Paper 135-25

Warehousing, Metadata, and Object-Based Analysis

Philip M. Pochon, Computer Task Group, Inc., Indianapolis, IN
Thomas H. Burger, Eli Lilly and Company, Indianapolis, IN

Abstract

Warehouse technology amasses teradata as the basis for Decision Support Systems (DSS). Key to knowledge formation is the organization and representation of such data. Identification of data structures enables the selection of decision support models. Use of object-classes with warehouse metadata allows creation of analytic frameworks. We discuss the production application of object-based SAS® analyses driven by warehouse metadata.

Introduction

In the current business climate, competitive advantage requires knowledge-based information delivery. The right information must be readily available for decision makers. Integral to information technology, Data Warehouses organize and maintain a steady stream of information.

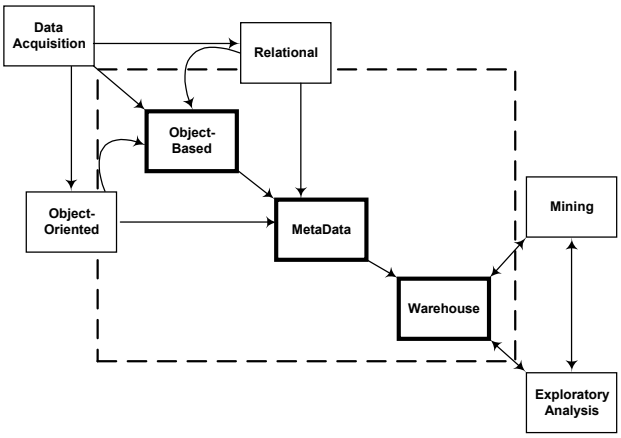
Welbrock (1998) defines Data Warehousing as "… a process of fulfilling Decision Support enterprise needs through the availability of information" using the key terms:

"Decision Support needs" and
"availability of information".

In the classical repository view (Sagar and Raval, 1999), warehouses exist primarily to organize input data for DSS. However, *business decisions* are typically based on condensed analytic results. Interpreting these results requires an understanding of *how* they were computed (application metadata). Welbrock's second term implies these metadata definitions must also be warehoused.

DSS are often built using object-oriented (OO) methodology, where data is encapsulated in objects. Kolosova and Berestizhevsky (1998) describe a table-driven approach to object-based SAS programming. Integrating relational warehouse metadata with object processing yields the hybrid architecture shown in Figure 1.

Figure 1



Our DSS warehouse was built for a multi-business area R&D division. Its vision (Welbrock, 1998) is to ensure that information about pharmaceutical compounds is accessible to researchers.

Warehouse Modeling Overview

The Conceptual Warehouse

The *conceptual warehouse* begins by defining high-level enterprise needs. The definition of the business model is then elaborated to progressively finer detail (business & software requirements, test conditions). This enables creation of a framework upon which the architecture is based.

The creation and maintenance of the *conceptual warehouse* in a requirements management tool allows complete traceability by linking requirements to the system's physical design and test scripts (Pochon and Burger, 1998).

At inception, the *conceptual warehouse* considered more enterprise needs than were addressed in the system's initial release. A cyclical development approach was used wherein a release was scheduled every 4-6 months. The scope (enterprise needs) of each release was defined by a prioritization committee.

The Logical Data Model

The logical model of the (meta)data was developed using Rumbaugh et al.'s (1991) object-oriented methodology. Objects, classes and attributes were defined while events and states were modeled *prior* to physical modeling and making decisions about application software.

Simultaneous modeling of process with structural, operational and application metadata allowed seamless physical modeling of these logical entities. See the section on *Object-Based Processing* for further discussion of modeling and design.

The Logical Processing Model

The primary warehouse is linked to a number of data acquisition systems which "push" collected results to it. User-defined processing metadata then directs the generation of computed results, reports and graphs. Figure 2 shows a high-level processing overview diagram.

Collected and computed results in the warehouse are available for visualization, mining, data marts and mini-marts.

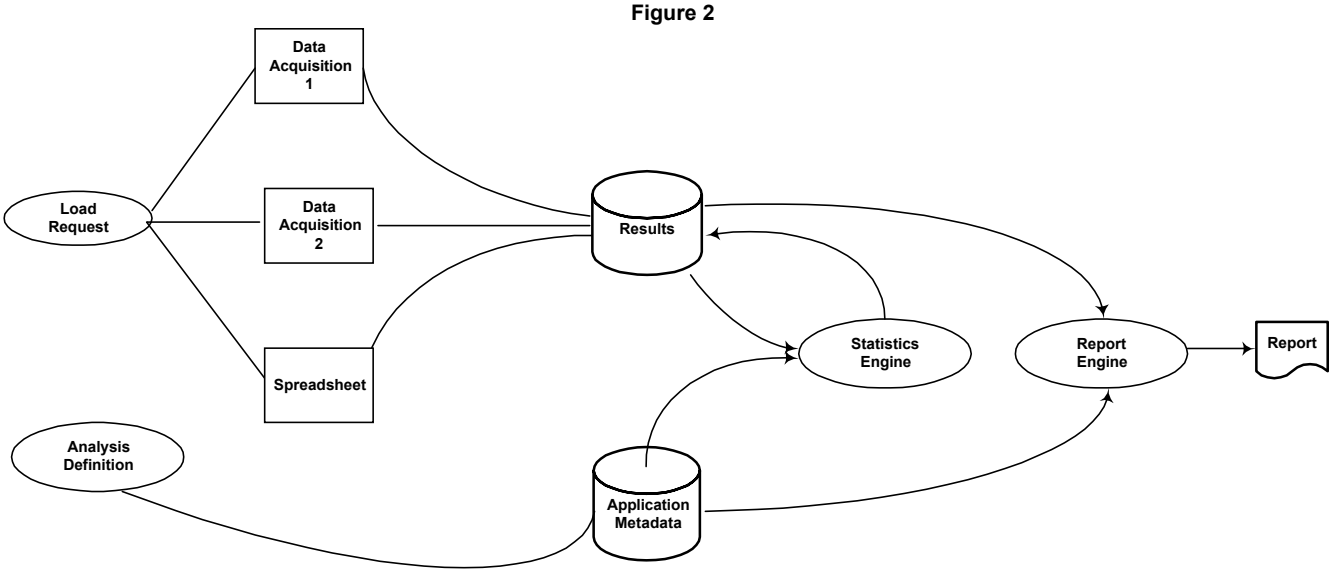
Metadata

The general definition of metadata is that it is "data about data". We find it useful to subdivide this notion into three distinct types: *structural* metadata, *operational* metadata and *application* metadata.

Structural Metadata

Structural metadata is the definition of how the data is physically represented. In a tabular storage system (SAS datasets , relational database), this includes table metadata and column metadata. *Structural* metadata is needed to load and maintain the warehouse and may impact computational and processing control.

Figure 2



Operational Metadata

Operational metadata is information about the result values (collected and computed) stored within the warehouse. *Operational* metadata is divided into a number of objects within our data model, which include common notions, such as:

Analysis Variable	(Body Weight, Blood Glucose)
Units of Measure	(kilograms, mmg/liter)
Subject	(Patient or Animal ID with defining characteristics such as gender and age)
Treatment	(Control Group or Treated)
Time	(Date or normative Day on Study)
Exclusion State	(Include or Exclude from Analysis)

Operational metadata is loaded from the data acquisition system or generated by application systems that write computed results to the warehouse. The values for most *operational* metadata exist in the warehouse; use of foreign keys allows a single value to be linked to many result records.

Application Metadata

Application metadata is information that allows an application to perform its designated task(s). In our example, statistical analysis and reporting are considered two applications, which possess their own metadata definitions and may have execution-timing metadata indicating which calculations must be performed for a report to be generated.

Statistical analysis metadata is application metadata when it is used to control the calculations performed within an analysis request. We contend that *once a set of statistical values have been computed and written to the warehouse, the analysis application metadata becomes operational metadata, since it is information about the result values stored in the warehouse*.

This shift from application to operational requires that metadata be:

- explicitly linked to the computed results
- stored in the warehouse as long as the computed results remain in the warehouse
- archived when the computed results are archived

Application control metadata need not be constructed from scratch every time an analysis is performed. Each phase of the pharmaceutical R&D chain tends to have a pool of standard calculations that are invoked as the statistical analysis section of the protocol is written.

For example, a one-factor ANOVA with linear trend test is common in toxicology studies. A standard analysis template can be created that specifies the calculations to be performed and which defines many of the options available for each calculation, as shown in Figure 3.

Once this standard template is linked to a concrete set of data, the Population, Variables, Covariate Population and Main Effect Levels can be defined.

That these values *must*, rather than *may*, be defined is enforced with structural metadata about the option, which indicates that it is required. Similarly, options that require ordering (Linear Trend Main Effect Levels) have structural metadata that define the order.

Figure 3

Template: One-factor ANOVA with Baseline Covariate and Linear Trend

Algorithm	Option	Option Value
	Population	
	Variable(s)	
	Main Effect	Dosage Level
Descriptive Statistics	Statistic	Mean
		Standard Deviation
		N of Observations
One-Factor ANOVA	Covariate Type	Baseline
	Cov. Population	
	Variance Structure	Homogeneous
	95% Conf. Limits	Yes
Linear Trend Test	Main Effect Levels	
	Test Type	Two-tailed
	Significance Level	0.05

Object-Based Processing

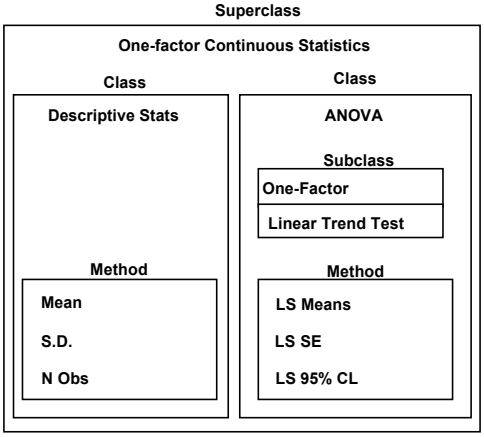
Application metadata is the process control data for statistical programming. We favor the object-based approach of Kolosova and Berestizhevsky (1998). Their method of table-driven statistical programming is well-suited to large scale, generalized systems. Our focus is to introduce the warehousing dimension of the problem and show how it is implemented through application metadata.

Methodology

Kolosova and Berestizhevsky (1998) define a *class* as "… the template or model for a statistical object, which includes data describing the object's characteristics (attributes) and actions that it can perform."

The nesting of Superclass: Class: Subclass considerably simplifies metadata structure and provides reusable program components. Figure 4 shows how the template in Figure 3 may be defined as a nested set of classes.

Figure 4



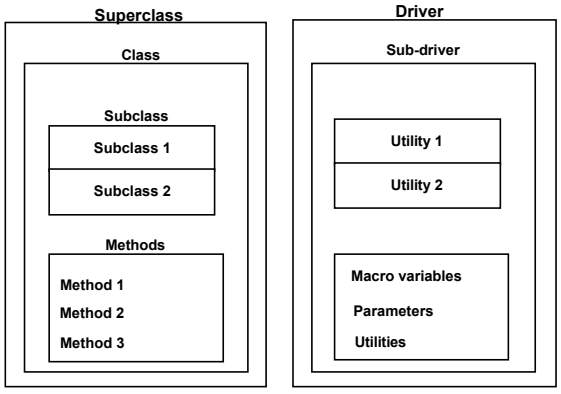
The superclass "One-Factor Continuous Statistics" bounds the available algorithms and defines options (attributes) common to all member classes (Population, Variable and Main Effect). The class "ANOVA" further bounds the available statistical algorithms and options. Finally, the subclass "One-Factor" completes the definitional limits. A subclass "One-Factor Repeated Measures ANOVA" would enable additional algorithms (main effect by time interaction tests) with corresponding options (the definition of the time effect and its levels).

SAS macro language and SAS/STAT allow classes to be implemented in a program hierarchy of Main Driver: Driver: Sub-Driver (Burger and Pochon, 1997).

The actual operations or calculations in a class are performed by its methods (Figure 5). The implementation of a specific method is a SAS macro utility function. The method obtains its required application metadata from:

- the metadata tables in the warehouse and/or
- data sets created by its parent class and/or
- macro variables created by its parent class and/or
- parameters passed by its parent class

Figure 5



These utilities are, in essence, code generators. For example, the implementation of the Linear Trend test is a utility macro that generates ESTIMATE statements within a larger code generator that creates PROC MIXED statements for a one-factor ANOVA. See Pochon (1993) and Stokes (1999) for discussions of dynamically generated SAS code using object-method routines.

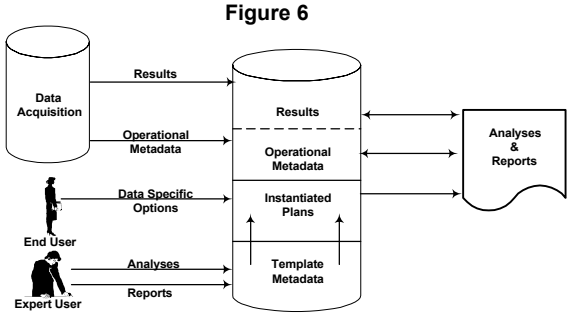
Implementation

RDBMS

The primary warehouse is implemented in an Oracle® relational database. This provides an open architecture allowing many applications (including the SAS analysis engine) to read and write data to and from the warehouse.

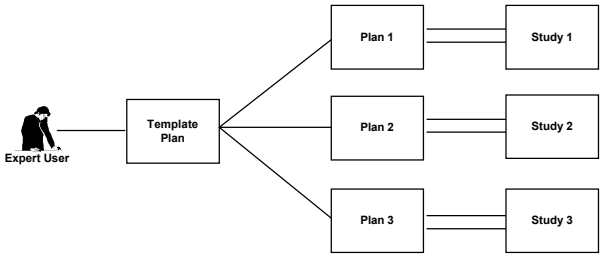
The data's relational structure has several benefits for the SAS analysis engine. The creation of Oracle views in the warehouse allows consistent and efficient data access across applications that would be difficult to achieve in any other fashion. The use of normalized integer foreign keys in views allows the analysis engine to use these keys, rather than denormalized composite values in DATA steps and procedures. The database is logically organized as shown in Figure 6.

Figure 6



Observed results and their accompanying operational metadata are loaded into the results section of the warehouse. Standard analysis and report templates are created by statisticians and expert scientific users then grouped into analysis plans reflecting standard study designs (Figure 7). These templates and plans are then stored as analysis and reporting metadata in the warehouse. The linkage of a template plan to a block of results instantiates a named plan, which may then have its remaining data-specific options filled in. As analysis jobs are executed, computed values are written to the results section of the warehouse. Report jobs then produce formatted reports and graphs.

Figure 7



Application Metadata Quality Control

The creation of application metadata is closely linked to study protocols and their amendments. Once a plan has been linked to a block of data, quality assurance steps are required. One application system is devoted to performing internal checks on the readiness of the metadata. The structural metadata that defines required values and multiplicity determines if an analysis specification is execution ready.

Application metadata is also checked for consistency. For example, if a data transformation is to be performed when a processing BY group's input results are distributed non-normally, the analysis template for that specification must contain a test for normal distribution. Once the application metadata for a plan is execution ready, Quality Assurance (QA) verifies the metadata against the study protocol.

Application Metadata and Process Control

Application metadata controls the processing path through the SAS analysis engine. At the highest level, the combination of data type and calculation type determine the superclass that controls processing. The algorithm(s) included in the analysis template determine the calculations to be performed. The options (attributes) of the specification (and algorithms) provide the details necessary to specify a procedure call or DATA step.

The combination of data type and calculation type permits definition of superclasses as multi-dimensional processing types. Each superclass is defined as compressing one or more dimensions in the data structure.

An algebraic equation (Blood Serum Globulin = Total Protein - Albumin) does *not* compress the subject axis. The output value (Blood Serum Globulin) must be linked to the same subject (animal or patient) as the two input values. It *does* compress the variable axis; two input variables are replaced by one output variable. Similarly, computation of body weight change does not compress the subject axis, but does compress the variable and time axes. Calculation of a control group mean compresses the subject axis completely, but leaves all other axes intact.

The presence (absence) of an algorithm is handled by creating Boolean macro variable flags, which dictate if a method is called. Figure 8 shows a typical sub-driver, which controls the calls for various types of data transformations.

Figure 8

```
%TRANSFORM(_RSLTDSN=WORK.INPTRSLT,
           _RSLTCOL=COLLRSLT);

%IF (&logtran=Y) %THEN %DO;
     %LOGTRAN(_INDSN=&_rsltdsn,
              _TRANSCOL=&_rsltcol)
%END;

%ELSE IF (&sqrttran=Y) %THEN %DO;
    %SQRTTRAN(_INDSN=&_rsltdsn,
              _TRANSCOL=&_rsltcol)
%END;
.
.
.
%MEND TRANSFORM;
```

The options associated with an algorithm allow proper specification of the procedure call. Figure 9 shows a simplified call for PROC MIXED to perform a one-factor ANOVA.

There are three types of control in this example. Embedding the macro variable &COVARIAT in the MODEL statement works since the macro variable is NULL when the covariate is not used. The use of the %IF statement for heterogeneous variance structures controls the creation of the RANDOM statement. Finally, the use of a Boolean algorithm flag controls the call for a macro utility which generates the ESTIMATE statements performing a linear trend test.

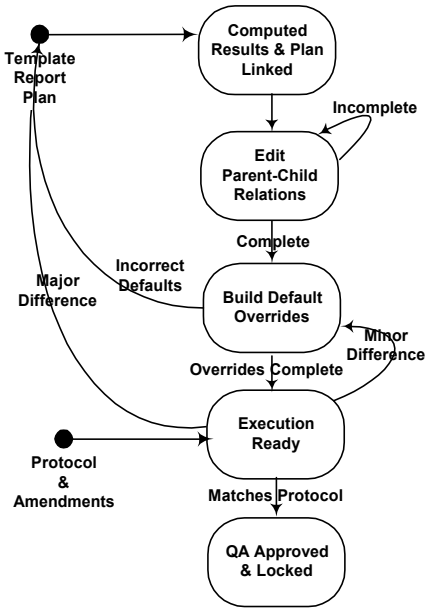
Figure 9

```
%MIX_1WAY(_DATADSN=WORK.INPTRSLT);

 PROC MIXED DATA=&_datadsn;
   CLASS MAINEFFT ANLYUNIT &covariat;
   MODEL RESULT=MAINEFFT &covariat;
   LSMEANS MAINEFFT;
   %IF (&varstcrt=Hetero) %THEN %DO;
       RANDOM ANLYUINT(MAINEFFT) /GROUP=HVRGRP;
   %END;


    %IF (&lintrnd=Y) %THEN %DO;
      %LNTRD_1W(_EFFECT=MAINEFFT,
                _EFFCT_N=&mnefft_n,
                _LABEL=LIN TRND)
    %END;
    MAKE "LSMeans" OUT=LSMEAN NOPRINT;
      .
      .
 RUN;

%MEND MIX_1WAY;
```

Computed Output Metadata

As computed results are written to the warehouse, they remain linked to the application metadata controlling the calculation. The parent-child relations defined between analyses and reports links the report writer metadata to results. Computed results are assigned measure names which are organized into standard report formats. The measures to be printed on the report and the row order for each measure are part of the report metadata. As with analyses, a report plan instantiates a copy of the standard report format, allowing the user to define default overrides. This process is illustrated in Figure 10.

Figure 10



Long Term Access to Application Metadata

Maintenance of application metadata is no different than maintenance of collected and computed results. When a study is archived, application metadata is archived with it. This requires periodic upgrading of archived metadata; as the system is enhanced, the stored metadata must be updated to include new options and features to maintain the original analysis.

Permanent storage of application metadata is critical to long-term use of the warehouse for historical research mini-marts. The ability to query years of accumulated information requires that application metadata be available. To determine the natural distribution of a collected variable, one queries the metadata for analysis specifications that performed the desired distribution test. To determine if a baseline or independent variable covariate is justified, one queries the metdata specifying covariate use.

The ability to produce the application metadata is also important in study audits. Analyses and reports can be reproduced and questions about the nature of an analysis quickly answered from a review of the metadata.

Conclusion

Warehousing computed results requires that the definition of how a value was calculated be stored with the value. Modeling analysis and report definitions as metadata in the warehouse allows the development of data driven object-based applications around the warehouse. These applications are highly flexible, while maintaining the linkage between results and metadata. Quality Assurance and auditing have a clear definition of how each value was computed and long-term research using warehouse data can quickly isolate relevant data in context.

References

Burger, Thomas H. and Philip M. Pochon. 1997. *Structured Design and CASE Tool Use in SAS® Macro Systems*. Proc. of the Eighth Annual MidWest SAS Users Group Conference. Chicago, IL.

Kolosova, Tanya. and Samuel Berestizhevsky. 1998. *Programming Techniques for Object-Based Statistical Analysis with SAS® Software*. Cary, NC: SAS Institute, Inc.

Phipps, Catherine A. 1999 *Integrating SAS® Products using Common Meta-Data.* Proc. of the Twenty-Fourth Annual SAS Users Group International Conference. Nashville, TN.

Pochon, Philip M. 1993. *Building Programs with the SAS® Macro Facility*. Proc. of the 1993 Midwest SAS Users Group. Indianapolis, IN.

Pochon, Philip M. and Thomas H. Burger. 1998. *Validation of SAS® Macro Systems*. Proc. of the 1998 Pharmaceutical SAS Users Group. San Francisco, CA.

Rumbaugh, James, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. 1991. *Object-Oriented Modeling and Design*. Prentice Hall. New York, NY.

Sagar, M. Kumar and Himanshu Raval. 1999. *Data Warehousing in Pharmaceuticals and Healthcare: An Industry Prespective.* Proc. of the Twenty-Fourth Annual SAS Users Group International Conference. Nashville, TN.

Stokes, Jonathan. 1999. *What's All This Metadata Good For, Anyway?* Proc. of the Twenty-Fourth Annual SAS Users Group International Conference. Nashville, TN.

Villiers, Peter. 1998. *The Clinical Data Warehouse.* Proc. of the 1998 Pharmaceutical SAS Users Group. San Francisco, CA.

Welbrock, P.R. 1998. *Strategic Data Warehousing Principles Using SAS Software*. SAS Institute Inc, Cary, NC. 384 pp.

Author Contact

Philip M. Pochon
Computer Task Group, Inc.
Castle Creek IV~Suite 208
5875 Castle Creek Parkway
Indianapolis, IN 46250-4344
Phone 317.578.5100

Thomas H. Burger
Eli Lilly and Company
Lilly Research Laboratories
P.O. Box 708, GL43
Greenfield, IN 46140
Phone 317.277.7266

Trademark Notice