Paper 1792-2014

Big Data/Metadata Governance

Sigurd Hermansen, Westat, Rockville, MD, USA

ABSTRACT

The emerging discipline of **data governance** *encompasses data quality assurance, data access and use policy, security risks and privacy protection, and longitudinal management of an organization's data infrastructure. In the interest of forestalling another bureaucratic solution to data governance issues, this presentation features database programming tools that provide rapid access to big data and make selective access to and restructuring of metadata practical.*

KEYWORDS
Data governance, data quality, data use, security, privacy, SAS®, SQL, networks, servers, databases, Big Data, metadata, SAS/Access.

Introduction
Big Data means some or all of terabytes on storage media, streams of semi-structured data on the Web, blobs of bits in memory, output of sensors, enterprise data infrastructures, federated relational databases, and complex public data releases. Data governance means coordinated management of data quality, access, use, security, privacy protection, updating, and persistence.

The Scandinavian countries, Scotland, England, Wales, Australia, and the Province of Manitoba in Canada have instituted Big Data governance of government administrative records, especially in health, employment, and social services. Many other jurisdictions do not coordinate or control data resources.

Fragmented control of data resources makes data governance especially challenging. A longitudinal study of US residents, for instance, has a hard time finding out whether its subjects are dead or alive. Subject tracing requires investigators to run a gauntlet of applications for data access, data use agreements, security plans, and probablisitic linkage on less than ideal keys. Fees for access to data records add to the time and resource costs of acquiring and managing data. Quality varies across data sources. Taking advantage of big data in scientific medical and social research is becoming increasingly difficult even as existing data and computing resources are expanding exponentially. Anything goes in the wild west of the global internet - except for critical research into human subjects' health and social interactions. A research project involving human subjects has to build a data governance program to fit within a complex of data providers and owners with various interests and standards.

How do research projects approach data governance within the framework of a fragmented data structure? In this context, three guidelines should guide development of a data governance program:

1. **Develop a logical data model first before drilling down to details of implementation.** If users do not have timely access to sufficient data, or they are having data quality, missing value, data exposure, data update, or data permanence issues, a cleverly efficient implementation will not count for much;
2. **Restructure data intially to simplify documentation and whenever necessary to fit user preferences.** If it takes a trained specialist with at least a year of experience to make good use of a research database or public release file, consider a more flexible database design.

3. **Confine exposures of sensitive and proprietary data to users with a "need to know" details as well as authentic credentials.** If data access to subsets or summaries or both suffice for the intended purpose, confining access to subsets or summaries reduces the risk of exposures of sensitive data while not limiting access to sufficient data for research work..

The what and why of each of these guidelines appear in the next three sections. All three taken together do not by any means constitute a complete data governance programme. They do help a data governance programme start out on the right foot.

FIRST GUIDELINE: DEVELOP A LOGICAL DATA MODEL

A good data model of an enterprise server describes how an organization's databases interact. A good Big Data model describes how servers *map* to internal and external data sources and reduce gushing streams of data to useful information. The map-reduce process has a key role.

Map-reduce has become the mantra of data scientists in the era of big data. A mapping function extracts {key,value} pairs from data sources. A reduce function transforms {key,value} pairs into counts, means, or other useful data summaries. A logical model defines the intended targets of the map and reduce functions, and it specifies the range/domain checks, key integrity constraints, business rules, and other basic building blocks of quality data.

A highly simplified example of map-reduce contrives mapping a billion {key,value} pairs and filters out all but those pairs with extreme values:

SAS Solution: Map-Reduce

SAS Proc SQL: WHERE clause

syntax

```
PROC SQL; CREATE TABLE … AS SELECT * FROM … WHERE <condition> ; QUIT;
```

examples

```
DATA r1;  /* Create a Billion Obs. */
    DO key = 1 TO 1e9;
          value = ROUND(1e9 * RANUNI(31313131),1.);
          OUTPUT;
    END;
RUN;
PROC SQL; /* Find Extremes Fast. */
CREATE TABLE extremes AS
SELECT * FROM r1
WHERE value BETWEEN 0 AND 100 OR value BETWEEN (1e9 - 100) and 1e9
;
QUIT;
```

caveats

Use scientific notation carefully. For best results (perhaps any results, set the compress option to 'yes').

A big data reduction program has a number of potential uses and extensions. The trick is to select high value data from a relentless "firehose" stream of big data coming from, say, sensors, the Internet, or credit card transactions. Programs have to execute fast enough to keep data from overwhelming storage media. Filtering out low value data and adding them back in at summary levels using estimates often proves to be an effective strategy. Geologists, for example, have no need for oft repeated measures of relative inactivity in a rock formation. A few small sample seismic reading during long periods of low seismic activity suffice to estimate normal levels of

activity. Putting more resources into collecting the details of seismic episodes makes sense from a scientific point of view. Adaptive design of systems capturing big data intelligently reduces both the scope and expense of data collection.

As SAS/Base users know, the SAS INPUT operator and SQL queries conveniently map and reduce vast expanses of data. In this ideal setting for database programming, the SAS Proc SQL compiler takes full advantage of the SAS System's fast table scanning and I/O capabilities, and it supercharges queries by adding smarter subsetting and indexing methods as opportunities arise for improving query performance. Products such as SAS/BI facilitate extractions of data values of interest from diverse data sources

Computing technologies do not appear to be the bottleneck in struggles to keep up with Big Data. Writing a billion rows of data to a very narrow table (a.k.a. *data set* in SAS circles or *relation* to my fellow hard-core relational database colleagues) actually takes longer in SAS than does a search for a few rows from among the billion using a SAS Proc SQL query with a WHERE clause. The Data step calculations and OUTPUT take less than four minutes, real time, when run on an ordinary Windows SAS PC equipped with a solid state drive (SSD); the search, select, and write in a SQL query took slightly less than half that time. Access to Big Data becomes fast and easy when SQL SELECT queries team up with SAS.

Figure 1 illustrates a data model of the linkage of vital status data sources to a study cohort. The study transmits identifying information (key ID's) to four data sources. The data requestor reveals subject identifying information to the data sources, but does not reveal subjects' attributes that may increase or decrease their mortality risk. The data sources return vital status of matches of the key ID values to vital staus records, but do not return identifying information other than which matches what the study has. The data sources receive and return a linking key that reveals nothing about the subject, and a vital staus. The study receives partially redundant vital staus from the data sources and reconciles these outomes and determines a vital staus for subjects based on information from multiple data sources.

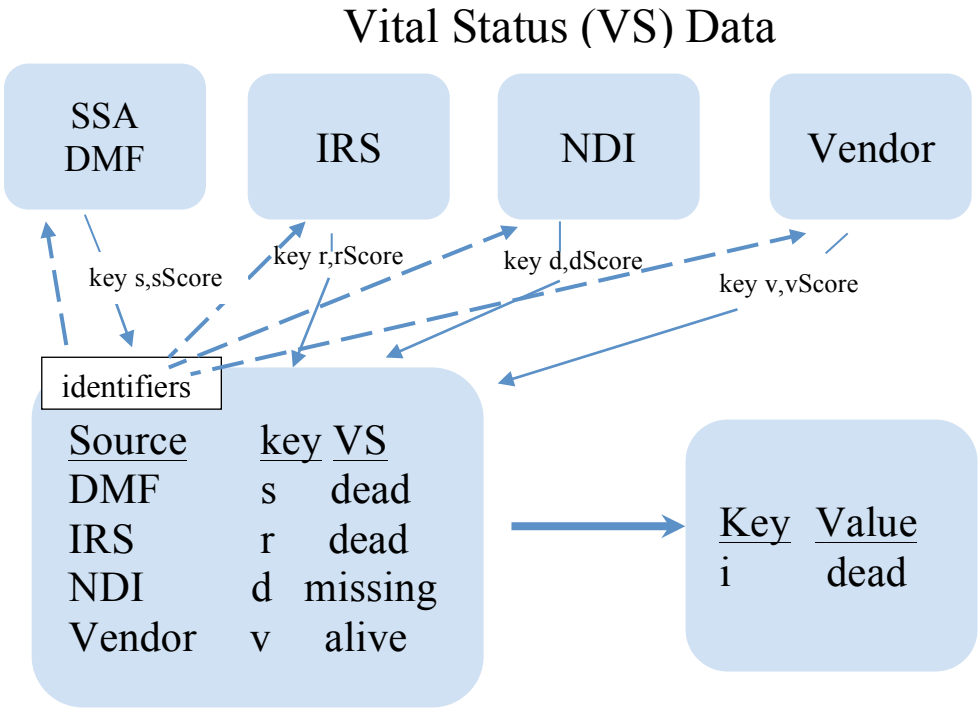

Figure 1: Data model linking vital status data sources to study cohort

Research database developers and those responsible for data governance have a common interest in technologies that move data selection filters and summarization programs closer to places where Big Data are being stored. Filtering out all but targeted data prior to streaming them to potential users not only improves computing efficiency, it also shifts responsibility for selecting data to those who know them best and minimizes exposure of confidential and private information.

Figure 2 presents a highly simplified illustration of a privacy data model for a long-term health research study. A small group within a trusted third party manages the extended database on behalf of study investigators. The study uses subjects' personal identifying information to link to accumulated exposure and outcome data, to external data sources such as the NDI and CMS, and to biologic specimens and results of laboratory testing of those specimens. Blinded ID's (wID and bID) link data sources at the subject level. The subject primary key wID remains the same over time for the same subject. Investigators have access to wID's that link data for the same subjects in analytic data (but not to subject PII). Under the terms of IRB's data use agreements, investigators must re-identify or contact study subjects. Linking data for the same subject using wID instead of PII prevents that from happening. To link subjects to external databases or contact subjects to collect additional data or send news about the study, the database administrators create different bID for each data request. Secure and carefully maintained cross-walk tables link randomly-generated bID back to WID and re-establish links to PII.

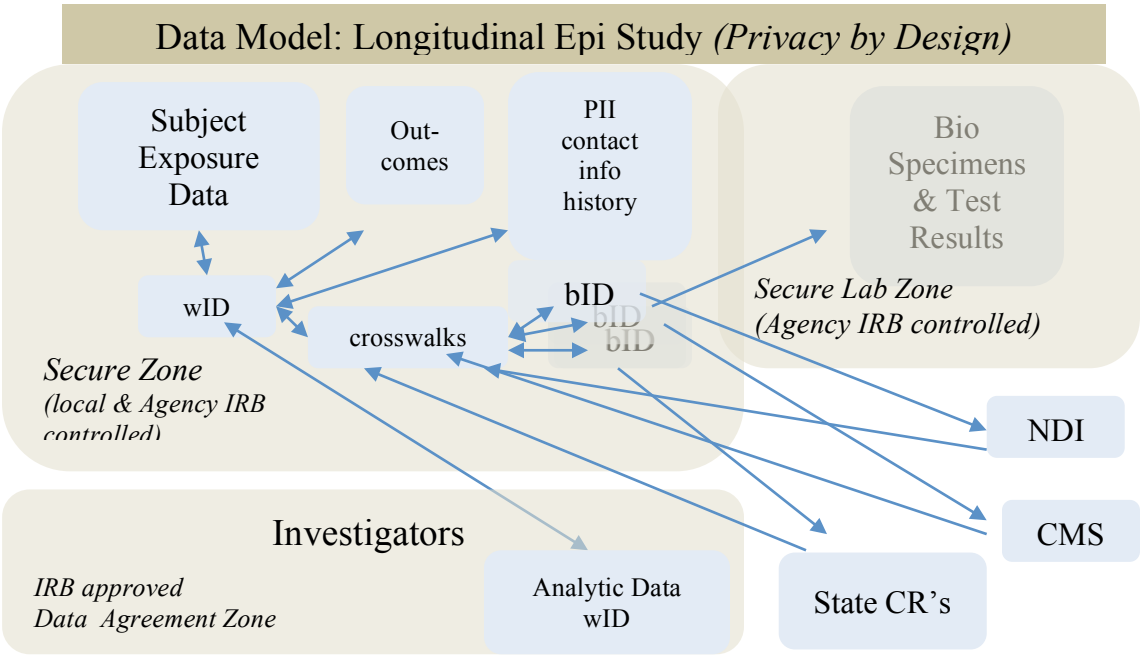

Figure 2: Privacy Data Model for governance of complex epidemiologic research

How Big Data projects govern data structures, storage, and exchanges depends on the nature of data and their purpose. Apache Hadoop and other solutions for Big Data storage offer fault tolerance through storage redundancy and fast access. The SAS System supports smaller scale solutions and, via partners, a variety of full Big Data solutions. A data model has another pragmatic role: it brings the scope of a Big Data infrastructure in perspective and guides data governance decisions.

SECOND GUIDELINE: RESTRUCTURE

Goverance of Big Data tends to bog down in catalogs of large numbers of data variables and their domains. Files and worksheets tend to have metadata mixed in with data, often as entangled as Kudzu ground cover.These and xml documents and objects in memory require special access methods. In contrast, RDBMS ANSI SQL supports offers full access to information about the

structures of data tables and associated key integrity and data quality constraints. Metadata reside in tables that database programmers can access in the same manner as base data tables. Access to metadata facilitates restructuring of relations among data elements.

SAS SQL exposes variable names, labels, types, and other data that describes the structure and content of SAS data sets through views in a SAS special LIBRARY: DICTIONARY. V*iews* because the SAS complier executes the programs they contain to create tables in the same way that it handles Data step and SQL views. DICTIONARY.TABLES contains metadata related to tables, for example, and DICTIONARY.COLUMNS contains metadata related to variables (columns, attributes). In the tradition of relational databases, metadata have all of the properties of other data.

Structure and content determine in part how easy it is to comprehend metadata; it depends on the eyes of the beholder. The World Bank's **DataBank** on the Web delivers economic development data, <http://databank.worldbank.org/data/home.aspx>, for each country in the world. Selecting all data for Tanzania and downloading them as an MS Excel® worksheet gives us 57 columns and 1,265 rows. The columns include country and indicator codes and names and values for years 1960 – 2012: The Indicator Code column has one row of annual time series for each of the indicator codes: e.g.,

Indicator Code
SH.STA.ARIC.ZS
EG.ELC.ACCS.ZS
SE.PRM.TENR
…
NY.ADJ.NNTY.KD.ZG
….
NY.ADJ.SVNX.GN.ZS
….
NY.ADJ.DCO2.GN.ZS
….
SP.ADO.TFRT
….
EN.ATM.METH.AG.ZS
….

The DataBank data structure facilitates access to indicators by country. It arrays data by these variables:

SAS Solution: Restructure Table

SAS PROC SQL/TRANSPOSE Pivot

Syntax example

```
/* MetaData */
libname lib "H:\My Documents";
libname TZA "H:\My
Documents\TanzaniaDataAll.xls" ;

PROC SQL;;
   CREATE VIEW vwTanzaniaData AS
   SELECT * FROM TZA."Sheet1$"n
   GROUP BY Country_Code,Indicator_Code
 ORDER BY Country_Code,Indicator_Code,
          Indicator_Name;
QUIT;

PROC TRANSPOSE DATA=vwTanzaniaData
      OUT=WORK.TanzaniaDataPivot;
BY Country_Code Indicator_Code
   Indicator_Name;
RUN;

PROC SQL;
   CREATE TABLE lib.WBDataTZA AS
   SELECT Country_Code,Indicator_Code,
          Indicator_Name,
          _Label_ AS year,Col1 AS Value
   FROM WORK.TanzaniaDataPivot
   WHERE NOT Col1 IS NULL ;
QUIT;
```

caveats

None

Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963
1964	1965	1966	1967	1968	1969	1970	1971
1972	1973	1974	1975	1976	1977	1978	1979
1980	1981	1982	1983	1984	1985	1986	1987
1988	1989	1990	1991	1992	1993	1994	1995
1996	1997	1998	1999	2000	2001	2002	2003
2004	2005	2006	2007	2008	2009	2010	2011
2012							

Many of the indicators have missing values for all but a few years. Repeating groups of year variables for each indicator inevitably bloats annual series with missing values.

Admittedly, some programmers and analysts appear to be innately inclined toward data structured hortizontally. When arrayed hortizonally, an analyst can select all data values for a country and an indicator by pointing to one row. Others appear to be more inclined to the vertical view. The content fits into a much more compact table when "pivoted" by creating a new row for each observed year, repeating the ID's, and putting each indicator value in a column named Value.

The pivoted worksheet retains the header. No need to keep a row with a missing indicator value for a year: the restructuring largely eliminates a very substantial volume of cells containing missing values. The year labels in the the hortizontal structure become values in the year column of the vertical structure, and the values appear in the Value column.

Country_Code Indicator_Name Indicator_Code **Year** **Value**

The SAS PROC SQL summary functions COUNT() and SUM() work as expected on a table with the vertical structure: e.g.,

```
SELECT Country_Code,Indicator_Code,COUNT(*) as n,SUM(Value) AS Total …
```

The vertical structure also lends itself to updating in a manner that preserves data independence. To add data for 2013 and beyond, simply insert rows. The row structure remains the same and summary queries will continue to generate the expected results.

From a more global viewpoint, we see data governance concerns about storage costs and missing values as no reason to force programmers and anlysts to go against a preference for one database structure over another. SAS PROC SQL and its ally, PROC TRANSPOSE, give us different options. Restructure to vertical. Restructure back to hortizonal. If content fit better in one structure than another, use the better one. If a statistical or graphics procedure requires another structure, transform data into it. Adapt to the *terroir* or change it.

Database documentation that spans hundreds of pages, paper or electronic, clues us right away into the fact that users will have to cut through a big thicket of disconnected metadata. Alternative database architectures may reduce dramatically the need for voluminous documentation and save a lot of time both when creating documentation and when attempting to access the database. As a simple example, one could choose among these alternatives for a DataBank architecture:

Tables	Variables	Number of Variable Names*	Type
1	Country-Indicator-Year	214 X 1,265 X 53 = 14,347,630	Cube
214	Indicator-Year	1,265 X 53 = 67,045	Cubes
1	Year	53	Year Arrays
1	Value	1	Fact Table

* Not including ID's.

Though seemingly absurd as a database architecture due to the overhead required for updating, storing data items in cells of cubes and hypercubes have many advocates who favor cubes for

reporting services. Partitioning this DataBank into 214 tables would add to the difficulty of comparing statistics across different countries. The actual DataBank architecture and the restructured architecture described earlier shift variable names in external pointer arrays to base table data that programmers and analysts can view and access. Better database architecture clears kudzu documentation and SAS PROC SQL navigates quickly through metadata in data tables.

THIRD GUIDELINE: REDUCE DATA EXPOSURES TO "NEED TO KNOW"

Data governance controls the access rights that various groups of individuals have at a given time. This important aspect of data governance protects data from unauthorized exposures, prevents unintended change or destruction of data values, and maintains consistency between documentation and data.

As a rule the default data access right is no access. Each individual with a need to access the database has access rights associated with his or her credentials. He or she enters his or her credentials to gain access to the database system.

Big data governance adds another level of access to the usual logging into a database system. One database system has access rights to a selective view of data that another database system controls. Users of a database system may inherit all or some of these access rights. Further, one database system may inherit all or some of the access rights that another database system has inherited. In these situations the burden of data governance increases dramatically.

Following the "need to know" guideline helps reduce the burden of big data governance by keeping key values private and independent. The simplified data model shown in Figure 2 describes a tested solution. As data stream from one database system to another, the system that owns these data substitutes randomly assigned key values for PII or keys. The random assignments preserve relations among data elements that a user needs to know, but transfer different key values to different database systems. For example, each {key,value} pair representing a person would have the same value of the person component of the key in data transferred to another database system, while another database system would receive a different value of the person component of a key.

Substituting randomly generated keys offers some additional protection against intruders who may gain access to a database tables containing sensitive data. The key substitution process would ideally begin within the database system that controls these sensitive data. Personal identifying information (PII) related to these sensitive data could be encrypted or stored off-line or both and linked back only when needed through cross-walk tables.

Substituting different randomly assigned keys in data transferred to different database systems has an important role in preventing re-identification of persons through the use of public or special use research data. The selection of data received by database system A may not contain enough information to re-identify a person, but a user with access both to A and another selection of data received by a different database system B could combine data from A and B and re-identify a person. For example, database A receives basic demographics for a group of persons and database B receives an ordered sequence of visits to clinics for a group of anonymized patients. While neither the demographics alone nor the sequence of visits to clinics alone would re-identify a person infected with HIV, linking the two sets of data on the same person ID and finding a pattern of visits to clinics treating AIDS patients could, when combined credit reports showing address changes, greatly increase the risk of disclosing a person's HIV positive status.

The database programming behind substitutions of different randomly assigned ID's for transfers of data to different users or database systems turns out to be straightforward. In SAS, a SQL query assigns random numbers to rows of a dataset and orders it by the random number variable. See Random Assignment of ID and cross-walk below. A Data step then assigns a row number to each row of the dataset. This step assigns a unique value to each row. A critical final

SQL query builds a cross-walk table for use in restoring the original (possibly repeating) ID. Limiting access to the cross-walk to a small group of authorized users who need to know the cross-walk minimizes the risk of re-identifying persons and sets up a short chain of accountability.

SAS Solution: Random Assignment of ID and Cross-Walk

SAS Proc SQL:

syntax

```
PROC SQL; CREATE TABLE … AS SELECT * FROM … ORDER BY <random number> ;
QUIT;
```

examples

```
proc sql;
   create table cohortSample as
   select put(ranuni(400713) * 10000000,z10.) as randNum,*
   from cohortSample
   order by randNum
   ;
quit;

data LNExp.cohortSampleID (drop = randNum);
    set cohortSample ;
   ID =put(_N_,z10.) ;
run;

proc sql;
   create table lib.sampleLinking as
   select ID, westatID from LNExp.cohortSampleID
   ;
quit;
```

caveats

Linking dataset needed to link back to person ID.

An effective data governance program controls grants of data access rights at the group and individual level and maintains records of who has access to what and when. It records security breaches, potential data losses, and breaks in continuity. A standard database system will handle records for a big data solution that emcompasses a small number of users and relatively few data sources. Big data solutions involving larger numbers of users or complicated by government regulations, data use agreements, enterprise policies, or business rules will benefit from specialized products such as SAS Data Governance.

CONCLUSION

The basics of data goverance do not change in a Big Data context; the difficulties certainly do. The speed and scale of data transfer and storage technologies is keeping up with demands for Big Data. It is the data suppliers, users, and their representatives who are having a difficult time keeping up with expanding networks of connections and data streaming through virtual machines. It helps to visualize the network of related data sources in the abstract first: that is, create a data model. If data users are spending too much time trying to understand how to access and use large volumes of data, consider restructuring to scale back repeating groups of variables and the task of documenting them. To avoid future concerns about human subject protection, confidentiality, and privacy, limit exposures of sensitive and proprietary data to as few individuals as possible. Substituting blinded keys for PII and links to sensitive data will pay off in no time.

DISCLAIMER

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

REFERENCES

Dobbs *et. al.*, *Big data: The next frontier for innovation, competition, and productivity*, McKinsey Global Institute, 2011.

CSO staff, *Big data policies lacking in Australian and New Zealand organisations: survey*, CSO online, 2013
http://www.cso.com.au/article/532590/big_data_policies_lacking_australian_new_zealand_organisations_survey/

ACKNOWLEDGMENTS

The author would like to thank the Paul's (Dorfman and Kent) for insights into SAS Proc SQL dynamic indexing, Westat's Michael Raithel for his helpful review of a draft of this paper, and the SAS and SAS-L communities for many years of gracious and timely technical support.

CONTACT INFORMATION

Sigurd W. Hermansen
Westat
1600 Research Blvd., WB310
Rockville, MD 20815

301.251.4268
Hermans1@westat.com