Daten wie von Hempels unterm Sofa? – Datenmanagement steuern über Access-Datenbank

Christiane Lober
IHF GmbH
Bremserstraße 79 Haus M
67063 Ludwighafen
lober@ihf.de

Zusammenfassung

Der erste Schritt zu einer erfolgreichen statistischen SAS-Programmierung ist die Implementation eines sauberen und nachvollziehbaren Datenmanagements.
Die Verarbeitung von unbearbeiteten Daten stellt den SAS-User vor verschiedene Probleme. Darunter z.B. unübersichtlicher Code, der bei der Verwendung von durchnummerierten Variablennamen entsteht, oder die Gefahr von Verwechslungen von Merkmalen bei der Auswertung der Daten.
Um diese Stolperfallen zu umgehen ist es eine wichtige Aufgabe im Rahmen der Datenverarbeitung den Überblick über die Variablen zu behalten. Die Daten werden aus den verschiedensten, oft externen Datenbanken geliefert. Die Tabellenstruktur, sowie die einzelnen Variablennamen werden von der zugrundeliegenden Datenbank definiert, der SAS-User hat meist keinen Einfluss auf die Struktur, oder die Benennung der einzelnen Merkmale in der zugrundeliegenden Quelle.
Dieser Vortrag zeigt eine Schritt für Schritt Anleitung, wie SAS-Entwickler am Beispiel einer Access-Datenbank die Variablenbenennung und Tabellenstruktur kontrollieren und steuern können.

Schlüsselwörter: Access Datenbank, Datenmanagement

1 Einleitung

Bei der Auswertung von klinischen Studien ist im ersten Schritt ein nachvollziehbares und sauberes Datenmanagement unverzichtbar. Das Datenmanagement in einer klinischen Studie ist integraler Teil der Qualitätssicherung in der klinischen Forschung. Neben den Aufgaben des Datenmanagements bei der Planung der Datenerhebung sind vor allem bei der Programmierung der Datenverarbeitung in SAS diverse Probleme zu vermeiden.

Das Ziel eines guten Datenmanagements als Vorbereitung für eine statistische Auswertung ist es, eine übersichtlich strukturierte Datengrundlage herzustellen, hierzu gehört eine klare Strukturierung der Tabellen und eine sinnvolle Benennung der einzelnen Analysevariablen.

Gelieferte Daten aus unterschiedlichen Quellen können Spaltennamen enthalten, die bei der Programmierung zu Problemen führen können. Hierzu gehört beispielsweise die Verwendung von durchnummerierten Feldern in vielen Datenbanksystemen oder interne

Abkürzungen, wie sie in vielen Laboren üblich sind. Diese Umstände erschweren eine übersichtliche Programmierung mit den gelieferten Variablennamen. Neben schwer verwendbaren Spaltennamen kann auch die Struktur der gelieferten Daten für eine Programmierung zu unübersichtlich sein. Gerade bei der Lieferung von Daten aus verschiedenen Quellen kann es schnell zu einer unübersichtlichen Anzahl von Tabellen mit den unterschiedlichsten Namens- und Strukturkonventionen kommen.

Bei einer Datenverarbeitung mit SAS werden typischerweise in einem ersten Schritt bei der Datenbearbeitung Importtabellen zusammengeführt, Spalten umbenannt und Labels vergeben. Diese Herangehensweise bietet eine schnelle und programmiertechnisch einfache Lösung und kann ohne weiteres in einem datastep durchgeführt werden.

Nachteil einer direkten Programmierung in SAS ist, dass bei einer großen Menge an Spalten die Umbenennung in einem Programm schnell unübersichtlich wird. Es kann nicht gesichert werden, dass Namen doppelt verwendet werden, auch spätere Änderungen von Spaltennamen können in einem SAS-Programm nur schwer nachverfolgt werden. Wird bei der Benennung von SAS-Variablen ein Name in verschiedenen Tabellen doppelt verwendet, gibt SAS beim mergen dieser Tabelle keine Fehlermeldung aus. Das bedeutet, dass eine fehlerhafte Umbenennung schnell zu Fehlern in der statistischen Auswertung führen kann.

Neben doppelter Namensgebung kann es bei der Programmierung „von Hand“ noch zu diversen anderen Programmierfehlern kommen. Diese sind unter Umständen nur schwer zu finden, und machen eine aufwändige Qualitätskontrolle der SAS-Programme nötig. Eine Möglichkeit, die oben genannten Probleme zu vermeiden, ist der kombinierter Einsatz von SAS und relationalen Datenbanken, in diesem Fall MS Access. Dieser Ansatz bietet die Möglichkeit, neben einer sauberen Programmierung unbedingt notwendige Dokumentation direkt mit zu erstellen. Der Zeitaufwand, die Dokumentation nach der Fertigstellung der Programmierung zu erstellen, ist enorm, außerdem spart ein projektübergreifend einheitliches Dokument bei der Übergabe von Projekten zwischen Mitarbeitern viel Zeit. Dieses Dokument enthält Informationen zu Importtabellen und wo die einzelnen Informationen aus diesen Tabellen in den späteren Analysetabellen zu finden sind.

Einen weiteren Vorteil der Bearbeitung unter Verwendung von Access bietet die Verallgemeinerung der Programmierung über mehrere Projekte hinweg. Durch eine einheitliche Struktur bei der Bearbeitung der Daten ergibt sich auf lange Sicht ein kleinerer Pool von projektübergreifenden SAS-Programmen, die einfacher auf Fehler hin überprüfbar sind. Diese Programme werden dann in unterschiedlichen Projekten angewendet. Vor allem aus Sicht der Qualitätssicherung ist dies ein enormer Vorteil, da bereits geprüfte Programme einfach auf mehrere Projekte angewendet werden können und Zeit bei der Prüfung der einzelnen Programme gespart werden kann.

Bei der Programmierung der statistischen Analysen in SAS ist es von großem Vorteil, wenn die Tabellen klar strukturiert sind und die Spaltennamen eine Auswertung der Daten vereinfachen. Neben „sprechenden“ Namen können klare Formate und Labels das Risiko für das Vertauschen von Variablen und eine falsche Interpretation verringern.

Die resultierenden Tabellen mit klarer Struktur und verwendbaren Variablen zusammen mit der erstellten Dokumentation tragen zu einer höheren Qualität im Datenmanagement bei.

Bei der Aufbereitung der Daten in Access können Aufgaben einfach auf mehrere Mitarbeiter verteilt werden, da mehrere User parallel an einer Datenbank arbeiten können. Diesen Vorteil bietet das Programmieren in SAS nicht so ohne weiteres. Bei großen Datenmengen und einer unübersichtlichen Struktur und Anzahl an Spalten können mehrere Mitarbeiter diese Daten gemeinsam schneller bearbeiten und für die weitere Verwendung und Analysen schneller vorbereiten.

2 Systemvoraussetzungen

Windows 10 Enterprise
Microsoft Access 2016
SAS 9.4, products SAS/Access Interface to PC files.

3 Aufbau Access Datenbank

Vor dem Befüllen der Access Datenbank ist es notwendig, einen Überblick über die auszuwertenden Daten zu bekommen. In klinischen Daten werden Pateinten meist durch Zentrums- bzw. Patienten ID oder der Kombination aus beiden identifiziert.

Die Herangehensweise folgt dem Entity-Relationship-Modell: Grundlage der Entity-Relationship-Modelle ist die Typisierung von Objekten, ihrer Beziehungen untereinander und der über sie zu führenden Informationen („Attribute“).

Werden im Projekt beispielsweise kardiologische Prozeduren untersucht, gibt es für einen Patienten die Möglichkeit mehrere Prozeduren zu bekommen. In jeder Prozedur können zum Beispiel mehrere Koronararterien mit diversen Stents behandelt werden. In diesem Beispiel werden die Importtabellen in folgende hierarchische Struktur eingeordnet (s. Abbildung 1).

Jede Importtabelle wird anhand der enthaltenen Identifikationsvariablen einer Hierarchiestufe zugeteilt. Dieser Zuteilung der Importtabellen folgt die Umstrukturierung in die fertigen SAS-datasets. Jede Stufe definieren wir als einen Entity Type, die Informationen aus den einzelnen zugeteilten Tabellen sind Attribute. Die Beziehung zwischen den Tabellen wird durch die Verwendung von gleichen Identifikationsvariablen in verschiedenen Tabellen definiert.

Bei der weiteren Verarbeitung der Importtabellen ist es nötig, dass alle ID-Variablen je Entity Type eindeutig bestimmt sind.

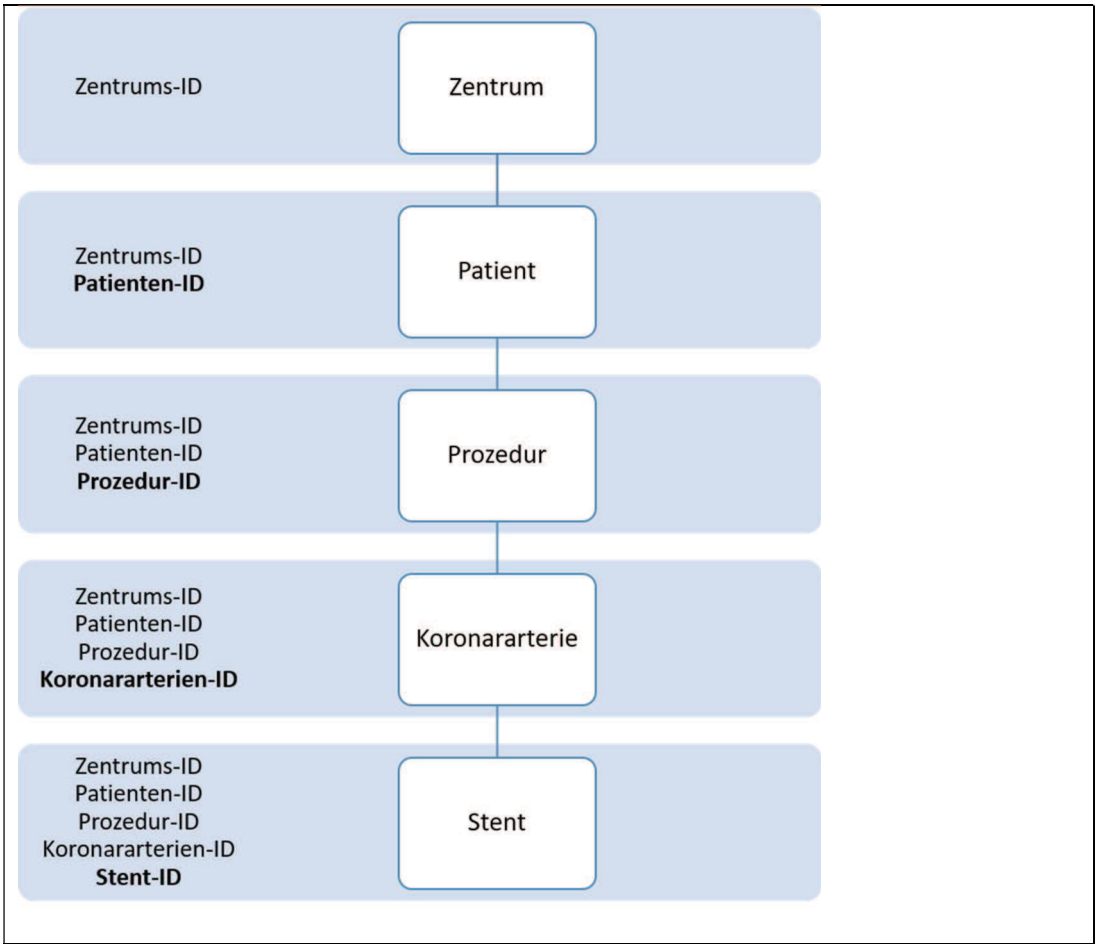


Abbildung 1: Hierarchische Datenstruktur inclusive zugehörigen Identifikationsvariablen pro Entity Type

Die unübersichtliche Tabellenstruktur der Importtabellen wird im ersten Schritt durch die Restrukturierung behoben. Die Daten aus den unterschiedlichen Importtabellen werden hier in die neuen Entity Type Tabellen integriert. Durch die Integration von Importtabellen, die aus verschiedenen Quellen stammen können, ist eine eindeutige Benennung der Spalten notwendig. Jede Spalte sollte außerdem ein Label und falls nötig ein Format bekommen.

Die Information zu jeder Spalte bestehend aus neuem Spaltennamen, Formate, Labels, ursprünglichem Spaltennamen und Information zur Importtabelle werden als Metainformation in einer Accessdatenbank gespeichert und verwaltet. Die Informationen können mit Hilfe von Access Berichten zu einer einheitlichen Dokumentation zu jedem Zeitpunkt ausgelesen werden.

3.1 Aufbau

Die Daten zur Steuerung der SAS-Programmierung und des gesamten Datenmanagements werden in einer Access Datenbank verwaltet. Diese enthält Informationen zu Entity Types und Importtabellen mit den Metainformationen zu den jeweiligen Spalten.

Die Daten werden zunächst in Tabellen organisiert, die über Formulare befüllt und bearbeitet werden können.

Die Access-Datenbank besteht aus mehreren Tabellen, Formularen und Beziehungen zwischen den Tabellen. Die Tabellen teilen sich in 2 Kategorien: Tabellen, in denen projektspezifische Informationen verwaltet werden, sowie Tabellen mit studienübergreifenden Informationen, die als Nachschlagetabellen verwendet werden. Die Nachschlagetabellen Column_status, Column_Types sind in Anhang B angegeben. Die Tabellen mit projektspezifischen Informationen werden im folgenden Abschnitt genauer beschrieben.

3.1.1 Entity Types

In dieser Tabelle werden die notwendigen Entity Types angelegt, die durch die Struktur der Importtabellen definiert werden.

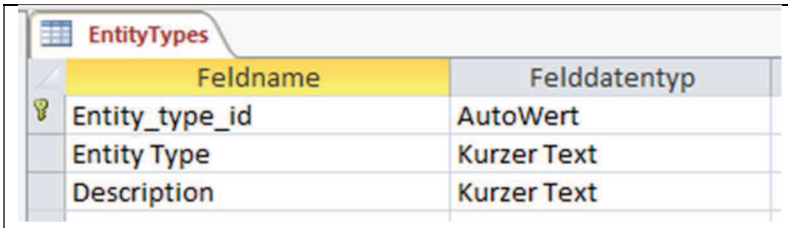


Abbildung 2: Tabelle EntityTypes

Entity_type_id: Von Access definierter Primärschlüssel
Entity Type: Name der einzelnen Entity Types
Description: Kurze Beschreibung der Entity Types

3.1.2 Tables

In der Tabelle Tables werden alle Informationen zu den Importtabellen der Studie gesammelt.

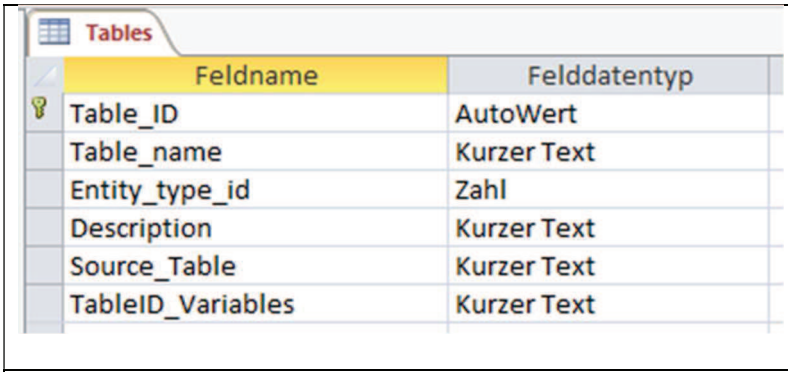


Abbildung 3: Tabelle Tables

Table_ID: Von Access Definierter Primärschlüssel
Table_name: Tabellenname der Importtabelle
Entity_type_id: Information aus Tabelle EntityTypes
Description: Bescheibung des Inhalts der Importtabelle
Source_Table: Name der Tabelle in SAS
TableID_Variables: Identifikationsvariablen, die in Importtabellen enthalten sind
Die Angabe der TableID_Variables ist wichtig, da hiermit die hierarchische Entity Type Struktur der Importtabellen definiert wird.

3.1.3 Columns

Diese Tabelle enthält die Metainformation zu den Analysevariablen aus den einzelnen Importtabellen.

Columns

Feldname	Felddatentyp
column_id	AutoWert
column_name	Kurzer Text
entity_type_id	Zahl
Source_column	Kurzer Text
Label	Kurzer Text
Format	Kurzer Text
Column_status	Kurzer Text
column_type	Kurzer Text

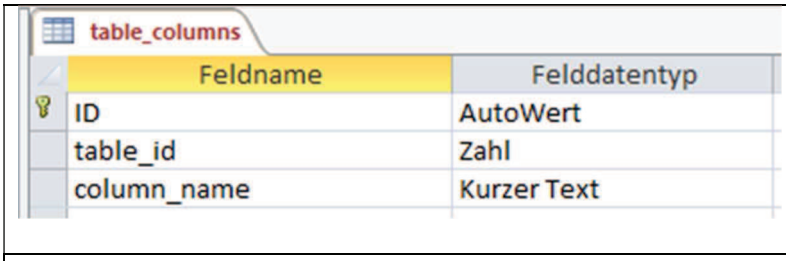
Abbildung 4: Tabelle Columns

In Access können Eigenschaften der Tabellenfelder definiert werden. Diese Einschränkungen können verwendet werden, um Fehler in SAS zu vermeiden. In der Tabelle columns wird der Eintrag column_name wie folgt beschränkt: Die Länge des Spaltennamens darf nicht länger als 32 Zeichen sein, um die Vorgaben von SAS zur maximalen Länge von Variablennamen zu folgen. Außerdem sind die Inhalte in column_name indiziert, es kann also kein Variablenname doppelt vergeben werden, was Verwechslungen vermeidet. Zu jeder Variable in columns wird ein Status definiert, der Status ist in der Nachschlagetabelle „Column_Status“ definiert. Status:
„Identify“: Identifizierung von ID-Variablen der einzelnen Tabellen
„Open“: Möglichkeit des Filterns, um noch zu bearbeitende Variablen zu identifizieren
„Drop“: Löschen von Variablen aus Analyse dataset (Überflüssige Variablen aus dem Export)
„Delete“: Variablen, die nach einem Update der Importtabellen nicht mehr vorhanden sind.
„Done“: Variableninformation vollständig

Bei der Bearbeitung der Variableninformationen kann der Status verwendet werden, um beispielsweise

3.1.4 Table_Columns

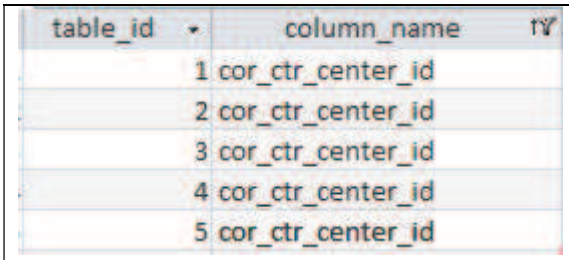
Die Tabelle table_columns verbindet Informationen zu Spalten mit den ursprünglichen Importtabellen. Jede Zeile enthält den Spaltennamen und die Tabellen ID der Importtabelle.


table_columns

Feldname	Felddatentyp
ID	AutoWert
table_id	Zahl
column_name	Kurzer Text

Abbildung 5: Tabelle Table_Columns

Eine Ausnahme bilden die Identifikationsvariablen. Da Spaltennamen eindeutig vergeben werden müssen, kann jede ID Variable nur einmal in columns angelegt werden. Zu diesem einzelnen Eintrag müssen dann alle Importtabellen angelegt werden, die die Variable enthalten. Es gibt also zu jeder einzelnen ID Variable mehrere Zeilen mit gleichem column_name und unterschiedlicher table_id.



table_id	column_name
1	cor_ctr_center_id
2	cor_ctr_center_id
3	cor_ctr_center_id
4	cor_ctr_center_id
5	cor_ctr_center_id

Abbildung 6: Beispiel ZentrumsID in 5 unterschiedlichen Importtabellen

3.2 Beziehungen

Neben Tabellen werden in Access Datenbanken auch Beziehungen zwischen den Tabellen und Formulare zur Dateneingabe definiert.

In Access werden Beziehungen zwischen Tabellen definiert um die Integrität der eingeebenen Informationen zu sichern. Die Beziehungen können über eine in Access definiert und bearbeitet werden. Das Diagramm ist in Abbildung 7 dargestellt.

Durch die Beziehungen zwischen den Tabellen ist es möglich, Informationen aus anderen Tabellen nachzuschlagen, beispielsweise werden Informationen aus der Tabelle column_status in der Tabelle columns verwendet.

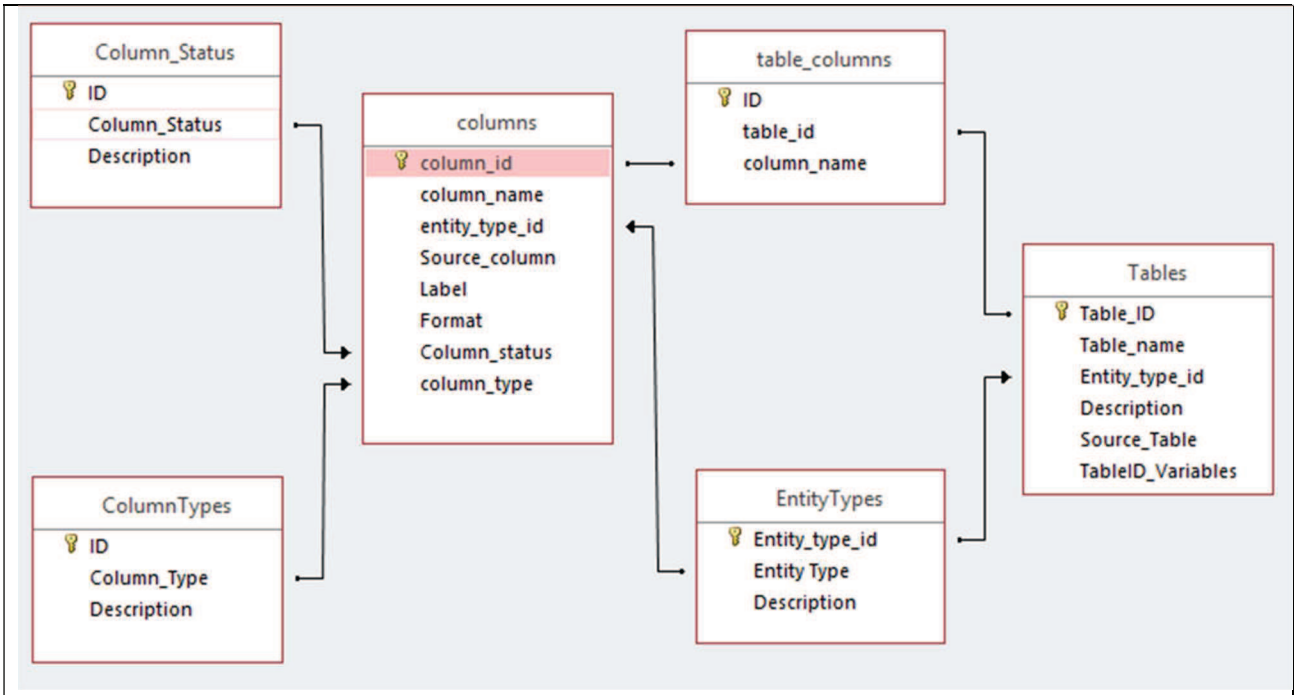


Abbildung 7: Diagramm der Beziehungen in Access

3.3 Formulare

Die Formulare bilden die Benutzeroberfläche in Access. Die Bearbeitung über Tabellen ist oft mühsam und unübersichtlich. Bei Formularen können neben einfachen Dateneingaben auch Informationen durch Dropdown Felder erleichtert und vom Entwickler über Lookup Tabellen gesteuert werden.

Neben der Dateneingabe können die Informationen in den Formularen auch verwaltet und gefiltert werden. Die Formulare in Access sind in Abbildungen 8 – 10 dargestellt.

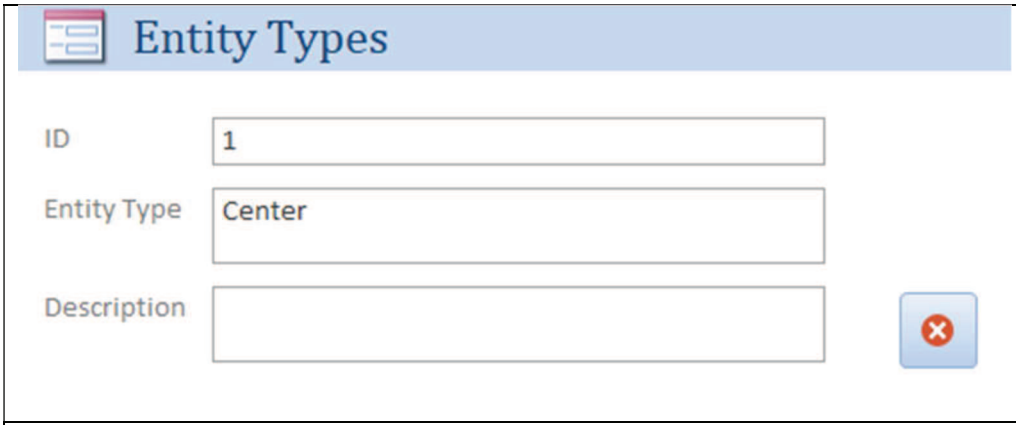


Abbildung 8: Formular EntityTypes

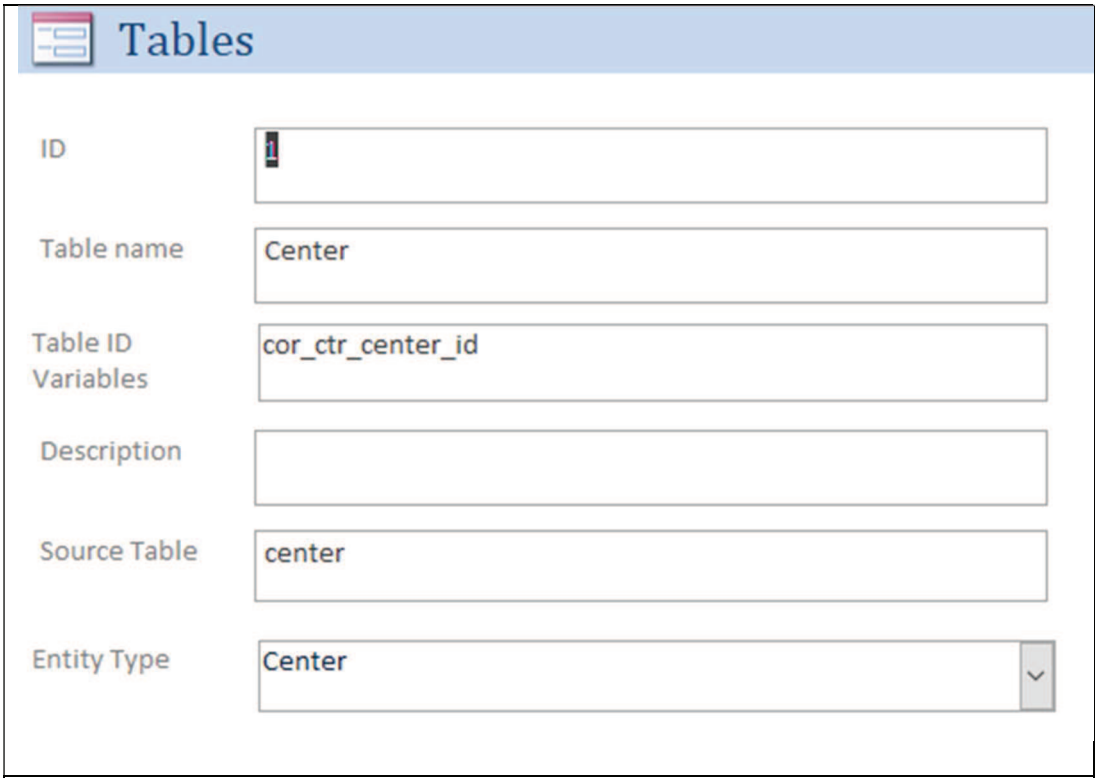


Abbildung 9: Formular Tables

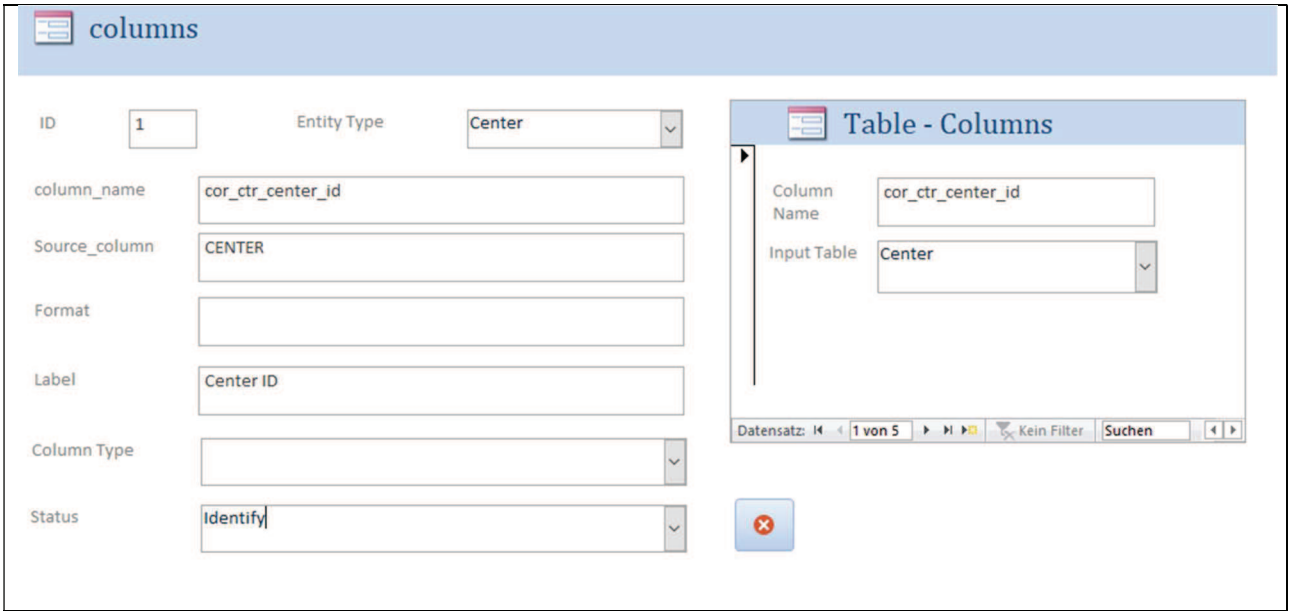


Abbildung 10: Formular Columns inclusive Unterformular Table-Columns

Im Formular Columns wird mit einem Unterformular die Eingabe in die Tabelle Table_Columns gesteuert.

4 SAS-Programme

Die SAS-Programmierung für den Aufruf der Access Datenbank teilt sich in zwei Teile: Snapshot und Restructure. In den SAS-Programme werden Datasteps, Prozedur: Proc Datasets und CALL Funktion verwendet. Der vollständige SAS-Code ist Anhang A zu entnehmen.

4.1 Snapshot

Im SAS-Programm snapshot.sas werden die Informationen aus der Access Datenbank ausgelesen. Die einzelnen ausgelesenen Tabellen werden durch mehrere merge-steps miteinander verbunden. Als Ergebnis werden zwei SAS-Tabellen:

- work_table mit Informationen zu den Import und den zugehörigen Entity Types
- work_table_columns mit allen Informationen zu den einzelnen Spalten der Importtabellen und der Informationen, die in Formular Columns eingetragen wurden, in die SAS-library snapshot abgelegt.

4.2 Restructure

Im SAS-Programm restructure.sas werden die Informationen, die in die Access Datenbank eingepflegt wurden, auf die vorhandenen Importtabellen angewendet. Das Ergebnis sind SAS-datasets zu jedem definierten Entity Type, die in einer permanenten SAS-library analysis abgelegt werden.

Im Programm wird zunächst das SAS-Makro „Columns“ definiert. Das Makro steuert die Umbenennung und die Definition der Labels und der Formate einer Spalte in jedem SAS-dataset.

Nachdem die in snapshot.sas erstellten SAS-datasets aus der library snapshot kopiert wurden, werden im ersten Schritt die ID-Variablen in den einzelnen Importtabellen umbenannt. Dies dient zur Vorbereitung des nächsten Schritts in dem die SAS-datasets zu jedem einzelnen Entity Type aus den Importtabellen definiert werden.

Die restlichen Columns der Entity Type Tabellen werden mit Hilfe des SAS-Makros Columns und des SAS-Befehls `call execute` aufgerufen. In den letzten Schritten werden alle Spalten mit dem Status „Delete“ aus dem dataset gelöscht und alle Entity Type datasets werden in der SAS-library analysis abgelegt.

5 Anwendung der Access Datenbank

Ein kurzer Überblick über die Anwendung der Access Datenbank:

1. Import der Daten in eine SAS-library

   Beim Import aus verschiedenen Importquellen ist darauf zu achten, dass alle ID Variablen genau definiert sind und konsistent in allen Importtabellen enthalten sind.

2. Identifikation der Entity Type Struktur zusammen mit Identifikationsvariablen

   Gruppierung der Importtabellen, die hierarchische Struktur wird durch die Identifikationsvariablen vorgegeben.

3. Anlegen der Entity Types in Access

   Die identifizierten Entity Types werden in der Access Datenbank im Formular Entity Type zusammen mit einer kurzen Beschreibung angelegt.

4. Anlegen der Importtabellen in Access

   Nach dem Anlegen der Entity Types wird jede einzelne Importtabelle im Formular Tables definiert. Die Identifikationsvariablen aus Schritt 2 werden hier bei den Importtabellen jeweils mit angegeben.

5. Eintragen der Columns aus Importtabellen (hier können Import/Export Access Anwendungen hilfreich sein)

   Zur Bearbeitung der Columns können neben der einzelnen Eingabe und Anlegen der Spalten über das Formular columns auch die Import- bzw. Export-Tools von Access verwendet werden. Zunächst wird jede Spalte der Importtabellen im Formular columns eingegeben. Zu jedem Eintrag muss mindestens die Source_Column, also der ursprüngliche Spaltenname, column_name also der neue Spaltenname, und der zukünftige Entity Type angegeben werden. Ein Großteil der Informationen kann aus den Importtabellen mit Hilfe von proc contents ausgelesen werden.

6. Anlegen der Table_Columns für jeden Eintrag in columns, Anlegen mehrerer Table_Columns für die Identifikationsvariablen

   Bei der Bearbeitung der einzelnen Spalten im Formular Columns wird nun zu jeder Spalte eine Verknüpfung mit der ursprünglichen Importtabelle über das Unterformular Table-Columns erstellt. Zu jeder Spalte wird angegeben, aus welcher Importtabelle die Spalte stammt. Bei der Bearbeitung der Identifikationsvariablen ist es wichtig, dass zu einer Spalte so viele Table-Columns Einträge angegeben

werden, wie es Importtabellen gibt, die diese Identifikationsvariable enthalten. Der Status der Identifikationsvariablen im Formular columns ist „Identify“.

7. Bearbeiten der einzelnen in Spalten in columns-Formular

   Sobald alle Spalten angelegt wurden und die Identifikationsvariablen belegt wurden, können nun alle weiteren Spalten bearbeitet werden. Dieser Schritt ist je nach Umfang der Importtabellen langwierig, kann aber durch die Struktur von Access auf mehrere Mitarbeiter aufgeteilt werden.

8. Durchführung der SAS Programme

   Die Importtabellen werden in der library work abgelegt. Zunächst werden die SAS libraries snapshot und analysis angelegt, danach werden die Programme snapshot.sas und restructure.sas in SAS ausgeführt. Das Ergebnis der Restrukturierung kann in der library analysis geprüft werden.

6 Zusammenfassung und Ausblick

Die Anwendungsbereiche der Access Datenbank mit den SAS-Programmen kann neben den gezeigten Funktionen noch erweitert werden. Einige Beispiele sind z.B. Historisierung: in Access können mit Hilfe von Historisierungstabellen alte Datenstände gespeichert werden. Diese können jederzeit angezeigt werden und auch zu einem späteren Zeitpunkt wiederverwendet werden.

- Die Access Datenbank kann helfen, bestimmte Unternehmensrichtlinien durchzusetzen. Hierzu zählen bspw. Regeln zur Benennung von Auswertungsvariablen (einzelne Namensteile können in Nachschlagetabellen in Access definiert werden)
- Mit Hilfe einer weiteren Tabelle für die Labels können Labels in mehreren Sprachen, oder mehreren Längen einfach angelegt werden.
- Mit Hilfe der Angaben zu Importtabellen kann die Integrität der Importtabellen überprüft werden. Gibt es strukturelle Fehler, wie beispielsweise doppelte PatientenIDs, können Fehlermeldungen an zentraler Stelle die Fehlersuche erleichtern.

Ein Zusammenspiel zwischen SAS und externen Systemen wie Access kann die alltägliche Arbeit im Datenmanagement erheblich erleichtern. Projektspezifische Importdaten können mit Hilfe des vorgestellten Systems von einer projektübergreifenden Access Datenbank und SAS-Programmen in projektspezifische Analysedatasets überführt werden. Die projektübergreifenden Systeme können losgelöst von einzelnen Projekten weiterentwickelt werden, Erfahrungen aus den einzelnen Projekten tragen zur Weiterentwicklung des Systems bei.

7 Beispiel

Die Beispieldaten zusammen mit einer befüllten Access Datenbank und passenden SAS-Programmen können unter dem folgenden Link heruntergeladen werden.

Abbildung 11: QR-Code für Beispiel
https://cloud.ihf.de/index.php/s/dulLZjDvtmSiVSp

Passwort: KSFEBerlin2019

Anhang A SAS-Programme

1 Snapshot.sas

```
/*Get access database information in SAS*/
libname metabase pcfiles path="C:\Users\lober\ownCloud\KSFE
2019\Beispiel\metabase_Beispiel.accdb" access=readonly;

data snapshot.tables;
set metabase.tables;
run;
data snapshot.entitytype;
set metabase.EntityTypes;
run;
data snapshot.columns;
set metabase.columns;
run;
data snapshot.table_columns;
set metabase.table_columns;
run;
libname metabase clear;

/*columns_tables mit Info auffüllen*/

/*Table info to columns_tables*/
proc sort data=snapshot.table_columns  out= table_columns ;
by table_id ;
run;

proc sort data=snapshot.tables  out= snapshot.tables ;
by table_id ;
run;

data table_columns;
```

```
merge  table_columns(in=columns) snapshot.tables ;
by table_id ;
run;

/*Column info to columns_tables*/
proc sort data= table_columns out= table_columns ;
by column_name ;
run;

proc sort data=snapshot.columns out= columns (where =(
Column_status ne "3")) ;
by column_name ;
run;

data table_columns;
merge table_columns columns(in=col drop = Source_column);
by column_name ;
if col;
run;

/*Entity Type info to columns_tables */
proc sort data=table_columns  out=table_columns  ;
by entity_type_id ;
run;

proc sort data= snapshot.entitytype out= snapshot.entitytype ;
by entity_type_id ;
run;

data snapshot.work_table_columns;
merge table_columns(in=col) snapshot.entitytype  ;
in=col;
by  entity_type_id;
run;


/*Entity Type info to tables */
proc sort data=snapshot.tables  out= snapshot.tables ;
by entity_type_id ;
run;

data snapshot.work_table;
merge snapshot.tables snapshot.entitytype ;
by entity_type_id ;
run;
```

2 Restructure.sas

```
%MACRO COLUMNS(
   INPUT_DATA =     ,
   VARNAME_NEW =    ,
   VARNAME_OLD =    ,
```

```
   FORMAT =         ,
   LIBRARY = work        ,
   LABEL =
   );

proc datasets lib=&LIBRARY memtype=data ;
modify &INPUT_DATA ;
rename &VARNAME_OLD = &VARNAME_NEW;
attrib &VARNAME_NEW label= "&LABEL."
%IF %LENGTH(&FORMAT) > 0 %THEN %DO;
format=&FORMAT
%END;
;
quit;
%MEND;

option mprint;

data table;
set snapshot.work_table;
run;
data table_columns;
set snapshot.work_table_columns;
run;

%*ID-Variablen in Source Datasets umbenennen;
data _null_;
set table_columns(where= (Column_status in ("5")));
call execute('%NRSTR(%%)%UNQUOTE(COLUMNS)(' );
call execute(' INPUT_DATA ='!!    trim(Source_Table)  !!', '
);
call execute(' VARNAME_NEW ='!!   trim(column_name) !!',' );
call execute(' VARNAME_OLD = '!! trim(Source_column)   !!',
'    );
call execute(' FORMAT = '!! trim(Format)     !!','    );
call execute(' LABEL = ' !! trim(Label) );
call execute(' );');
run;


%*Sort: erste Input Tabelle pro Entity Type wird direkt in
Entity Type Dataset geschrieben;
data _NULL_;
set table;
by Entity_type_id;
if first.Entity_type_id then do;
call execute(' proc sort data= '!! trim(Source_Table) !!' out='
!! trim(Entity_Type) !! ' ;');
end;
else do;
call execute(' proc sort data= '!! trim(Source_Table) !!' out='
!! trim(Source_Table) !! ' ;');
```

```
end;
call execute('by ' !!  trim(tableid_variables) !! ' ;');
call execute('run;  ');
run;

%* Entity Type Datasets ersellen ;
data _NULL_;
set table;
by Entity_type_id;
if not first.Entity_type_id then do;
call execute('data  ' !! trim(Entity_Type) !! ';');
call execute('merge '   !!   trim(Entity_Type)!!   "      "   !!
trim(Source_Table) !! ';') ;
call execute('by    ' !! trim(tableid_variables) !! ';');
call execute('run;  ');
end;
run;


%*Umbenennung der restlichen Columns;
data _null_;
set table_columns(where= (Column_status in ("1","2")));
call execute('%NRSTR(%%)%UNQUOTE(COLUMNS)(' );
call execute(' INPUT_DATA ='!!    trim(Entity_Type)   !!', '
);
call execute(' VARNAME_NEW ='!!    trim(column_name) !!',' );
call execute(' VARNAME_OLD = '!! trim(Source_column)   !!',
'    );
call execute(' FORMAT = '!! trim(Format)     !!','    );
call execute(' LABEL = ' !! trim(Label) );
call execute(' );');
run;


%*Drop columns with Column_Status = 4;
data _NULL_;
set table_columns(where= (Column_status in ("4")));
call execute(' data ' !! trim(Entity_Type) !! ';');
call execute(' set  ' !! trim(Entity_Type) !! ';') ;
if not missing(column_name) then do;
call execute(' drop ' !! trim(column_name) !! ';');
end;
call execute(' run; ');
run;

%*Copy datasets to library analysis;
data _NULL_;
set table;
by Entity_type_id;
if first.Entity_type_id then do;
call execute(' data analysis.' !! trim(Entity_Type) !! ';');
call execute(' set  ' !! trim(Entity_Type) !! ';') ;
```

```
call execute(' run; ');
end;
run;
```

Anhang B Nachschlagetabellen der Access Datenbank

1. Column_status

Tabelle 1: Column_status Tabelle in Access Datenbank

ID	Column_Status	Description
1	Open	Still work to be done
2	Done	Column information updated
3	Drop	Drop column
4	Delete	Delete column from statistical analysis
5	Identify	ID-Variable

2. Column_Types

Tabelle 2: Column_Type Tabelle in Access Datenbank

ID	Column_Type	Description
1	num	Numeric Data
2	char	Character Data