Paper 033-2012

Beyond Star Schema: Exploring the Next Query Generation with the SAS® Enterprise BI Server

Fred Levine, SAS Institute Inc, Cary NC

ABSTRACT

Structured Query Language (SQL) generation is an essential function of the SAS Enterprise BI Server that enables consistent, self-service access to data for many users. This paper explores enhanced ways that the BI server generates "intelligent" queries with SAS® 9.3. The new Intelligent Query generator generates SQL based on complex dimensional and relational data models, while providing users with an easy and unique way of asking very specific questions about their data.

INTRODUCTION

Business intelligence (BI) data models used for querying and reporting can be extremely complex and can impose challenges in generating correct and efficient SQL. A prime example of this complexity is when an SQL query is generated from a BI data model that contains multiple measures that reside in different tables. For example, assume a business analyst needs to generate a report that shows sales and budget figures for each department in an organization where the sales and budget detail data live in separate tables (as is often the case in a large enterprise). The generated query needs to be constructed in such a way that eliminates the multiplicative effect of the many-to-many cardinality that exists between the sales and budget tables; incorrect results can occur if this cardinality effect is not taken into account.

The data model example described is often referred to as multiple fact tables (or multiple star schema) because there is more than a single fact table (sales and budget). This is in contrast to a basic star schema model that consists of a single fact table and one or more dimension tables.

Beginning in SAS 9.3, the new Intelligent Query generator can generate queries from multiple fact table models and also from complex relational models. The Intelligent Query generator is unique in that it always generates correct queries without needing to know anything specific about the underlying data model.

The ultimate purpose of the new model-independent query generator is to enable users to perform their jobs more efficiently. Users should be able to focus on asking questions about their data without needing to understand the structure of that data.

The remainder of this paper explains in more detail the multiple fact table problem that the Intelligent Query generator solves through the discussion of BI concepts, dimensional and relational data models, and inter-table cardinality. It also discusses the functionality of the Intelligent Query generator in SAS 9.3 as well as its future potential to construct very granular business questions in a novel way.

BUSINESS INTELLIGENCE CONCEPTS

Facts, categories, aggregations, measures, and filters are fundamental BI topics that underlie a BI data model.

A fact is a business metric that results from an event. A common example of a fact is a purchase that is stored as a specific dollar amount.

A category represents a specific attribute of a dimension. A dimension is a collection of attributes (or categories) that provides descriptive information about a fact. For example, customer is a dimension that might contain attributes such as name, address, age group, date of birth, and so on. Each of these attributes is stored as columns in a table. In a BI model, we refer to these attribute columns as categories. Categories allow users to specify how they want to view their data in a report. For example, a user might want to view the median housing cost by the categories of city and age group.

An aggregation is a summary operation that is applied to a group of facts for statistical analysis. In the preceding example, median is the aggregation that is applied to house sales as grouped by city and age group.

A measure is a fact column associated with an aggregation. MEDIAN(house_sale), SUM(car_sale), and AVG(purchase) are examples of measures.

A filter subsets the data returned from a query and can be applied to measures, categories, and facts (pre-aggregated detail data).

The aforementioned topics are the basic tools of a business analyst and are an integral part of any BI data model.

The Intelligent Query generator uses these BI topics as defined in a SAS Information Map to generate queries that answer business questions.

DATA MODELS

Data modeling can be a complex subject with regard to the structure of a data warehouse (the underlying transformed data store that facilitates data analysis for querying and reporting). For the purposes of this discussion, this paper focuses on data models as they are surfaced to users in the SAS Enterprise BI Server. Two basic types of data models are relevant to the Intelligent Query generator: dimensional and relational.

DIMENSIONAL MODELS

A dimensional model stores dimensions and facts in separate tables.

In a dimension table, each row represents a unique set of attributes for that dimension. For example, in a customer dimension, a row might look like the following:

CUST_ID	CUST_NAME	CUST_ADDRESS	CUST_AGE
1	Levine	10 Elm St, Akron, OH	39

Table 1. Sample Dimension Table Attributes

The customer "Levine" is uniquely identified by the CUST_ID of 1. The remaining customer rows in this table also have a unique customer ID.

In a fact table, each row represents an event (often a transaction) along with one or more unique dimension IDs and possibly other attributes that might be pertinent to the event. For example, in a "Purchase" fact table, a row might look like the following:

PURCHASE	CUST_ID	PRODUCT_ID	STORE_ID	TRANSACTION_ID
$20	1	47	8	10

Table 2. Sample Fact Table Row

In this example, the preceding row represents a $20 purchase for a particular customer, product, and store. The transaction ID is a concise way to uniquely represent this fact. Note that the various dimension IDs can occur multiple times within a fact table. That is, a customer can make many purchases, a product can be purchased many times, and many purchases can be made at the same store.

Stated mathematically, in a dimensional model, a dimension table has a one-to-many cardinality to a fact table. Cardinality is the number of times a value occurs in one table relative to another table.

The simplest dimensional model is a star schema (or data mart) that represents a specific business process as represented by a single fact table and multiple dimension tables. The preceding example is a star schema where "Purchase" is the business process (the "Purchase" fact table) and "Customer", "Product", and "Store" are the dimensions that provide context to the facts (or transactions) contained in the "Purchase" table.

Because an enterprise typically has many business processes that use the same dimensions, an enterprise-wide dimensional model is an integration of multiple data marts, all of which share the same dimensions. In an enterprise-level dimensional model, the dimensions that are common to all business processes are called conformed dimensions. This integrated collection of data marts in an organization comprises the data warehouse. The integration of data marts is important because users often need to ask questions that pertain to more than one business process. For example, a business analyst in a car dealership might want to know which customers leased cars and also purchased those cars. Leased car sales and purchased car sales are separate business processes that potentially have their own fact tables and also share the customer dimension.

This integrated data mart model is commonly referred to as multiple star schemas or multiple fact tables.

RELATIONAL MODELS

In a relational model, data is simply stored in relational tables without regard to BI topics and inter-table cardinality. That is, facts, categories, aggregations, and measures can exist separately or together in any combination in any table. This is in stark contrast to a dimensional model where categories and facts live in separate tables with a clearly defined one-to-many cardinality between dimension tables and fact tables.

DIMENSIONAL VERSUS RELATIONAL MODELS

Although dimensional models are a standardized and more common way to represent BI data, relational models can also be used in the SAS Enterprise BI Server. This gives users added flexibility when constructing SAS BI models with SAS Information Maps.

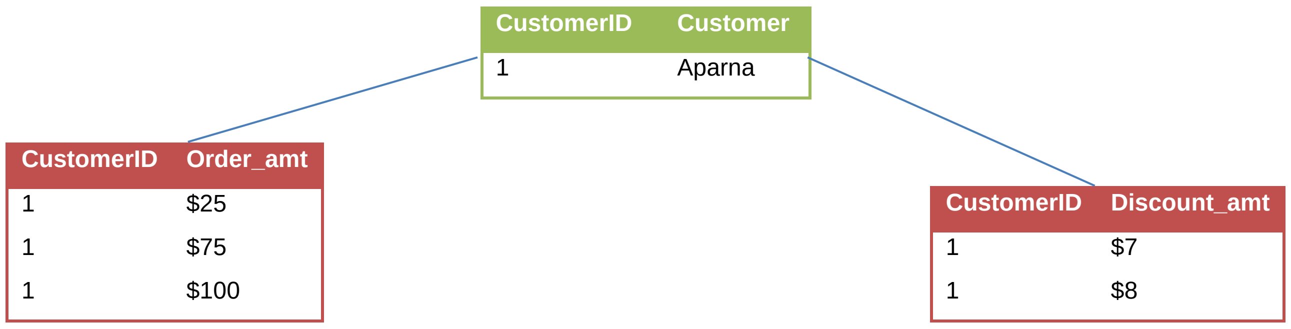
Regardless of whether the underlying data model is dimensional or relational, incorrect results can occur when a business question relies on more than one set of facts that live in different tables. This will be discussed in the next section.

THE MULTIPLE FACT TABLE JOIN TRAP

SQL queries based on a BI model that contains two or more fact tables can generate incorrect results if the fact tables are joined together without taking into account the many-to-many cardinality that exists between these fact tables. This problem is often referred to as the "join trap".

The following example illustrates this problem.

Assume that a business analyst wants to know the total order amount and total discount amount for each customer in a company. Orders and discounts are separate business processes that are represented in two different fact tables. The measures for this query are SUM(order_amt) and SUM(discount_amt) because the business analyst wants to know the total of each per customer. In the scenario illustrated in the following figure, we have one customer (Aparna) along with the abbreviated Orders and Discount fact tables:



CustomerID	Customer
1	Aparna

CustomerID	Order_amt
1	$25
1	$75
1	$100

CustomerID	Discount_amt
1	$7
1	$8

TOTAL: $200 TOTAL: $15

Figure 1. Sample Data Model Illustrating Join Trap

When we aggregate correctly, Aparna has a total of $200 from ORDER_AMT and $15 from DISCOUNT_AMT (by summing the three rows from ORDER_AMT and the two rows from DISCOUNT_AMT separately from each table). That is the answer we are looking for.

But if we were to join these three tables together in a single SQL query, we would get SIX rows (Cartesian product of the two tables); each row in ORDER_AMT is joined to each row in DISCOUNT_AMT, resulting in duplicate rows. If we were to then total those values (which are duplicated), the summed values are inflated:

Aparna	$25	$7
Aparna	$25	$8
Aparna	$75	$7
Aparna	$75	$8
Aparna	$100	$7
Aparna	$100	$8
Total:	**$400**	**$45**

Table 3. Example of Inflated Aggregated Results from Duplicate Rows

The problem in the preceding example is that there is a many-to-many cardinality between the customers in both fact tables. Because an individual customer ID can have many transactions in both fact tables, the same customer ID can occur many times in both tables. The resulting duplicate rows (as seen in the example) cause inflated results when those values are aggregated.

HOW THE INTELLIGENT QUERY GENERATOR SOLVES THE PROBLEM

The SAS 9.3 Intelligent Query generator solves the join trap problem without needing to know anything specific about the underlying relational or dimensional data model. This unique feature of the Intelligent Query generator also frees up the user from needing to understand the data model. For example, knowing the inter-table cardinality in an Information Map becomes irrelevant to generating correct queries. Users also do not need to know anything about where facts and measures live in the model. The Intelligent Query generator figures this out automatically.

The Intelligent Query generator automatically examines an Information Map to determine whether there is a potential join trap. It does this by first identifying all measures, categories, and filters in the map and determining what tables they live in.

If a join trap is detected, then the Intelligent Query generator knows not to simply join all the tables, but rather generate a query that eliminates the join trap problem. It accomplishes this by gathering detail data individually for each underlying fact table (from which a measure derives its data) and then appending all the fact-specific result sets into a single result set. Each fact-specific result set joins a given measure’s detail table to the joined results of the category tables and other measure detail tables in a way that eliminates the multiplicative effect inherent in multiple fact table queries. The logic of the fact-specific result sets allows for model-independent query generation.

INVOKING THE INTELLIGENT QUERY GENERATOR IN SAS 9.3

In SAS 9.3, the Intelligent Query generator can be activated in PROC INFOMAPS with a new option on the NEW INFOMAP and UPDATE INFOMAP statements:

```
JOIN_MODEL=BASIC | ADVANCED
```

With option JOIN_MODEL=BASIC, the SAS 9.2 and earlier query generator is used and is appropriate for a star schema query. Option JOIN_MODEL=ADVANCED invokes the Intelligent Query generator. The Intelligent Query generator generates either multiple fact table queries or star schema queries, depending on whether multiple measures in the Information Map live in separate tables (the many-to-many cardinality problem).

The default is JOIN_MODEL=BASIC. In SAS 9.3, you must specify JOIN_MODEL=ADVANCED in order to use the Intelligent Query generator.

This new JOIN_MODEL option can be used when creating new Information Maps in SAS 9.3 or when updating existing Information Maps. If an Information Map already exists (in SAS 9.3 or earlier), you can update the map with the UPDATE INFOMAP statement and apply the JOIN_MODEL=ADVANCED option.

Note that JOIN_MODEL=ADVANCED must be specified for each Information Map that requires the Intelligent Query generator.

EXAMPLES OF USING THE NEW JOIN_MODEL OPTION IN SAS 9.3

```
proc infomaps metauser="your-user-ID"
     metapass="your-password"
     metaserver="your-server-name"
     metaport=8561;

    /* If creating new Map in 9.3: */
     new infomap "Employee Info"
          mappath="/Users/sasdemo/My Folder"
          auto_replace=yes
          join_model=advanced;
```

After setting JOIN_MODEL=ADVANCED for a new map in PROC INFOMAPS, a user can then continue constructing the map with PROC INFOMAPS. The Intelligent Query generator is now invoked when querying from that map in SAS 9.3.

```
/* If updating existing Map in 9.3:
   (NOTE:  Map could have been created in 9.3 or pre-9.3) */
update infomap "Employee Info"
     mappath="/Users/myUserID/My Folder"
     description="Use new Query generator"
     join_model=advanced;
     save;
```

After updating an existing map with JOIN_MODEL=ADVANCED, a user can simply save the map and the Intelligent Query generator is now invoked when querying from that map in SAS 9.3.

A NOVEL WAY TO CONSTRUCT BUSINESS QUESTIONS

This paper focuses on how the SAS 9.3 Intelligent Query generator solves the classic BI join trap problem without requiring any knowledge of the underlying data model. In future releases, the Intelligent Query generator will also have the ability to construct queries based on very granular business questions by allowing users to associate measures, categories, and filters to each other.

The following sections show examples of BI topic associations:

CATEGORIES TO MEASURES

Here are two examples of how categories associate to measures:

ACTION: Exclude category groups from the result set if the associated measure(s) has no detail data (SAS 9.3 behavior).

Example:

If there is no SALES data for the SOUTHEAST region, do not show that SOUTHEAST group in the result set.

ACTION: Include category groups in the result set even if the associated measure(s) has no detail data (future capability).

Example:

If there is no SALES data for the SOUTHEAST region, show the SOUTHEAST group in the result set anyway. In this case, SOUTHEAST shows up with a SALES value of NULL.

MEASURES TO MEASURES

Two examples of how measures associate to measures:

ACTION: Exclude all category groups from the result set unless a set of associated measures has detail data (SAS 9.3 behavior).

Example:

Unless there is detail data for *both* LEASED cars and PURCHASED cars for customer age group, do not return any results.

ACTION: Include category groups in the result set if at least one of a set of measures has detail data (future capability).

Example:

If there is detail data for *either* LEASED cars or PURCHASED cars for customer age group, return results.

MEASURES TO FILTERS

When associating filters to measures, filters can apply to the fact columns that the measures are based on (pre-aggregate filters) as well as to the measures themselves (post-aggregate filters). The following example represents a pre-aggregate filter.

ACTION: Exclude a Measure(s) from the result set unless the Measure(s) has detail data when a filter(s) is applied.

Example 1 (SAS 9.3 behavior):

Unless there is detail data for both LEASED cars exceeding $20,000 and PURCHASED cars for customer age group, do not return any results.

Example 2 (future capability):

If there is detail data for either LEASED cars exceeding $20,000 or PURCHASED cars for customer age group, return results.

Note that in these two examples the LEASED measure is associated with the >20000 filter in different contexts. In Example 1 detail data must exist for both the LEASED and PURCHASED tables, whereas in Example 2 detail data need exist in only one of the two fact tables. In both examples, however, the filter association to LEASED stays the same.

Because a measure-to-filter association can be "nested" within a measure-to-measure association, very granular business questions can be answered.

BI TOPIC ASSOCIATIONS – SAS 9.3 AND LATER

The preceding examples that are annotated as "SAS 9.3 behavior" represent what the Intelligent Query generator automatically does today in SAS 9.3. The "future capability" annotations represent what the Intelligent Query generator will be capable of in the future. These future capabilities will enable users to ask more granular questions of the data model. Specifically, users will be able to fine-tune the generated queries by setting the associations (or relationships) between categories, measures, and filters without needing to understand the underlying data model or SQL. The preceding examples represent just a few of these user actions planned for SAS 9.3 and later.

CONCLUSION

The SAS 9.3 Intelligent Query generator can generate SQL queries based on complex dimensional and relational data models that extend far beyond a star schema. It accomplishes this without needing to know the specifics of a data model (such as inter-table cardinality).

We believe the Intelligent Query generator will provide huge benefits to SAS® Web Report Studio users as well as SAS® Information Map Studio users. Our ultimate goal is to empower users to focus on their business questions rather than the structure of their data. We will continue to add more intelligence to our query generation process in the future.

We welcome your feedback in all areas of the SAS Enterprise BI Server.

CONTACT INFORMATION

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

Fred Levine
SAS Institute Inc.
100 SAS Campus Drive
Cary, NC 27513
Work Phone: (919) 531-6826
E-mail: fred.levine@sas.com