Flugs zur Datenqualität (ohne DataFlux)

Christian Kothenschulte
LBS Westdeutsche Landesbausparkasse
Himmelreichallee 40
48149 Münster
christian.kothenschulte@lbswest.de

Zusammenfassung

Die DSGVO ist einer von vielen Bausteinen aus der Regulatorik, die (nicht nur) an die Finanzbranche höhere Anforderungen - auch - an das Thema Datenqualität stellt. Wenn ein Data Warehouse (DWH) mit SAS betrieben wird, muss dann zwangsläufig DataFlux eingesetzt werden?
Unabhängig von der Beantwortung dieser Frage können im Sprachumfang von SAS Base bereits geeignete Möglichkeiten für Datenqualitätsmanagement gefunden werden. Die drei wichtigsten Anforderungen sind:

- Der Ablauf muss in die vorhandene DWH-Architektur eingebunden werden
- Regeln müssen ohne Softwareanpassung ein- und ausschaltbar sein
- Regeln müssen für alle Beteiligten transparent und nachvollziehbar sein

In diesem Vortrag wird ein prototypisches Framework vorgestellt, das diese Anforderungen erfüllt.

Schlüsselwörter: Datenqualität

1 Motivation / Einleitung

Vor allem die Regulatorik stellt immer mehr Anforderungen an Datenqualität (DQ) und Datenqualitätsmanagement (DQM). Beispielsweise seien hier die Mindestanforderungen an das Risikomanagement (MaRisk, Modul AT 4.3.4) und die Datenschutz-Grundverordnung (DSGVO, Artikel 47) genannt.
Weitere Gründe sich mit dem Thema DQ zu beschäftigen, sind Veränderungen in der Anwendungslandschaft. Wenn zum Beispiel das operative System, ein Provisionssystem oder gar Teile des Data Warehouse ausgelagert werden, muss man sich zwangsläufig Gedanken zu DQM machen. Der erhöhte DQ-Bedarf resultiert also immer mehr aus äußeren Faktoren und externen Quellen ohne Einflussmöglichkeit. Gründe sich bisher nicht mit dem Thema beschäftigt zu haben, können begrenzte Kapazitäten und begrenztes Budget sein. Oder es gibt außerhalb des IT-Bereichs keinen Treiber. Oder man hatte bisher einfach keinen Bedarf bzw. diesen nicht erkannt!

Aber woher will man das wissen?

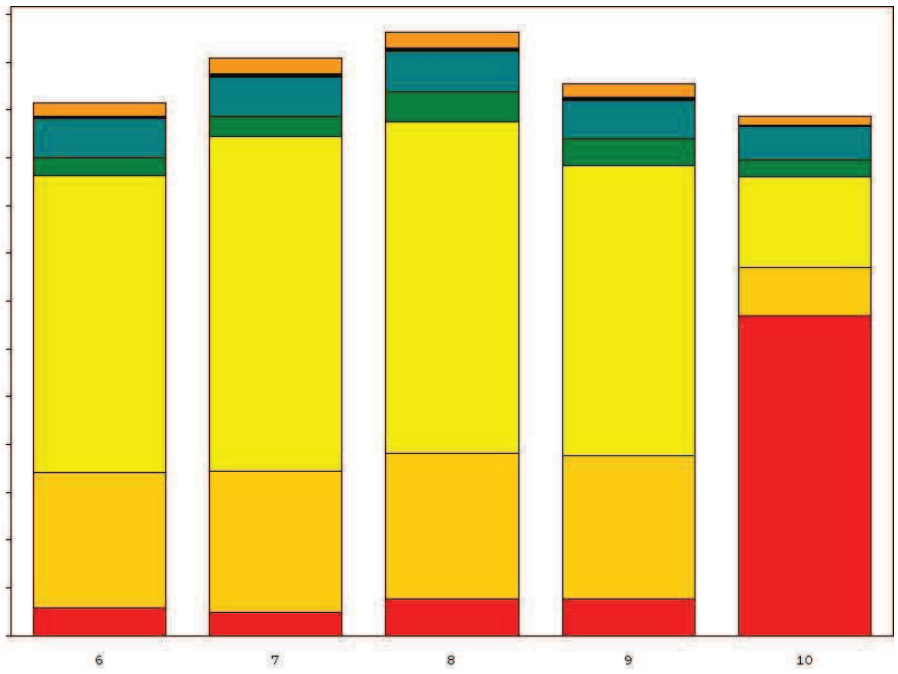

Abbildung 1: Beispiel für grafisch unterstützte DQ-Analysen

Die Grafik zeigt an einem konkreten (abstrahierten) Praxisbeispiel für die Monate sechs bis zehn eine Gesamtsumme und die Aufteilung innerhalb eines Monats nach verschiedenen Kennziffern. Auch wenn die Gesamtsumme jeden Monat einen ähnlichen Wert hat, sieht man dem Balkendiagramm gut an, dass die untere Kennziffer sich im Monat 10 stark zu den vorherigen Monaten verändert hat. Ist das plausibel? Dies gilt es zu bewerten. Aber zunächst muss dieser Sachverhalt aufgedeckt werden.

2014 wurde auf der KSFE der Vortrag „Datenqualität – Auf dem Weg in eine genauere Zukunft“ gehalten [1]. Darin wurden bereits gute Möglichkeiten aufgezeigt, wie man Datenqualität prüfen kann. Neben neu aufgezeigten Varianten findet man auch Dinge, die man bereits, bewusst oder auch unbewusst, für die Datenqualität macht.

2 Vorhandene DQ-Maßnahmen

Bei der Prüfung vorhandener Programme stellt man fest, dass sie bereits einiges zur Sicherstellung der Datenqualität beitragen. Ein paar Prüfungen werden im Folgenden kurz vorgestellt.
Zunächst kann das SAS-Log (automatisiert) ausgewertet werden, da ein paar direkt von SAS generierte Meldungen auf Datenfehler hindeuten:

- Formatfehler beim Einlesen, zum Beispiel alphanumerische Werte in eine numerische Variable oder ungültige Datumsformate

```
NOTE: Invalid data for NUMMER in line 27 1-5.
RULE:      ----+----1----+----2
27         ABCDE
NUMMER=.  _ERROR_=1  _N_=2
```

- Typkonvertierungen, zum Beispiel alphanumerische Werte in eine numerische Variable

```
NOTE: Character values have been converted to numeric values at the
places given by: (Line):(Column).
      25:5
NOTE: Invalid numeric data, 'ABCDE' , at line 25 column 5.
a=. _ERROR_=1 _N_=1
```

Häufig anzutreffen sind einzelne Abfragen, die genau an einer Stelle eine konkrete Konstellation prüfen.

```
  if VERMITTLER eq . then do;
    put 'WARNING: Kein Vermittler hinterlegt';
    abort abend;
  end;
WARNING: Kein Vermittler hinterlegt
ERROR: Execution terminated by an ABORT statement at line 26 column
5, it specified the ABEND option.
VERMITTLER=. _ERROR_=1 _N_=1
```

Beim MERGE kann bequem ausgegeben werden, wenn ein erwarteter Eintrag in einer Steuertabelle fehlt.

```
data FAKTEN_PLUS_MERKMAL;
  merge FAKTEN (in=A)
      STEUERTABELLE (in=B);
  by TARIF;
  if A and not B then do;
    put "WARNING: Kein Eintrag in Steuertabelle! " @;
    put TARIF=;
  end;

  if A then output;
run;
WARNING: Kein Eintrag in Steuertabelle! TARIF=2
```

Bei der Massenverarbeitung von Tabellen wird sichergestellt, ob alle Tabellen verarbeitet wurden.

```
proc contents data=SASHELP._ALL_
              out=WORK.CONTENTS(keep=MEMNAME)
              memtype=DATA noprint;
RUN;
data _null_;
  if ANZAHL ne 0 then do;
    putlog "Es liegen noch Tabellen zur Variante vor. Die Verarbei-
tung wird abgebrochen";
  end;
  set WORK.CONTENTS nobs=ANZAHL;
  stop;  run;
Es liegen noch Tabellen zur Variante vor. Die Verarbeitung wird ab-
gebrochen
```

Jede Maßnahme für sich ist gut und richtig. Trotzdem sollte man sie abstrahieren und in Form von Regeln zusammenfassen. SAS bietet mit Data Quality eine Lösung an, die separat lizensiert werden muss und die Infrastruktur komplexer werden lässt. Es macht daher durchaus Sinn, sich mit dem Funktionsumfang von SAS Base [2] zu beschäftigen.

3 Regeln

Obige Beispiele lassen sich leicht weiter abstrahieren, zum Beispiel:
„Sind Provisionsdaten gefüllt?“ entspricht „Ist ein Feld gefüllt?“ (Regel 1).
Weitere Beispiele für Regeln können sein:

Tabelle 1: Beispiele für Regeln

Nr.	Regel
1	Ist ein Feld gefüllt?
2	Ist eine Tabelle gefüllt?
3	Ist ein Feld Primärschlüssel bzw. mit eindeutigen Werten belegt?
4	Sind alle Werte eines Feldes in einer Steuertabelle enthalten?
5	Sind alle Werte eines Feldes in einem SAS-Format definiert? (Umsetzung mit SAS folgt)
6	Befinden sich alle Werte eines Feldes in einem Wertebereich (Umsetzung mit SAS folgt)

Wichtig ist es, die für die jeweilige Aufgabenstellung relevanten Regeln zu definieren und diese dann konsequent anzuwenden [3].

4 Architektur

Nachdem die Regeln definiert wurden, gilt es, diese in die vorhandene DWH-Architektur einzubinden. Schon in kleinen DWHs werden die einzelnen Programme parallel ausgeführt.
Um die Ergebnisse der einzelnen Regeln auswerten zu können, empfiehlt es sich die Erkenntnisse in CSV- oder Text-Dateien abzulegen, beispielsweise als DQ_*PGM*_REGEL1.CSV.

Der Aufbau der Datei muss einheitlich sein und sollte in einem Makro gekapselt sein.

```
Aenderungszeitpunkt;Bibliothek;Tabelle;Feld;Regel;User_ID;Programm
14JUN2017:11:31:42.390000;SASHELP;CLASS;AGE;1;SYS_LEV5;DW#CLASS
10JUL2017:13:59:44.494000;SASHELP;CLASS;AGE;1;SYS_LEV5;DW#CLASS
```

Abschließend kann ein SAS-Programm alle DQ*.CSV-Dateien einlesen und die Inhalte aufbereiten. Der SAS Enterprise BI-Server bietet bereits zahlreiche Möglichkeiten für das Reporting.

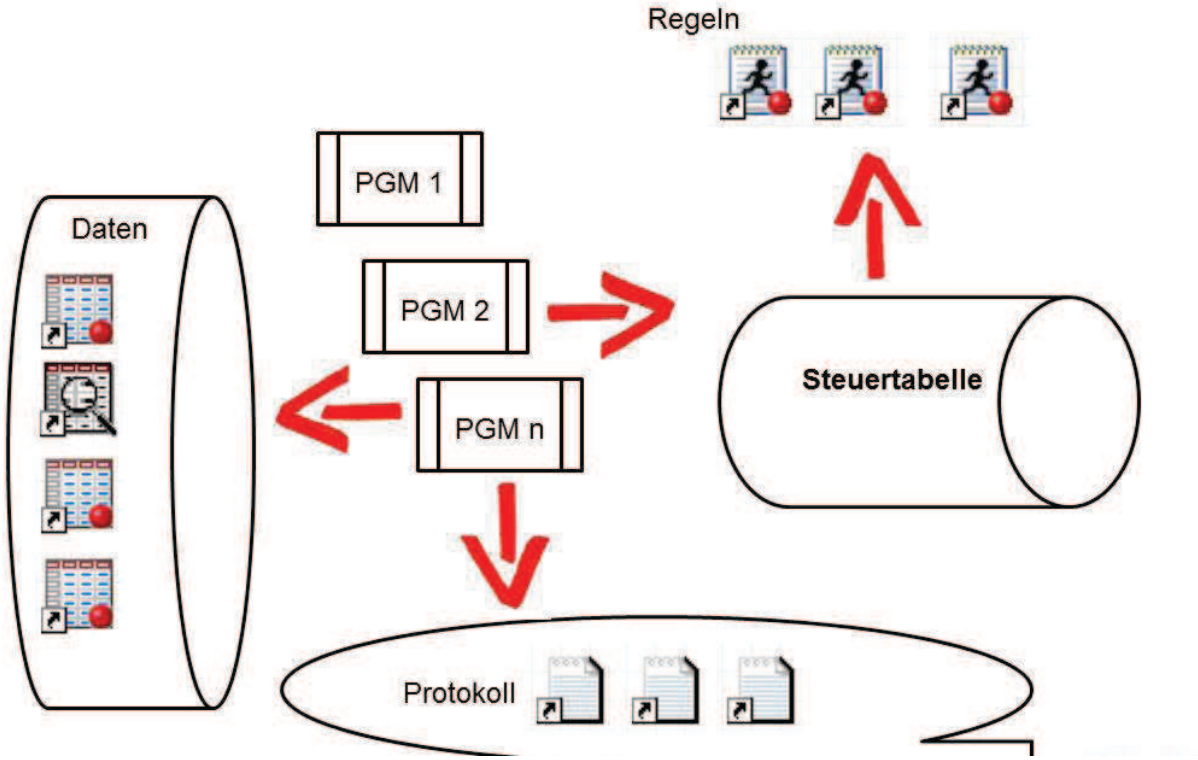

Abbildung 2: Schematische Darstellung Architektur

Eine wichtige Anforderung an das DQM ist die Flexibilität bezüglich der Anwendung von Regeln in Programmen. Um nicht jedes Mal eine Softwareanpassung vornehmen zu müssen, bietet es sich an, die Zuordnung von Regeln zu Programmen in einer Steuertabelle (siehe nachfolgendes Kapitel) zu pflegen.
Die Anwendung der Regeln sollte grundsätzlich im Rahmen des Bewirtschaftungsprozess, also so früh wie möglich im ETL-Prozess, erfolgen, damit fehlerhafte Daten nicht bereits fortgeschrieben werden bzw. sich auf Folgeprozesse auswirken.

5 Steuertabelle

Über eine Steuertabelle können Regeln Programmen zugeordnet werden. Die Pflege kann jederzeit und somit unabhängig von einem bestimmten Einsatztermin erfolgen.

Am einfachsten kann eine SAS-Datei mit DATALINES-Statement genutzt werden. Letztendlich ist die Art der Pflege der Tabelle aber nicht relevant.

```
OPTIONS LRECL=606;
data DWH_DQ (label='Steuertabelle Data Quality');

  infile datalines truncover;

input @01 DQ_PROGRAMM $10.
      @11 DQ_VARIANTE $8.
      @20 DQ_LFD_NR   2.
      @23 DQ_REGEL     5.
      @29 DQ_TABELLE   $32.
```

```
      @62 DQ_BIBLIOTHEK $8.
      @71 DQ_FELD $32.
      @104 DQ_ABBRUCH $1.
      @106 DQ_INDIVIDUELLER_TEIL $500.
;
datalines;
PARTNER          01 00001  TEST123                   WORK     VERTRAGS_NR                 J
VERTRAG          01 00003  TEST123                   WORK     VERTRAGS_NR                 N
VERTRAG          01 00002  TEST123                   WORK                                 N
VERTRAG          01 00004  TEST123                   WORK     TRANCHE_NR                  N
VERTRAG          01 00005  TEST123                   WORK     EV                          N $ERSTVERTRAG
VERTRAG          01 00005  TEST123                   WORK     TARIF_NR                    N TARIF
EREIGNIS         01 00001  TEST123                   WORK     BUCHUNG_DTM                 N
EREIGNIS         01 00008  TEST123                   WORK     SCHLUESSEL                  N
EREIGNIS         01 00007  TEST123                   WORK     PARTNER_NR                  N
KUNDEN           01 00007  TEST123                   WORK     PARTNER_NR                  N
KUNDEN           01 00008  TEST123                   WORK     KEY                         N
KUNDEN           01 00009  TEST123                   WORK     POSTLEITZAHL                N
PROVISION        01 00006  TEST123                   WORK     TARIF_NR                    N 10 75
ADRESSE          01 00001  KND_ADRESSE_UNBEKANNT     BIBLIOT1 VERTRAGS_NR                 N
ADRESSE          01 00001  KND_ADRESSE_UNBEKANNT     BIBLIOT1 HAUS_NR                     N
;;;;
run;
```

Tabelle 2: Aufbau der Steuertabelle

Spalte DQ_	Format	Inhalt
Programm	$10.	(Technischer) Name des Programms, das DQ-Regeln ausführen soll
Variante (*optionales Feld*)	$8.	Falls ein Programm in mehreren Varianten ausgeführt wird, muss die Variante (z.B. MONAT oder TAG) angegeben werden
LFD_NR	2.	Falls bei komplexen bzw. mehrschrittigen Programmen an verschiedenen Stellen (z.B. zu Beginn und zum Ende) DQ-Regeln geprüft werden sollen, können unterschiedliche Nummern vergeben werden.
Regel	5.	Nummer der Regel, die geprüft werden soll
Tabelle	$32.	Tabelle, in der die Regel geprüft wird
Bibliothek	$8.	Bibliothek, in der die Tabelle gespeichert ist
Feld (*optionales Feld*)	$32.	Feld aus der Tabelle, in dem die Regel geprüft werden soll
Abbruch	$1.	Kennzeichen, ob bei Verletzung der Regel ein kontrollierter Abbruch erfolgen soll (J/N)
INDIVIDUELLER_TEIL (*optionales Feld*)	$500.	Individueller Teil der Regel, z.B. Wertebereiche

Die Tabelle ist erweiterbar, zum Beispiel um einen fachlichen Ansprechpartner oder um eine E-Mail-Adresse, die bei Verletzung einer Regel informiert werden soll.

6 Regeln (Umsetzung)

Zunächst empfiehlt es sich, jede Regel einzeln als Makro umzusetzen. Dabei gilt es einen einheitlichen Aufbau und Namenskonventionen zu beachten. Ein einheitlicher Kommentarkopf lässt sich zum Beispiel mit Doxygen (www.doxygen.org) auswerten und für eine Dokumentation aufbereiten.

Grundsätzlich muss geklärt werden, ob wenige generische Regeln oder viele konkrete Regeln gewünscht sind. An dieser Stelle kann die Frage nicht pauschal beantwortet werden.
Da eine weitere Anforderung die Lesbarkeit der Regeln ist, sollte die Umsetzung einzelner Regeln mittels SQL erfolgen. Natürlich ist aber jeglicher SAS-Code möglich. Über einen individuellen Parameter, der je Regel anders interpretiert werden kann, ist bereits eine hohe Flexibilität erreicht.

Die folgende Umsetzung der Regel 1 (Ist ein Feld gefüllt?) zeigt das grundsätzliche Vorgehen. Zunächst erfolgt die eigentliche Prüfung der Regel. Anschließend wird das Ergebnis ausgewertet.

```
%macro DQ_REGEL1(BIBLIOTHEK=, TABELLE=, FELD=, ABBRUCH=N, DQ_IND=);

/* Speichern der Satzanzahl in der Variable DQ_COUNT */
  proc sql noprint;
    select count(*)
    into :DQ_COUNT
    from &BIBLIOTHEK..&TABELLE.
    where &FELD. is missing;
  quit;

/* Auswertung des Ergebnisses */
  %if &DQ_COUNT. ne 0 %then %do;
    /* DQ-Regel verletzt: Ausgabe ins Log */
    /* Weitere Ausgaben/Maßnahmen möglich */
    %put "Regel 1: Verletzung &BIBLIOTHEK..&TABELLE..&FELD.";
  %M_DQ_PROTOKOLL(BIBLIOTHEK=&BIBLIOTHEK., TABELLE=&TABELLE.,
                  FELD=&FELD., REGEL=1);
    %if "&ABBRUCH" eq "J" %then %do;
      %M_DQ_ABBRUCH;
    %end;
  %end;

%mend DQ_REGEL1;
```

Innerhalb der Regel werden zwei weitere übergreifende Makros aufgerufen.
M_DQ_PROTOKOLL: siehe oben
M_DQ_ABBRUCH: Über das Makro kann ein kontrollierter Abbruch erfolgen, falls eine weitere Verarbeitung aufgrund einer Regelverletzung nicht durchgeführt werden darf. In einer weiteren Ausbaustufe können die möglichen Optionen erweitert werden, beispielsweise um die Separierung der regelverletzenden Datensätze.

Bei der oben definierten Regel 5 (Sind alle Werte eines Feldes in einem SAS-Format definiert?) wird zunächst die Prozedur FORMAT genutzt, um alle in einem Format vorhandenen Werte auszulesen. Dabei muss zwischen numerischen und alphanumerischen Format unterschieden werden.

Anschließend kann wieder geprüft und ausgewertet werden.

```
%macro DQ_REGEL5(BIBLIOTHEK=, TABELLE=, FELD=, ABBRUCH=N, DQ_IND=);
  /* Alle Formate exportieren */
  proc format cntlout=WORK.DQ_FORMATE library=APFMTLIB;
  run;
  /* übergebenes Format selektieren */
  /* dabei nach numerisch und alphanumerisch unterscheiden! */
  data WORK.DQ_FORMATE (drop=START);
    set WORK.DQ_FORMATE (keep=FMTNAME START);
    if substr("&DQ_IND.",1,1) ne "$"
       and FMTNAME=strip("&DQ_IND.") then do;
         WERT_NUM=input(START,25.);
         call symput("WERT", "WERT_NUM");
         output;
    end;
    else do;
      if FMTNAME=substr(strip("&DQ_IND."),2) then do
        WERT_STR = START;
        call symput("WERT", "WERT_STR");
        output;
      end;
    end;
  run;

  proc sql noprint;
    select count(*)
    into :DQ_COUNT
    from &BIBLIOTHEK..&TABELLE.
    where &FELD not in (select &WERT. from WORK.DQ_FORMATE)
    ;
    drop table WORK.DQ_FORMATE;
  quit;
/* Auswertung des Ergebnisses */
/* [...]*/
%MEND DQ_REGEL5;
```

Regel 6 (Befinden sich alle Werte eines Feldes in einem Wertebereich) nutzt beispielhaft einen individuellen Parameter, in dem der zu betrachtende Wertebereich übergeben wird.

```
%macro DQ_REGEL6(BIBLIOTHEK=, TABELLE=, FELD=, ABBRUCH=N, DQ_IND=);
  proc sql noprint;
    select count(*)
    into :DQ_COUNT
    from &BIBLIOTHEK..&TABELLE.
    where not (&FELD ge input(scan("&DQ_IND.",1," "),8.)
        and &FELD le input(scan("&DQ_IND.",2," "),8.))
```

```
    ;
  quit;
/* Auswertung des Ergebnisses */
/* [...]*/
%MEND DQ_REGEL6;
```

Für den Aufruf

Bibliothek = SASHELP
Tabelle = CLASS
Feld = AGE
DQ_IND = 10 14

wird folgender SQL generiert:

```
  proc sql noprint;
    select count(*)
    into :DQ_COUNT
    from SASHELP.CLASS
    where not (AGE ge input(scan("10 14",1," "),8.)
        and AGE le input(scan("10 14",2," "),8.))
    ;
  quit;
```

bzw. wie folgt aufgelöst

```
  proc sql noprint;
    select count(*)
    into :DQ_COUNT
    from SASHELP.CLASS
    where not (AGE ge 10
        and AGE 14)
    ;
  quit;
```

Es wird die Anzahl der Datensätze ermittelt, bei denen die Mitglieder der Klasse nicht zwischen 10 und 14 Jahren alt sind.

7 Steuerung

Die Steuerung erfolgt zweistufig. Zunächst werden alle Regeln zum betroffenen Programm (im folgenden Beispiel PARTNER) in der entsprechenden Variante (LEER) und zur laufenden Nummer (1) selektiert und alle relevanten Felder

- Bibliothek
- Tabelle
- Feld
- Regel
- Abbruch
- Individueller Teil

in jeweils eigene und durchnummerierte Makrovariablen geschrieben. Die Anzahl wird ebenfalls in einer Makrovariablen gespeichert.

```
%let DQ_PROGRAMM=PARTNER; /* Aufruf für Programm PARTNER */
%let DQ_VARIANTE=; /* Keine Variante */
%let DQ_LFD_NR=1; /* Erster Aufruf im Programmablauf */

data _null_;
  set STEUERDT.DWH_DQ(WHERE=(DQ_PROGRAMM="&DQ_PROGRAMM."
                        AND DQ_LFD_NR=&DQ_LFD_NR.
                        AND DQ_VARIANTE="&DQ_VARIANTE")) end=EOF;
  COUNT + 1; /* Zähler COUNT hochzählen */
  /* Jeden Steuertabelleneintrag in globale Makrovar. schreiben */
  call symputx(COMPRESS("DQ_BIBLIOTHEK_"!!PUT(COUNT,8.)),
                              strip(DQ_BIBLIOTHEK),'G');
  call symputx(COMPRESS("DQ_TABELLE_"!!PUT(COUNT,8.)),
                              strip(DQ_TABELLE),'G');
  call symputx(COMPRESS("DQ_FELD_"!!PUT(COUNT,8.)),
                              strip(DQ_FELD),'G');
  call symputx(COMPRESS("DQ_REGEL_"!!PUT(COUNT,8.)),
                              strip(put(DQ_REGEL,8.)),'G');
  call symputx(COMPRESS("DQ_ABBRUCH_"!!PUT(COUNT,8.)),
                              strip(DQ_ABBRUCH),'G');
  call symputx(COMPRESS("DQ_IND_"!!PUT(COUNT,8.)),
                              strip(DQ_INDIVIDUELLER_TEIL),'G');

  /* Die Anzahl über den Zähler in einer Makrovariable speichern */
  if EOF then do;
    call symput("DQ_ANZAHL_REGELN", strip(put(COUNT,8.)));
  end;
run;
```

In Schritt zwei wird anhand der gespeicherten Anzahl der Regelaufrufe in einer Schleife jeder einzelne Regelaufruf durchgeführt.

```
/* Per Schleife jeden relevanten Eintrag aus der Steuertabelle abar-
beiten */
%macro DQ_REGELAUFRUF;
  %do i=1 %to &DQ_ANZAHL_REGELN;
        %DQ_REGEL&&DQ_REGEL_&I.(BIBLIOTHEK=&&DQ_BIBLIOTHEK_&I.,
                                TABELLE=&&DQ_TABELLE_&I.,
                                FELD=&&DQ_FELD_&I.,
                                ABBRUCH=&&DQ_ABBRUCH_&I.,
                                DQ_IND=&&DQ_IND_&I.);
  %end;
%mend DQ_REGELAUFRUF;

%DQ_REGELAUFRUF;
```

8 Weitere Schritte

Neben der technischen Seite muss auch die organisatorische Seite von DQM betrachtet werden. Wichtig ist die klare Definition einer Aufbau- und einer Ablauforganisation. [4] Dieser Vortrag beschäftigt sich ausschließlich mit der technischen Seite. Trotzdem sei herausgestellt, dass die organisatorischen Fragestellungen unabhängig von der technischen Lösung beantwortet werden müssen.

9 Fazit

Eine technische Lösung für DQM liegt vor. Sie basiert auf SAS Base-Funktionalität und integriert sich in die vorhandene IT-Landschaft. Die Komplexität ist beherrschbar.

Literatur

[1] D. Schulte: Datenqualität – auf dem Weg in eine genauere Zukunft. In: T. Friede, R. Hilgers, R. Minkenberg (Hrsg.): Proceedings der 18. KSFE Göttingen. Shaker-Verlag, Aachen 2014

[2] SAS(R) 9.4 Programming Documentation

[3] White Paper, Clean vor Big, InfoZoom, Online verfügbar unter http://www.infozoom.com/download/White_Paper-Clean_Data.pdf, zuletzt geprüft am 05.12.2017

[4] D. Apel, W. Brehme, R. Eberlein, C. Merighi: Datenqualität erfolgreich steuern. HANSER, 2010