Using Views and Queries with DB2® and MSAccess® Databases to Employ Your Business Rules

Lynn Palmer, State of California, Berkeley, CA

ABSTRACT

With new and easier ways to access data, more users are attempting to get their data directly from the organization database. Data often requires some knowledge of the organization's "rules" for summarizing and presenting information. These rules may not be readily available to or known by all staff. One way to set up and mange these rules is to set up views which govern how the data is reported. You can create DB2® views; SAS® Proc SQL views, queries, and in-line views; and MSAcess® queries that are transparent to your users and allow you to enforce your organization's data rules.

INTRODUCTION

The rules this paper addresses are referred to as "Business Rules". A business rule is defined as a statement intended to define, constrain, influence or guide behavior of an organization. Defining the rules by which you do business allows them to be accountable and measurable. Unprofitable rules can be changed.

Change is become part of running a successful business or organization. However, database and information systems require time to initially construct and more time to manage later changes. If you build the rules into your database, your ability to respond to change will be reduced. It has been suggested that business rules belong in a "virtual tier", between the database tables and the code where they can be easily identified and changed.

SAS® and SQL have a solution for handling the rules in a virtual tier, views. Views can hold the code that is your organization's rules, thereby allowing rules to be recorded, changed easily, and their effect documented.

CATEGORIES OF RULES

In some data systems, the knowledge of how to operate in certain situations, or what should be interpreted from a particular event, or when to generate a special report reside in the head of the senior staff. Business rules are frequently vague declarative statements. To make it easier to uncover these rules, Ellen Gottesdiener defines six categories: term definitions which are a word, a phrase, or a sentence; term connections which are relations between terms; constraints which are undesirable circumstances; action enablers which trigger an action; derivations which are new knowledge using calculations, algorithms; and inferences which are "if, then" clauses. These categories will help to show how rules can be coded into views.

WHAT ARE VIEWS?

Views are virtual datasets (tables) that contain no data, but instead contain code that describes or defines data stored elsewhere. Views are named and used like a table, data set, or file. However, they are only code. There are two advantages to views: they take little space, and they access the data or database at execution, so they reflect the most current data.

From the SAS user's perspective there are two groups of views. The first group consists of those views created and used in SAS code: SAS data views, Proc SQL views, in-line views and Access descriptor views. The second group are the views created inside your database, frequently with SQL or with Proc SQL. These views are code residing inside the database. Your database administrator must grant you permission to create these views.

VIEWS IN SAS CODE

SAS DATA VIEWS

SAS DATA step views tell SAS how to process data using the code that you specify in the DATA step. Only one view can be created in a DATA step and the view name must match one of the DATA step names. These views can also generate datasets when the view is executed. A view is executed when it is used in a PROC or DATA step. The view can be stored in a SAS library as a view type if you create it with a two part name (library.viewname). The syntax is:

```
DATA VIEWNAME / VIEW=VIEWNAME;
```

SQL VIEWS

SQL views are defined in PROC SQL using a CREATE VIEW statement. They are based on SAS datasets, sequential files, databases, and other views. They can contain code to connect to an outside database using the PROC SQL pass-through facility. They can be in the program code or stored in a SAS library. These views are more efficient because at execution they utilize the SQL processor which optimizes the access strategy. The following is an example:

```
PROC SQL;
CREATE VIEW AS LIBNAME.VIEWNAME;
      SELECT TABLE.FIELD
      FROM LIBNAME.TABLE
      WHERE SOME CONDITION IS TRUE;
```

or the following for connecting to DB2:

```
PROC SQL;
CONNECT TO DB2 AS MYCON
(SSID=YOURID)
CREATE VIEW AS LIBNAME.VIEWNAME
      SELECT TABLE.FIELD
      FROM LIBNAME.TABLE
      WHERE SOME CONDITION IS TRUE;
DISCONNECT FROM MYCON;
```

IN-LINE VIEWS

In-line views are defined in a PROC SQL query and used as part of the query. These views are in the code in your program. Using these views can simplify complex code, for example:

```
PROC SQL;
CREATE TABLE AS MYTABLE
SELECT *
  FROM (CREATE VIEW AS MYVIEW
     SELECT TABLE1.FIELD
     FROM LIBNAME.TABLE1
     WHERE SOME CONDITION IS TRUE;)
       ,
     LIBNAME.TABLE2 MYTABLE
WHERE MYVIEW.FIELD = MYTABLE.FIELD;
```

ACCESS DESCRIPTOR VIEWS

SAS/ACCESS views are views which interface with the data of other software products. A descriptor is created for your database table or file using the SAS/ACCESS product for your non-SAS database or files. These descriptors can only access one table at a time. The view is based on this descriptor.

In these views, you can format fields with SAS formats. You can assign different names to the database fields in the view and change the field length. You can limit the fields available to your users and you can include SAS code to change field values or create new fields from the fields in the underlying database table. With this type of view, you can change the data in the underlying database table by updating, modifying, or deleting.

The SAS/ACCESS descriptor for the table can be created interactively or in batch. The following code will create a view and its descriptor for a MSExcel worksheet:

```
PROC ACCESS DBMS=XLS;
 CREATE LIBNAME.TABLE.ACCESS;
  PATH='C:\YOURFILE.XLS';
  GETNAMES=YES;
  LIST=ALL;
 CREATE LIBNAME.TABLE.VIEW;
  SELECT FIELD;
  SUBSET WHERE SOME CONDITION;
  LIST VIEW;
RUN;
```

VIEWS RESIDING IN DATABASES

DB2 VIEWS

Views in DB2 are SQL code. They can be created using PROC SQL, if you are granted permission to create views in your organization's

database. The SQL features available depend on the database not SAS. These views can be very powerful and efficient as they are part of the database and can utilize the database processing tools. Code to produce a DB2 view is as follows:

```
PROC SQL;
CONNECT TO DB2 AS MYCON (SSID =
 YOURID);
  CREATE VIEW AS FULL.NAME.MYVIEW
  SELECT TABLE.FIELD
  FROM FULL.TABLE.NAME TABLE
  WHERE SOME CONDITION;
DISCONNECT FROM MYCON;
```

These views can also be programmed in SQL executed by the database administrator. The view will be stored in the database like the tables in the database. However, unlike the tables, the views will not require much space.

MSACCESS QUERIES

You can set up MSAccess queries to act like views in a MSAccess database. The query will contain SQL code(Microsoft's version). You can access this query the same way that you access a table using the PROC SQL pass through facility(CONNECT TO) and ODBC. These queries can be created visually using MSAccess menus or by programming SQL code in MSAccess. The following code will allow you to use a query in MSAccess with SAS. You will first need to set up the ODBC connection on your PC.

```
PROC SQL;
CONNECT TO ODBC AS MYCON
 (DSN='MYLINK');
CREATE TABLE MYTABLE AS
SELECT *
FROM CONNECTION TO MYCON
 (SELECT QUERYNAME.FIELD
  FROM QUERYNAME
  WHERE SOME CONDITION);
DISCONNECT FROM MYCON;
```

EXAMPLES OF RULES AND VIEWS

With the categories of rules and the different types of views in mind, the argument for views as an excellent tool to build a virtual layer between your database and your users can be made. Some of the following examples of the rule types managed in views are from the referenced articles and some from my experience.

An example of term definition being applied is the use of DB2 views in our research unit. The administrator of our DB2 database named the fields and tables in the database with 16 character names to reflect the source names. Our unit writes SAS code to retrieve data from DB2. We shares our SAS code within the unit. We like standardized names for fields and we needed eight character names for fields and tables. So, we developed SAS names for the fields, grouped in ways that make analysis easier. We, then, created views with eight character names in DB2 that renamed and regrouped the fields. These views don't take up storage space, they allows us to find fields faster, and they run as fast as using the underlying tables.

The database administrator now wants to change the structure of the underlying tables, combining several together. As long as he recreates the views by changing only the table name, so the views appear to be unchanged, he can make the changes without having to disturb our method of working. Thereby, avoiding a battle over how and where fields are grouped.

In our data, one client can have many records. Which record to analyze can become difficult to determine without detailed coding. A view stored in a SAS library with the indicator pointing to the correct record to choose for analysis allows analysts to avoid coding by using the view. Because this view describes the relations between terms on different records, it is an example of a term connection.

The following code shows the application of a constraint. By using this view, new, the dataset, old is checked for missing dates and they are filtered out. At the same time, a dataset of the missing dates is created called, errors.

```
data new errors / view=new;
set old;
if date = . then output errors;
else output new;
run;
```

The dataset errors is created only when the view

is executed and there are missing dates.

The remaining types of rules: action enablers, derivations, and inferences also lend themselves to easily being coded in a view. SAS analysts derive new data from old creating new variables with SAS code, such code can be included in a view. IF THEN statements which are defined as inference rules are again simple to code in SAS and therefore can be contained in a view.

CONCLUSION

The examples above are simple, but the application of views can be as complex as your situation needs. Views can be a powerful tool when accessing data in your database. Views can allow your users to proceed without misusing the data in your database.

REFERENCES

SAS® Language, Reference, Version 6, First Edition, SAS Institute Inc., Cary, NC, USA, 1990.

SAS® Procedures Guide, Version 6, Third Edition, SAS Institute Inc., Cary, NC, USA, 1990.

SAS® Guide to the SQL Procedure, Usage and Reference, Version 6, First Edition, SAS Institute Inc., Cary, NC, USA, 1989.

SAS/ACCESS Interface to IMSDL/I:Reference; DATA; LIBNAME, SAS/ACCESS; The SQL Procedure; On-line SAS Documentation, SAS Institute Inc., Cary, NC, USA, 1998

Ellen Gottesdiener , EBG Consulting, Inc.©, Business Rules Show Power, Promise Application Development Trends, vol. 4, no. 3, March, 1997 at http://www.ebgconsulting.com /powerpromise_article.htm

Knowledge Partners, Inc., Business Rules Today: A KPI Position Paper, http://www. kpiusa.com/featured.htm, 1999

Lather, Kirk Paul, 'Frame Your View of Data with the SQL Procedure', Proceedings of the Third Annual Western Users of SAS® Software Regional Users Group Conference, 1995, Pages 333-336.

Lather, Kirk Paul, 'Querying the Data Warehouse with the SQL Procedure SELECT Statement', Proceedings of the Sixth Annual Western Users of SAS® Software Regional Users Group Conference, 1998, Pages 302-306.

Wilson, Steven A., 'Using SAS® Views for Data Source Visualization', Proceedings of the Fourth Annual Western Users of SAS® Software Regional Users Group Conference, 1996, Pages 389-398.

Kuligowski, Andrew T., 'An Overview of Techniques to Introduce External Data Into the SAS® System', Proceedings of the Sixth Annual Western Users of SAS® Software Regional Users Group Conference, 1998, Pages 491-509.

TRADEMARKS

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ®indicates USA registration.

IBM is a registered trademark or trademark of International Business Machines Corporation in the USA and other countries. ®indicates USA registration.

MSWindows®, MSDOS®, MSVisual Basic®, MSWord® and MSExcel® are products of the Microsoft Corporation. Copyright © 1983-1997 Microsoft Corporation. All rights reserved.

Other brand and product names are registered trademarks or trademarks of their respective companies.

AUTHOR

Lynn Palmer
Email: lpalmer@dhs.ca.gov,
Lpalmer363@aol.com,
Lipalmer@compuserve.com
Phone: (510) 540-3296