retail star-schema with four-dimension tables: STORE, PRODUCT, SALES_PERIOD, and SUPPLIER. The dimension table STORE has a primary key of STORE_ID, the PRODUCT table has a primary key of PRODUCT_ID, the SALES_PERIOD has a primary key of SALE_DATE, and the table SUPPLIER has a primary key SUPPLIER_ID. The fact table will have a four-column primary key where the four columns in the fact table are foreign keys, and each column will map to a corresponding primary key of the one of the dimension tables.

Star Schema Diagram:



The values in these four columns will match the values of the primary keys for the dimension tables. The column STORE_ID in the fact table will contain the same set of values as the STORE_ID in the STORE dimension table, the difference being, as a foreign key in the fact table; the values are almost always repeated. Even though the values in a foreign key column might be repeated, the three values across the primary key of the fact table must be unique. Example: The STORE dimension table has a row with a key value of 10456, the PRODUCT table has a row with a key value of 10045, and the SALES_PERIOD table has a row with a key value for the sales week ending on 04JAN2009. The combination of these values can be used only once as the identifying primary key values for a row in the fact table.

STAR SCHEMA AND DATA NORMALIZATION

The organization of data is central to the Star Schema because it achieves a reduction of the physical footprint of data tables on disk by elimination of duplication. Below is an example set of tables common to a retail business organized in a star schema. The table provides the number or rows for each table, width of the rows, and a computed physical size in gigabytes. A star schema for retail business can contain other tables; for the purpose of explaining the physical size of a star schema, this paper will limit three tables. In a star schema, the only duplication of data are the columns used that map rows between the fact table and the dimension tables. Even with a central fact table that contains 1 billion rows, the total space required to physically store the data is less than 100 gigabytes. The physical size that has been computed does not include indexes or take into account possible data compression.

Table	Number of rows	Row width	Size
CUSTOMER_TRANS (Fact)	1,000,000,000	100	93 gigabytes
STORE (dimension)	1,000	600	586K
SALES_PERIOD (dimension)	208	200	40K
PRODUCT (dimension)	10,000	400	3906K
All combined tables	N/A	N/A	Less than 94 gigabytes

Because descriptor and categorical information are located in columns of the dimension tables of a star schema, the subsetting of rows will also occur through the dimension tables. With the fact of the business for a star schema located in the columns of the fact table, this means that almost every query will involve a join of at least two tables in a star schema. After the subsetting is done on the dimension tables, they are joined to the central fact table. A later section will discuss the star schema optimization used which involves subsetting of the fact table. Subsetting and joining of dimension tables to a fact table requires computer resources.

The queries that are submitted to run against a star schema will have similar patterns of subsetting from the dimension tables and then joining those results to the fact table. Because of these patterns, it is reasonable to consider alternative ways to organize data. One alternative is to de-normalize the tables. De-normalization is a method of using tables that will allow queries to access fewer tables or a single table that can provide the same result as the queries running against a star schema would provide without having to perform any joins. De-normalization of tables can greatly improve query performance. However, the physical footprint of the data will increase significantly because of the duplication of data in the pre-joined table.

In a star schema, the only data columns that are duplicated are the columns needed to join tables. With de-normalized of tables additional, often all, columns are represented on the fact table. Representing additional columns on the fact tables shifts the location of subsetting from occurring on a dimension table to the de-normalized table and eliminates the need to join tables. De-normalizing tables needs to be weighed carefully against the process of loading tables and the physical footprint. The chart below shows the physical footprint of: first, the SALES_PERIOD dimension table; second, the STORE; and third, the PRODUCT dimension tables being de-normalized with the fact table. The number of rows in the fact table remains at 1 billion but the row width will increase. The table below shows this change and computes a new size.

Table	Number of rows	Row width	Size
CUSTOMER_TRANS (Fact)	1,000,000,000	100	93 gigabytes
CUSTOMER_TRANS and SALES_PERIOD	1,000,000,000	300	279 gigabytes
CUSTOMER_TRANS, SALES_PERIOD, and STORE	1,000,000,000	900	837 gigabytes
CUSTOMER_TRANS, SALES_PERIOD, STORE, and PRODUCT	1,000,000,000	1300	1,209 gigabytes
Star schema	N/A	N/A	Less than 94 gigabytes

From the table above, we can see that the physical size (excluding indexes or not accounting for compression) of the single de-normalized table results in a footprint approximately 13 times larger than if the data were organized in a star schema.

In a star schema, to bring the necessary columns together into a single table for subsetting the fact table, the software must perform a join of the tables requested in the query. A paper given at the SAS Global Forum in 2008 (292-2008) demonstrated the costs for the type of processing that are needed to join tables. With a star schema, it will involve first subsetting and then sorting, followed by match-merging or index points, to bring together the necessary columns of the tables. This type of process will be performed for every dimension tables involved in a join.

A data management technique used by customers is to de-normalize tables into fewer tables to make subsetting easier, plus reduce sorts, match merging, or index points needed. De-normalization is organizing data tables by pre-joining two or more tables so that the columns necessary would be in fewer tables which will reduce the joining and increase query response time.

Like everything, there is a trade off to using de-normalized tables: De-normalized tables it will require far more disk space for the data tables and indexes. And, the de-normalized tables can be more difficult to load and manage. This paper will discuss only the trade-off of disk space. To demonstrate the change in disk space utilization for de-normalized tables, we will start with the fact table and the smallest dimension table in the star schema and combine them into a single table. These two tables are CUSTOMER_TRANS and SALES_PERIOD tables. The number of rows in the de-normalized table will match the number of rows in the star schema fact table at 1 billion rows. Therefore the physical size of the new table will be larger due to the additional columns that need to be added. Without reducing the width of the combined table by 8 bytes to account for the duplicated column in the two tables that joins the two tables from the star schema together, if all columns from both tables were to be stored in single de-normalized table that represented the joined product it would be 300 bytes wide, or three times the size of the original fact table (93 gigabytes). The physical size of the new table will be 279 gigabytes, not including indexes.

Knowing the number of rows and width of the tables in our example star schema, we can compute the physical space needed to store a de-normalized table. We can start with the fact table and the smallest dimension table in the star schema. These two tables are CUSTOMER_TRANS and SALES_PERIOD tables. Combined together if all the columns from both tables were to be kept, the new table would be 300 bytes wide. The two tables have a column that shares matching values in which a star schema matches rows in the fact table to a single row in the dimension table. If the two tables are de-normalized, then this redundant data column can be eliminated making the table width 292 bytes. When we compute the new size of this table, it will be 273 gigabytes. The size of the new table is about three times the size of the original

two tables which were less than 94 gigabytes. This is a significant increase in table size, and we can start to understand why de-normalizing of tables is not always possible.

Continuing on, we can compute a final table size if the entire star schema is de-normalized into a single table. The next two tables are 400 and 600 bytes wide with only two columns having duplicate data between the tables. If these tables are de-normalized with the fact table, they will add 392 bytes and 592 bytes to the width of the de-normalized table. The final de-normalized table will be 1276 bytes wide and will contain 1 billion rows. The physical uncompressed size of this table, not including indexes, will be 1.16 terabytes. This final table is over 10 times larger than the total physical size of the star schema.

REWRITING OF THE QUERIES

One of the strengths of SAS Scalable Performance Data Server is its ability to subset data very quickly and efficiently. The organization of a star schema causes the subsetting of rows to occur on the dimension tables where the descriptor, demographic, and geographic columns are located. With pair-wise joins the subsetting of the fact table occurs when a dimension tables is joined to the fact table. This is performed via the join keys one table at a time. In a query with four dimension tables there will be four pair-wise joins performed. During the pair-wise join process when a dimension table is joined to the fact table intermediate results are saved. Then, another dimension table is joined to these intermediate results and those are saved. This process of joining a dimension table with the fact table and saving intermediate results will continue until all the requested dimension tables have been joined with the fact table.

When the SQL planner chooses the order of dimension tables to join to the fact table, it will (based on heuristics) try to choose the dimension table that will result in the smallest possible result set of rows compared to the other dimension tables. This is done to create minimize the size of the intermediate results which reduces the work necessary to join the next dimension table. As an example take a star schema with a fact table that contains 500 million rows, subsetting on four dimension tables where the heuristics indicate the joins of the dimension tables to the fact table will result in 20%, 16%, 23%, and 34% of the rows being selected joined to the fact table respectively, and the final result from joining all four dimension tables to the fact table return only 4% of the rows possible from the fact table. Selecting the dimension table that when joined to the fact table will return only 16% of the rows, the size of the scope of the remaining joins has been reduced as a much as possible.

As we can see in this example, the problem with this approach is after all dimension tables have been joined to the fact table the actual number of rows returned could be far less than the 16% returned from the first pair of tables joined. This is the key reason that pairwise joins are not always optimal for performing a join in a star schema.

The star schema facility optimizes the query by constructing new WHERE clauses predicates using the dimension tables which are included in a rewritten query, and expedite subsetting of the fact table. The additional WHERE clauses allow the execution of the query to subset the fact table down to the smallest set of rows required for the query to be executed. In other words in our earlier example rather than having to execute a four-way pairwise join to retrieve the 4% of rows for the query, the subsetting of the fact table will occur with a single pass. This single pass returns the 4% of rows necessary to resolve the query. This allows the processing of the query to be performed in 3 steps: subsetting of the dimension tables, subsetting of the fact table, and then using a single pass of the remaining fact table rows to probe back, using indexes, to the dimension tables to gather any columns necessary for the final result.

With a star schema, this is possible because the relationships between the fact table and the dimension tables are known. During the first phase of executing the query, the subsetting of the dimension tables

occurs. The primary keys of the dimension tables provide the sets of unique values that are required to create a WHERE clause that can subset the fact table to the smallest possible set of rows required to complete the query. Below is an example of what the query will look like before and after having the subsetting WHERE clause for the fact table applied.

INITIAL QUERY

```
PROC SQL ;
  Create table dans_table as
  Select f.store_id as store_id,
         f.units_sold as units_sold,
         f.total_revenue as total_revenue,
         f.product_id as product_id,
         sa.sales_period as sales_period,
         p.product__id as product_id,
         p.product_name as product_name,
         p.product_type as product_type
         st.store_id as store_id,
         st.state as state,
         st.store_size as store_size
   From spdslib.customer_trans f,
         spdslib.store st,
         spdslib.sales_period sa,
         spdslib.product p
  Where f.store_id     = st.store_id
    and f.product_id   = p.product_id
    and f.sales_period = sa.sales_period
    and st.state in ('CT','NC')
    and st.store_size  >   40000   /* sq/ft */
    and p.product_type in ('PAPER GOODS')
    and sa.sales_year = '2009'
    and sa.day_of_week = 7 /* saturday */
   ;
quit ;
```

The queries where clause can be split into two sections that provide a map between the tables and provide the subsetting criteria. The star schema optimization will look at this map to assure that there is a central table that all other tables are being joined to. When the planner recognizes a query with tables arranged in a star schema, it will avoid the pair-wise join method and optimize the processing for subsetting of the fact table. The section of the WHERE clause below shows that the three dimension tables are being joined to a single fact table.

```
Where f.store_id     = st.store_id
  and f.product_id   = p.product_id
  and f.sales_period = sa.sales_period
```

The second section of the WHERE clause contains subsetting criteria that are applied to the dimension tables in the star schema. Below, we can see the section of the WHERE clauses for the three dimension tables.

```
and st.state in ('CT','NC')
and st.store_size  >   40000   /* sq/ft */
and p.product_type in ('PAPER GOODS')
and sa.sales_year = '2009'
```

```
    and sa.day_of_week = 7 /* saturday */
```

REWRITTEN QUERY

```
proc sql ;
   create table dans_table as
   select f.store_id as store_id,
          f.units_sold as units_sold,
          f.total_revenue as total_revenue,
          f.product_id as product_id,
          sa.sales_period as sales_period,
          p.product__id as product_id,
          p.product_name as product_name,
          p.product_type as product_type
          st.store_id as store_id,
          st.state as state,
          st.store_size as store_size
     from spdslib.customer_trans f,
          spdslib.store st,
          spdslib.sales_period sa,
          spdslib.product p
    where f.store_id     = st.store_id
      and f.product_id   = p.product_id
      and f.sales_period = sa.sales_period
      and f.store_id in (store_id list from dimension table)
      and f.product_id in (product_id list from dimension table)
      and f.sales_period in (sales_period list from dimension
                             table)
      and st.state in ('CT','NC')
      and st.store_size  >   40000   /* sq/ft */
      and p.product_type in ('PAPER GOODS')
      and sa.sales_year = '2009'
      and sa.day_of_week = 7 /* saturday */
       ;
quit ;
```

PERFORMANCE OF THE STAR SCHEMA FACILITY

This section investigates the performance of the star schema facility. To measure the performance of the facility, two common star schemas created were created with mocked-up data for retail and financial customers. Each star schema contained a single central fact table and four dimension tables. The fact table for each schema has a four-column primary key with four numeric columns. The fact table for each schema is indexed using a simple index on the four primary key columns. Each dimension table will consist of three types of columns: a single column for the primary key, columns which are blank filled to represent additional descriptor columns, and a subsetting column that contains numeric values between 1 and 10. Each dimension table is indexed on the primary key column and the subsetting numeric column.

The numeric column used in the WHERE clause to subset the dimension tables is independent across the dimension tables. For example, the rows selected with the value of 1 from two different dimension tables when joined to the fact table will not result in the selection of 10% of the rows in the fact table.
The subsetting numeric column that contains numbers between 1 and 10 will be used to select subsets from 10% to 60% from each of the dimension tables using a WHERE clause with a single number or set of numbers. There were six queries executed against each star schema starting with a query that selected 10% of the rows from each dimension table and ending with a query that selected 60% of the rows from the dimension tables.

The performances of the two star schemas differ due to the differences in the number of rows in the two fact tables and the number of rows in the dimension tables. The retail schema has a much narrower fact table

but more than 5 times the number of rows. The ratio of rows between the fact table for each schema and the dimension tables for each schema will also cause differences in performances.

RETAIL STAR SCHEMA EXAMPLE

The retail star schema will consist of five tables: RETAIL_FACT, STORE, PRODUCT, SALES_PERIOD, and SUPPLIER. The RETAIL_FACT table is the central fact table and has a primary key of STORE, PRODUCT, SALES_PERIOD, and SUPPLIER. The fact table will consist of rows of transactions summarized for each week, product, store, and supplier. The four dimension tables are STORE, PRODUCT, SALES_PERIOD, and SUPPLIER. These four tables are linked to the fact table through a column that is part of the RETAIL_FACT primary key.

The fact table RETAIL_FACT contains 375 million rows with a table width of 100 bytes. It was created to match the cross product of the four dimension tables with 60, 100, 125, and 500 rows respectively. The RETAIL_FACT table also contains additional columns for units sold and transactional amounts.
The SALES_PERIOD table will contain 60 rows, and it will be 124 bytes wide. Each row in this table pertains to a weekly period during. There are 60 rows because the tables will contain five years of historical information about a monthly basis. So 4 years multiplied by 12 months in a year gives us 60 rows. The primary key to this table is a numeric column called SALES_PERIOD. This column will contain a date value that is the last day of each weekly SALES_PERIOD. Other columns that might be included on this table are season, month, quarter, and year.

The SUPPLIER dimension table will contain 100 rows with a table width of 536 bytes. Each row in this table pertains to a specific supplier of goods for the supermarket chain. The primary key for this table is a column called SUPPLIER. Other columns that this table would contain could be columns such as supplier names and address information.

The STORE dimension table will contain 125 rows, and the rows will have a width of 536 bytes. The primary key for this table is a numeric column called STORE. The other columns that would typically be found in this dimension table would be geographic information such as the street address, city, state, and ZIP code. Other columns could contain information about the square footage of the store and when the store was opened. These columns will be represented by large character-filled columns.

The PRODUCT table will contain 500 rows, and it will be 536 bytes wide. The primary key is a numeric column called PRODUCT. This table will represent all possible products sold by the grocery store chain. Some of the other columns on the table are product description, UPC code, quantity, and size.

QUERIES

There will be six queries executed against the retail star schema where each query selects a larger percentage of rows from each dimension table. The first query will select 10% of the rows from each dimension table. In this star schema setup, this does not mean 10% of the total rows across the entire schema will be selected. The queries were executed using the star schema facility and using pairwise joins as a comparison. Below is the syntax of the query.

```
proc sql _method ;
  connect to sasspds (dbq='bigdata'
                      server=&serv..&port
                      user='anonymous') ;

  execute (create table junk as
           select b.product as product,
                  c.sales_period as sales_period,
```

```
            d.supplier as supplier,
            e.store as store,
            a.units_sold as units_sold,
            a.sales_price as sales_price
        from retail_fact a,
            product b,
            sales_period c,
            supplier d,
            store e
        where a.product = b.product
          and a.sales_period = c.sales_period
          and a.supplier = d.supplier
          and a.store = e.store
          and b.product_subset_10 in (1)
          and c.sales_period_subset_10 in (1)
          and d.supplier_subset_10 in (1)
          and e.store_subset_10 in (1)
      ) by sasspds ;

  disconnect from sasspds ;
quit ;
```

RETAIL RESULTS

The results of queries executed with the star schema facility compared to pairwise join execution are in the table below. Because the star schema facility bypasses multiple sorts and merges necessary for pairwise joins and it rewrites the query to optimize the query by subsetting the fact table to the smallest possible row set, it has a distinct performance advantage. The results compare the two types of joins using the total time necessary to execute the query along with the total CPU utilization.

Percent Subset of dimension tables	Rows returned by query	Run time Star schema	CPU-utilized Star schema	Run time Pairwise join	CPU-utilized Pairwise join
10 percent	39,000	13 sec	13 sec	22 min	42 min
20 percent	624,000	48 sec	49 sec	37 min	74 min
30 percent	3,159,000	4 min	4 min	58 min	97 min
40 percent	9,984,000	12 min	12 min	86 min	132 min
50 percent	24,375,000	30 min	30 min	120 min	164 min
60 percent	49,896,000	60 min	61 min	160 min	210 min

CREDIT CARD STAR SCHEMA

The credit card star schema will consist of five tables: FINANCIAL_FACT, HOUSE_HOLD, CARDHOLDER, TRANS_PERIOD, and VENDOR. The FINANCIAL_FACT table is the central fact table and has primary keys of HOUSEHOLD, CARDHOLDER, TRANS_PERIOD, and VENDOR. The fact table will consist of rows of transactions summarized for each week, product, store, and supplier. The four dimension tables are HOUSE_HOLD, CARDHOLDER, TRANS_PERIOD, and VENDOR. These four tables are linked to the fact table through one of the columns that is part of the FINANCIAL_FACT primary key.

The fact table FINANCIAL_FACT contains 72,009,720 million rows with a table width of 552 bytes. It was created to match the cross product of the four dimension tables which have 24, 1000000, 1000000, and 3000405 rows respectively. The FINANCIAL_FACT table also contains an additional column for the transaction amount.

The TRANS_PERIOD table will contain 24 rows, and it will be 124 bytes wide. Each row in this table pertains to a monthly time period. There are 24 rows because the tables will contain two years of historical information on a monthly basis. The primary key to this table is a numeric column called TRANS_PERIOD.

This column will contain a date value that is the last day of each monthly TRANS_PERIOD. Other columns that might be included on this table are season, month, quarter, and year.

The VENDOR dimension table will contain 1000000 rows with a table width of 512 bytes. Each row in this table pertains to a specific vendor of goods that the CARDHOLDER purchased. The primary key for this table is a column called VENDOR. Other columns that this table could be columns such as vendor name and address information.

The HOUSEHOLD dimension table will contain 1000000 rows, and the rows will have a width of 512 bytes. The primary key for this table is a numeric column called HOUSEHOLD. The other columns that would typically be found in this dimension table would be geographic information such as the street address, city, state, and ZIP code. Other columns could contain information about the household such as the number of members in it and income level.

The CARDHOLDER table will contain 3000405 rows, and it will be 520 bytes wide. The primary key is a numeric column called CARDHOLDER. This table will represent every CARDHOLDER that has been issued a credit card. Some of the other columns on the table are cardholder name, date of birth, address information, credit limit, education, and income.

QUERIES

There will be six queries executed against the retail star schema where each query selects a larger percentage of rows from each dimension table. The first query will select 10% of the rows from each dimension table; in this star schema setup, this does not mean10% of the total rows across the entire schema will be selected. The queries were executed using the star schema facility and using pairwise joins as a comparison. Below is the syntax of the query.

```
proc sql _method ;
   connect to sasspds (dbq="&domain"
                       server=sunflare.6150
                       user="anonymous") ;

   execute (create table &domain..star_results as
            select f.household as household,
                   f.cardholder as cardholder,
                   f.trans_period as trans_period,
                   f.vendor as vendor,
                   f.trans_amt as trans_amt,
                   h.household_info_s as household_info_s,
                   c.cardholder_info_s as cardholder_info_s,
                   t.trans_period_info_s as trans_period_info_s,
                   v.vendor_info_s as vendor_info_s
              from financial_fact f,
                   household h,
                   cardholder c,
                   trans_period t,
                   vendor v
              where f.household = h.household
                and f.cardholder = c.cardholder
                and f.vendor = v.vendor
                and f.trans_period = t.trans_period
                and c.cardholder_subset_10 in (1)
                and h.household_subset_10 in (1)
                and t.trans_period_subset_10 in (1)
                and v.vendor_subset_10 in (1)
            ) by sasspds ;
```

```
quit ;
```

FINANCIAL RESULTS

The results of queries executed with the star schema facility compared to pairwise join execution are in the table below. Because the star schema facility bypasses multiple sorts and merges necessary for pairwise joins and it rewrites the query to optimize the query by subsetting the fact table to the smallest possible row set, it has a distinct performance advantage. The results compares the two types of joins using the total time necessary to execute the query along with the total CPU utilization.

Percent Subset of dimension tables	Rows returned by query	Run time Star schema	CPU-utilized Star schema	Run time Pairwise join	CPU-utilized Pairwise join
10 percent	100,557	6 min	3 min	11 min	18 min
20 percent	760,587	12 min	5 min	16 min	25 min
30 percent	2,514,804	11 min	9 min	25 min	31 min
40 percent	5,878,686	13 min	15 min	32 min	41 min
50 percent	10,634,774	20 min	24 min	40 min	52 min
60 percent	17,425,430	30 min	37 min	53 min	66 min

CONFIGURATION AND LIMITATIONS

How to configure the data tables to make use of the star schema optimizations and limitations for the star schema facility can be found in the SAS SPD Server User guide, located online at the SAS tech support website.

CONTACT INFORMATION

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

Daniel Sargent
SAS
SAS Campus Drive
Cary, NC 27513
Work phone: 919-531-6477
Fax: 919-677-4444
E-mail: Daniel.Sargent@sas.com
Web: www.sas.com