Normalizing SAS® Datasets Using User Define Formats

David D. Chapman, US Census Bureau, Washington, DC

ABSTRACT

Normalization is a database concept used to eliminate redundant data, increase computational efficiency and minimize data storage space. This paper discusses the normalization of data and how to do it with user defined formats and SAS datasets. Normalization and its characteristics are reviewed together with the basics of creating user defined formats. Examples are given that show applying user define formats to achieve the results associated with normalizing data.

INTRODUCTION

How you structure your data has implications for efficient storage and processing. This is true whether your data is a SAS dataset or a database. Normalization is a concept that applies to data in general and not just to database systems. The techniques used in database systems can be used with SAS datasets; but, user defined formats unique to SAS can also be used to achieve the same end.

THE PROBLEM

Many – perhaps most – large datasets contain extensive duplication. This duplication is caused by records having common characteristics. Duplication increases storage space, CPU usage, the amount of input-output, and the overall run time. The United States has an estimated 309 million people living in 3,077 counties. This is approximately 100,000 people in each county. If we had a dataset with one record per person, a total of 100,000 records would have the same state, county, geographic division, and region (e.g. Baltimore City, Maryland, South Atlantic Division, North East Region, United States). This example shows duplication related to both geography and a hierarchal classification. There are many other hierarchal classifications used in datasets. An important characteristic of a hierarchical classification is that it is a mutually exclusive set of codes that covers an entire universe or domain-of-study and that can be used to assign any record to a set of progressively larger groups (zip code –>county-->state–>geographic division–>geographic region–>US).

There many other common sources of duplication. Duplication often originates in single value resulting in a constant set of associated values such an a hierarchy or standard set of variable values. For example a business ID always being associate with a standard mail, phone, and electronic address, name, geographic location (latitude and longitude) or business characteristics (legal form of organization, type of business, contact name or other characteristic) .

A SOLUTION - NORMALIZATION

“Normalization” is a systematic way of making a database suitable for general-purpose querying and free of undesirable characteristics. The concept was introduce by E. F. Codd in the 1970's. There are six different levels of normalization; but, usually a database is considered normalized if it adheres to the first three (1NF, 2NF, and 3NF). The process of “Normalization” strives to eliminate redundant data and ensure data dependencies make sense. A database(or set of datasets) is in normal form if all tables in the database are in normal form.

First Normal Form(1NF)

The database is in a regular organization that includes (1) no duplicate columns, (2) every row with a unique set of columns (primary key), and (3) separate tables for each set of related data.

SECOND NORMAL FORM(2NF)

A database is 2NF meets all the requirements of 1NF. The database does not contain subsets of data that apply to multiple rows (they are placed in separate tables). Relationships exist between the original table and new tables through foreign keys.

THIRD NORMAL FORM(3NF)

A database meets all the requirements of 2NF. The database does not contain columns that are not dependent on the primary key.

EXAMPLE OF UN-NORMALIZATION DATA

The data in figure 1 is an example of "Un-Normalization" data.

Figure 1: De-Normalized Data

ID	NAME	BIRTH YEAR	AGE	HOME ADDRESS				EDUCATION	
				STREET	CITY	STATE	ZIP	COLLEGE	LOCATION
1	ADAM	2001	21	123 MAIN	BALTIMORE	MD	12345	NCSU	NC
2	BILL	1976	34	223 MAIN	BALTIMORE	MD	12345	UVA	VA
3	MARY	1976	34	323 MAIN	BALTIMORE	MD	12345	NC SU	NC
4	BOB	1992	18	423 MAIN	BALTIMORE	MD	12345	UVA	VA
5	ALICE	1990	20	523 MAIN	BALTIMORE	MD	12345	NC SU	NC
6	ADAM	1978	32	623 MAIN	BALTIMORE	MD	12345	UVA	uva
7	ADF	1987	23	528 A STREET	Wash;ington	dc	34567	NC SU	NC
8	SALLY	1989	21	528 B STREET	Wash;ington	dc	34567	UVA	VA
9	BARRY	1988	22	528 C STREET	Wash;ington	dc	34567	NC SU	NC
10	DAVID	1985	25	528 D STREET	Wash;ington	DC	34567	UVA	VA
11	HINADI	1993	17	528 E STREET	Wash;ington	DC	34567	NCSU	NC
12	SELVA	1992	22	528 F STREET	Wash;ington	DC	34567	UVA	VA

The data in figure 1 represents un-normalized data. This lack of normalization is illustrated by:

1. There are two sets of repeated groups
   a. The zip code, state and city repeat for first and second group of six records
   b. The college and location repeat when sorted by location
2. There is a data dependency between birth year and age
3. There is a primary key in the ID number.
4. The are two foreign keys
   a. Zip code can be a foreign key for repeating group based on zip, state, and county
   b. College can be the foreign key for the repeating group based on college.
   c. Birth year can be a foreign key for the dependency age

EXAMPLE OF NORMALIZATION DATA

The data in figure 1 can be "Normalizaed" in the following way by creating one primary table and three child tables. There are often a number of different ways for normalizing a set of data. The normalized data consist of the four tables below and their indexes.

PARENT TABLE

The data in figure 2 is an example of the "Normalization" data. The parent table has the same number of rows as the parent table; but, it has four fewer variables. The parent table has four Indexes. The primary index based on ID and three foreign keys or indexes are based on year born, zip, and college.

Figure 2: Un-Normalized Data "NAME"

ID	NAME	BIRTH YEAR	HOME ADDRESS		EDUCATION
			STREET	zip	College
1	ADAM	2001	123 MAIN	12345	NCSU
2	BILL	1976	223 MAIN	12345	UVA
3	MARY	1976	323 MAIN	12345	NC SU
4	BOB	1992	423 MAIN	12345	UVA
5	ALICE	1990	523 MAIN	12345	NC SU
6	JIM	1978	623 MAIN	12345	UVA
7	ADF	1987	528 A STREET	34567	NC SU
8	SALLY	1989	528 B STREET	34567	UVA
9	BARRY	1988	528 C STREET	34567	NC SU
10	DAVID	1985	528 D STREET	34567	UVA
11	HINADI	1993	528 E STREET	34567	NCSU
12	SELVA	1992	528 F STREET	34567	UVA

CHILD TABLE FOR REPEATING GEOGRAPHIC GROUP

The child table removing the redundant information due to the relationship of zip code and state and county is given below. The child table zip table would have an index based on zip variable and have zip as the foreign key in the parent table.

Figure 3: ZIP Child Table

COUNTY	state	zip
BALTIMORE	MD	12345
Wash;ington	DC	34567
ZIP is primary key		

Child Table for Repeating college Group

The child table removing the redundant information due to the relationship of college and location is given below. It has an index based on college variable and have college as the foreign year in the parent table.

Table 4: College Child Table

College	location
NCSU	NC
uva	Va.
College is the primary key	

CHILD TABLE FOR AGE AND BIRTH DATE DEPENDENCY

The age and birth year data dependency can be handled several ways. Two alternatives are to use either a child table with a foreign key or a function. The dependency exists because age is the difference between the current date and

the current year. In 2010 Adam will be age 21 (2010 - 1989 = 21). The alternative to using a function is to use the AGE child table given below. The table has an index based on birth year that would have birth year as the foreign key in the parent table.

Figure 5: BIRTH YEAR CHILD TABLE

AGE	BIRTH YEAR	AGE	BIRTH YEAR	AGE	BIRTH YEAR	AGE	BIRTH YEAR
18	1992	18	1982	28	1972	38	1962
19	1991	19	1981	29	1971	39	1961
20	1990	30	1980	30	1970	40	1960
21	1989	21	1979	31	1969	41	1959
22	1988	22	1978	32	1968	42	1958
23	1987	23	1977	33	1967	43	1957
24	1986	24	1976	34	1966	44	1956
25	1985	25	1975	35	1965	45	1955
26	1984	26	1974	36	1964	46	1954
27	1983	27	1973	37	1963	47	1953

USING NORMALIZED DATA WITH THE DATA IN THE DATA STEP

Normalize data can be used in many ways. In databases and often in SAS, SQL is used. In SAS, the alternative to SQL, is the Data Step. In this paper, the dataset will be used to “de-normalize” data. The “Merge” feature of the data step can be use to reconstruct the data from the normalized information. The “normalized” data consists of the following data datasets and indexes.

Figure 6: Relationship of Normalize Tables

Table	Index	
	Name	Type
NAME (Parent)	ID	PRIMARY KEY
	BIRTH YEAR	FOREIGN KEY
	ZIP	FOREIGN KEY
	COLLEGE	FOREIGN KEY
ZIP (Child)	ZIP	PRIMARY KEY
COLLEGE (Child)	COLLEGE	PRIMARY KEY
BIRTH YEAR (Child)	BIRTH YEAR	FOREIGN KEY

The SAS code to “de-normalize” the data is given below.

```
PROC DATASETS LIBRARY =WORK ;
                select PARENT ;
                index create college;
                quit;
```

```
DATA NAME_ZIP;
MERGE
        PRIMARY
        ZIP;
BY ZIP ;
RUN;

PROC DATASETS LIBRARY=WORK;
MODIFY COLLEGE;
        INDEX COLLEGE;
QUIT;

DATA NAME_ZIP_COLLEGE;
MERGE
        NAME_ZIP
        COLLEGE;
BY COLLEGE ;
RUN;

PROC DATASETS LIBRARY=WORK;
MODIFY COLLEGE;
INDEX COLLEGE;
QUIT;

PROC DATASETS LIBRARY=WORK;
MODIFY BIRTH_YEAR ;
INDEX BIRTH_YEAR  ;
QUIT;

DATA NAME_DE_NORMALIZED;
MERGE
        NAME_ZIP_COLLEGE
        BIRTH_YEAR;
        BY BIRTH_YEAR  ;
RUN;

PROC DATASETS LIBRARY=WORK;
MODIFY NAME_DE_NORMALIZED;
INDEX ID ;
QUIT;
```

The final SAS dataset, ”NAME_DE_NORMALIZED”, contains all the information in the table in Figure 1.

Creating User Defined Formats Based on the Foreign / Child Tables

The information in the child tables can also be created as used define formats. In our example there are three child tables: BIRTH YEAR, COLLEGE, and ZIP. These can be represented and created as three user defined formats. These is no need to create an index because the functionality of an index is included with the user defined format.

CREATING THE USER DEFINED FORMATS

User defined format set up a one-to-one relationship between the original value and reference value. In the format $COLLEGE below the original value is college and the reference value is location. The format $COLLEGE contains exactly the same information as the child table COLLEGE table above.

```
PROC FORMAT ;
        VALUE $COLLEGE ;
        ‘NCSU’=’NC’
        ‘UVA ‘=’VA’;
```

```
RUN;

PROC FORMAT ;
        VALUE BYEAR;
        1981=17
        1982=18
RUN;
```

Formats can reference single values such as with the format COLLEGE or multiple values such as the format $ZIP below. When formats reference multiple values, the need to be broken apart before they can be used in an application.

```
PROC FORMAT ;
        VALUE $ZIP;
        '12345'='MDBALTIMORE   '
        '67890'='DCWASHINGTON  ';
RUN;
```

The three formats above allow us to retrieve data associated with the reference values.

Using the USER DEFINE FORMATS

Creating the user defined formats is equivalent to creating the child tables associated with a normalized set of data. These formats are utilized with a SAS PUT function. The PUT function uses the user defined format and a starting value to retrieve the stored data. The PUT function does both the Look-Up and the Merge at the same time. The code for our example is given below.

```
LOCATION  = PUT (COLLEGE,  $COLLEGE.) ;
AGE       = PUT  (BIRTH_YEAR, AGE.);
```

For formats that contain two or more pieceS of information such as ZIP. The information is retrieved and then parsed. For the ZIP information.

```
STATE_COUNTY  =   PUT( ZIP, $ZIP.)
STATE         =   SUBSTR ( STATE_COUNTY, 1, 2);
COUNTY        =   SUBSTR (STATE_COUNTY, 3, 12)
```

The complete code to create the "de-normalized" SAS dataset is given below.

```
DATA NAME_ZIP_COLLEGE ;
    SET NAME;
    LOCATION        = PUT(COLLEDGE, $COLLEGE.) ;
    STATE_COUNTY    = PUT(ZIP, $ZIP.)
    STATE           = SUBSTR ( STATE_COUNTY, 1, 2);
    COUNTY          = SUBSTR( STATE_COUNTY, 3, 12)
RUN;
```

This proves the old adage that there is more than one way to skin a cat.

CONCLUSIONS

Normalization is a method to increase efficiency by reducing storage space and cpu resources. It is usually thought of as a technique used primarily in databases. User define formats can be used to apply the same principals with SAS datasets.

REFERENCES

Carpenter, Arthur L. (2004) "Building and Using User Defined Formats", SUGI 29 Proceedings ,Montreal, Canada, 2004.

Proc Format, Base SAS 9.2 Procedures, SAS Press, Cary, North Carolaina

Kent, William (1983) "A Simple Guide to Five Normal Forms in Relational Database Theory", Communications of the ACM 26(2), Febrary, 1983, 120-125.

"Database normalization", Wikipedia, the free encycopedia 8 August, 2010 (http://en.wikipedia.org/wiki/datgabase_normalization)

Chapple, Mike. "Database Normalization Basics", About.com Guide August 22, 2010. (http://databases.about.comn/od/specificproducts/a/normalization.htm)

DISCLAIMER

This paper reports the results of research and analysis undertaken by Census Bureau staff. It has undergone a more limited review by the Census Bureau than its publications. This report is released to inform interested parties and to encourage discussion.

ACKNOWLEDGMENTS

CONTACT INFORMATION

Please let me know your comments. You can contact me at:

Author: David D. Chapman
COMPANY: Quality Assurance Staff
Economic Planning and Coordination Division
US Census Bureau
Washington, DC 20233
WORK PHONE 301-763-6535
EMAIL david.d.chapman@census.gov

****