entimo

Hash Objects - Reloaded

Arne Leißner

20. KSFE 2016 in Greifswald

Hash-Objekt: Techniken & Use Cases

- Lockup-Turbo
- Data Set Split
- Join
- Sortierung
- Mehrfachlesen
- Zählung
- De-Duplizierung
- Fuzzy Merge
- Relationsprüfung
- Top-<n>%
- Gruppenstatistiken
- Summierung
- Rekursives Lockup
- Durchwanderung
- Compare
- ...

Hash-Objekt: Techniken & Use Cases

NEU

Vergleich von Hash-Objekten (Informationsmengen)

Rekursive Verarbeitung mittels Hash-Objekt

NEU

Hash-Objekte: Charakteristika

- Zusätzliche dynamische Datenstruktur(en) im Direktzugriff des DATA Steps
- Können bedingungsabhängig zur Execution Time erzeugt werden und sind flüchtig („leben“ nur für die Dauer eines Data Steps)
- Einträge über Schlüsselwerte indiziert und referenziert (Speicherposition via Hashkey)
  -> unsortierte Ablage
- Simple und composite Keys unterstützt (num, char)
- Sehr schnelle Suchalgorithmen unabhängig von der Anzahl der Einträge (Hashfunktionen in Memory)

Hash-Objekte: Charakteristika

- Freie Navigation (via Schlüssel od. sequentiell)
  -> Direktadressierung
  -> Multiadressierung
  -> schrittweise vorwärts / rückwärts mittels Iterator
- Können und dürfen groß werden
- Dynamische und agile Techniken für das anlegen, modifizieren und löschen von Einträgen (manuell und automatisch)
- Verlinkung zu Data Step Variablen über den Namen
  -> Automatischer Informationsaustausch zwischen PDV und Hash-Objekt (wahlweise auch manuelle Zuweisungen)

Hash-Objekte: Charakteristika

- Einfacher Import / Export von SAS Data Sets
- Eingebaute Summenfunktion
- Steuerung des Verhaltens via attribute tags
- Objektorientiert (Methoden, Attribute, Dot-Notation)

Hash-Algorithmus: Beispiel

Mein Hash-Objekt:
Der alte Schlüsselkasten
in der Hotelrezeption
mit 11 nummerierten Haken

Was zu verteilen ist:
16 Hotelschlüssel mit
Zimmernummern
1. OG: 11, 12, 14, 15, 16, 17
2. OG: 21, 22, 23, 24, 25, 26
3. OG: 31, 32, 33, 34

Hash-Algorithmus: Beispiel

Zimmernummer	Modulo 11	+1 = Haken
11	0	1
12	1	2
14	3	4
15	4	5
16	5	6
17	6	7
21	10	11
22	0	1
23	1	2
24	2	3
25	3	4
26	4	5
31	9	10
32	10	11
33	0	1
34	1	2

Hash-Algorithmus: Beispiel

Haken	Zimmernummern
1	11, 22, 33
2	12, 23, 34
3	24
4	14, 25
5	15, 26
6	16
7	17
8	
9	
10	31
11	21, 32

Gleich schnelle Suche:

Wo hängt Schlüssel zu Zimmer 16?
MOD(16, 11) + 1 = **6**

Wo hängt Schlüssel zu Zimmer 1016?
MOD(1016, 11) + 1 = **5**

Mehrfach besetzte Haken auch dann möglich, wenn mehr Haken als Zimmer vorhanden sind.

Gute (gleichmäßige) Verteilung mittels guter Hash-Funktion und angemessener Größe des Hash-Objektes

Alle Hash-Beispiele: One-Step-Samples

```
DATA …;
    …
    Deklaration und Initialisierung

    …

    Schreiben und Lesen

    …

    Suchen und Navigation

    …

    Ausgabe

    …
RUN;
```

```
<IF … THEN DO;>
   DCL HASH hash (<attribute tags>);
   hash.DEFINEKEY (<key-definition>);
   hash.DEFINEDATA (<data definition>);
   hash.DEFINEDONE();
   DCL HITER iter (<attribute tags>);
<END>;
```

```
hash.ADD(<arguments>);
hash.REF(<arguments>);
hash.REPLACE(<arguments>);
hash.REMOVE(<arguments>);
hash.CLEAR();
```

```
hash.CHECK(<arguments>);
hash.REF(<arguments>);
hash.FIND(<arguments>);
hash.FIRST();
hash.LAST();
hash.NEXT();
hash.PREV();
```

```
hash.OUTPUT(<arguments>);
```

Bsp. 1: Vergleich von Hash-Objekten

Aufgabe:	Prüfe die Struktur einer SAS-Datei gegen eine durch Metadaten deklarierte Soll-Vorgabe	
Fallbeispiel:	Patientendaten mit Adverse Events	
Features:	attribute-tag: DATASET	
	Data Set Optionen	
	Hash Iterator	
	Methoden	EQUALS()
		FIRST()
		NEXT()
		FIND()
		REMOVE()
		REPLACE()
	Attribut	NUM_ITEMS

Bsp. 1: Patientendatei

	STUDYID	USUBJID	AETERM	AESTDTC	AESEQ	AEDECOD	AEBODSYS	AESEV
1	CDISCPILOT01	01-701-1015	VERBATIM_0995	2014-01-03	1	APPLICATION SITE ERYTHEMA	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	MILD
2	CDISCPILOT01	01-701-1015	VERBATIM_1126	2014-01-09	3	DIARRHOEA	GASTROINTESTINAL DISORDERS	MILD
3	CDISCPILOT01	01-701-1015	VERBATIM_1219	2014-01-03	2	APPLICATION SITE PRURITUS	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	MILD
4	CDISCPILOT01	01-701-1023	VERBATIM_0300	2012-08-07	1	ERYTHEMA	SKIN AND SUBCUTANEOUS TISSUE DISORDERS	MILD
5	CDISCPILOT01	01-701-1023	VERBATIM_0300	2012-08-07	4	ERYTHEMA	SKIN AND SUBCUTANEOUS TISSUE DISORDERS	MILD
6	CDISCPILOT01	01-701-1023	VERBATIM_1549	2012-08-07	2	ERYTHEMA	SKIN AND SUBCUTANEOUS TISSUE DISORDERS	MODERAT
7	CDISCPILOT01	01-701-1023	VERBATIM_1650	2012-08-26	3	ATRIOVENTRICULAR BLOCK SECOND DEGREE	CARDIAC DISORDERS	MILD
8	CDISCPILOT01	01-701-1028	VERBATIM_0578	2013-08-08	2	APPLICATION SITE PRURITUS	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	MILD
9	CDISCPILOT01	01-701-1028	VERBATIM_1157	2013-07-21	1	APPLICATION SITE ERYTHEMA	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	MILD
10	CDISCPILOT01	01-701-1034	VERBATIM_0555	2014-11-02	2	FATIGUE	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	MILD
11	CDISCPILOT01	01-701-1034	VERBATIM_1219	2014-08-27	1	APPLICATION SITE PRURITUS	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	MILD
12	CDISCPILOT01	01-701-1047	VERBATIM_0130	2013-02-12	1	HIATUS HERNIA	GASTROINTESTINAL DISORDERS	MODERAT
13	CDISCPILOT01	01-701-1047	VERBATIM_0130	2013-02-12	2	HIATUS HERNIA	GASTROINTESTINAL DISORDERS	MODERAT
14	CDISCPILOT01	01-701-1047	VERBATIM_0197	2013-03-10	4	BUNDLE BRANCH BLOCK LEFT	CARDIAC DISORDERS	MILD
15	CDISCPILOT01	01-701-1047	VERBATIM_0579	2013-03-06	3	UPPER RESPIRATORY TRACT INFECTION	INFECTIONS AND INFESTATIONS	MILD
16	CDISCPILOT01	01-701-1097	VERBATIM_0300	2014-01-03	1	ERYTHEMA	SKIN AND SUBCUTANEOUS TISSUE DISORDERS	MILD
17	CDISCPILOT01	01-701-1097	VERBATIM_0758	2014-03-21	5	PRURITUS GENERALISED	SKIN AND SUBCUTANEOUS TISSUE DISORDERS	MODERAT
18	CDISCPILOT01	01-701-1097	VERBATIM_0758	2014-04-19	7	PRURITUS GENERALISED	SKIN AND SUBCUTANEOUS TISSUE DISORDERS	MODERAT
19	CDISCPILOT01	01-701-1097	VERBATIM_0990	2014-02-20	2	PRURITUS GENERALISED	SKIN AND SUBCUTANEOUS TISSUE DISORDERS	MODERAT
20	CDISCPILOT01	01-701-1097	VERBATIM_0990	2014-03-31	6	PRURITUS GENERALISED	SKIN AND SUBCUTANEOUS TISSUE DISORDERS	MODERAT
21	CDISCPILOT01	01-701-1097	VERBATIM_1219	2014-02-21	4	APPLICATION SITE PRURITUS	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	MILD
22	CDISCPILOT01	01-701-1097	VERBATIM_1219	2014-02-21	10	APPLICATION SITE PRURITUS	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	MODERAT
23	CDISCPILOT01	01-701-1097	VERBATIM_1230	2014-04-19	8	PHARYNGOLARYNGEAL PAIN	RESPIRATORY, THORACIC AND MEDIASTINAL DISORDERS	MILD
24	CDISCPILOT01	01-701-1097	VERBATIM_1522	2014-02-20	3	APPLICATION SITE VESICLES	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	MILD

Bsp. 1: Metadatendatei

	name	label	type	length	seq	key	format	formatl	formatd	mand	null	intkey	docvar	de
1	STUDYID	Study Identifier	c	12	1	1		.	.	Yes	Yes	No	.	
2	USUBJID	Unique Subject Identifier	c	11	2	2		.	.	Yes	Yes	No	.	
3	AETERM	Reported Term for the Adverse Event	c	13	3	.		.	.	Yes	Yes	No	.	
4	AESTDTC	Start Date/Time of Adverse Event	c	10	4	.		.	.	Yes	Yes	No	.	
5	AESEQ	Sequence Number	n	8	5	4		.	.	Yes	Yes	No	.	
6	DOMAIN	Domain Abbreviation	c	2	6	3		.	.	Yes	Yes	No	.	
7	AESPID	Sponsor-Defined Identifier	c	3	7	.		.	.	No	Yes	No	.	
8	AEDECOD	Dictionary-Derived Term	c	46	8	.		.	.	Yes	Yes	No	.	
9	AEBODSYS	Body System or Organ Class	c	67	9	.		.	.	Yes	Yes	No	.	
10	AESEV	Severity/Intensity	c	8	10	.		.	.	Yes	Yes	No	.	
11	AESER	Serious Event	c	1	11	.		.	.	Yes	Yes	No	.	
12	AEREL	Causality	c	8	12	.		.	.	Yes	Yes	No	.	
13	AEOUT	Outcome of Adverse Event	c	12	13	.		.	.	Yes	Yes	No	.	
14	AESCAN	Involves Cancer	c	1	14	.		.	.	Yes	Yes	No	.	
15	AESCONG	Congenital Anomaly or Birth Defect	c	1	15	.		.	.	Yes	Yes	No	.	
16	AESDISAB	Persist or Signif Disability/Incapacity	c	1	16	.		.	.	Yes	Yes	No	.	
17	AESDTH	Results in Death	c	1	17	.		.	.	Yes	Yes	No	.	
18	AESHOSP	Requires or Prolongs Hospitalization	c	1	18	.		.	.	Yes	Yes	No	.	
19	AESLIFE	Is Life Threatening	c	1	19	.		.	.	Yes	Yes	No	.	
20	AESOD	Occurred with Overdose	c	1	20	.		.	.	Yes	Yes	No	.	
21	AEENDTC	End Date/Time of Adverse Event	c	10	21	.		.	.	Yes	Yes	No	.	
22	AESTDY	Study Day of Start of Adverse Event	n	8	22	.		.	.	Yes	Yes	No	.	
23	AEENDY	Study Day of End of Adverse Event	n	8	23	.		.	.	Yes	Yes	No	.	
24	AEDTC	Date/Time of Collection of Adverse Event	c	10	24	.		.	.	Yes	Yes	No	.	
25	AEACN	Action Taken with Study Treatment	c	30	25	.		.	.	Yes	Yes	No	.	

Bsp. 1: sashelp.vcolumn für Pat.datei

	libname	memna	memtype	name	type	length	npos	varnum	label	format	info
1	D_LIB	AE	DATA	STUDYID	char	12	24	1	Study Identifier		
2	D_LIB	AE	DATA	USUBJID	char	11	36	2	Unique Subject Identifier		
3	D_LIB	AE	DATA	AETERM	char	13	47	3	Reported Term for the Adverse Event		
4	D_LIB	AE	DATA	AESTDTC	char	10	60	4	Start Date/Time of Adverse Event		
5	D_LIB	AE	DATA	AESEQ	num	8	0	5	Sequence Number		
6	D_LIB	AE	DATA	AEDECOD	char	46	70	6	Dictionary-Derived Term		
7	D_LIB	AE	DATA	AEBODSYS	char	67	116	7	Body System or Organ Class		
8	D_LIB	AE	DATA	AESEV	char	8	183	8	Severity/Intensity		
9	D_LIB	AE	DATA	AESER	char	1	191	9	Serious Event		
10	D_LIB	AE	DATA	AEREL	char	8	192	10	Causality		
11	D_LIB	AE	DATA	AEOUT	char	12	200	11	Outcome of Adverse Event		
12	D_LIB	AE	DATA	AESCAN	char	1	212	12	Involves Cancer		
13	D_LIB	AE	DATA	AESCONG	char	1	213	13	Congenital Anomaly or Birth Defect		
14	D_LIB	AE	DATA	AESDISAB	char	1	214	14	Persist or Signif Disability/Incapacity		
15	D_LIB	AE	DATA	AESDTH	char	1	215	15	Results in Death		
16	D_LIB	AE	DATA	AESHOSP	char	1	216	16	Requires or Prolongs Hospitalization		
17	D_LIB	AE	DATA	AESLIFE	char	1	217	17	Is Life Threatening		
18	D_LIB	AE	DATA	AESOD	char	1	218	18	Occurred with Overdose		
19	D_LIB	AE	DATA	AEENDTC	char	10	219	19	End Date/Time of Adverse Event		
20	D_LIB	AE	DATA	AESTDY	num	8	8	20	Study Day of Start of Adverse Event		
21	D_LIB	AE	DATA	AEENDY	num	8	16	21	Study Day of End of Adverse Event		
22	D_LIB	AE	DATA	AEDTC	char	10	229	22	Date/Time of Collection of Adverse Event		
23	D_LIB	AE	DATA	AEACN	char	30	239	23	Action Taken with Study Treatment		

Bsp. 1: Lösungsidee und Schritte

(1) Die Metadatendatei und Subset aus sashelp.vcolumn in zwei Hash-Objekte einlesen.

(2) Prüfen, ob Hash-Objekte in Inhalt und Struktur identisch sind

(3) In der Patientendatei nicht enthaltene optionale Spalten aus der Betrachtung ausschließen und erneut prüfen

(4) Abweichungen identifizieren

Bsp. 1: Schritt 1 (Initialisierung)

```
data _null_;
   /* 0 Initialisierung */
   /* 0.1 Variablen deklarieren */
   IF 0 THEN SET m_lib.meta_ae;

   /* 0.2 Metadaten in Hash Object einlesen */
   DCL HASH h_meta (DATASET: "m_lib.meta_ae");
   h_meta.DEFINEKEY("name");
   h_meta.DEFINEDATA("name", "label", "type", "length", "seq", "format");
   h_meta.DEFINEDONE();

   /* 0.3 Strukturinfos der Datendatei via Dictionary Table einlesen" */
   DCL HASH h_data (DATASET: "sashelp.vcolumn(WHERE=(libname = 'D_LIB'
                    and memname = 'AE') RENAME = (varnum = seq))");
   h_data.DEFINEKEY("name");
   h_data.DEFINEDATA("name", "label", "type", "length", "seq", "format");
   h_data.DEFINEDONE();
...

NOTE: There were 25 observations read from the data set M_LIB.META_AE.
NOTE: There were 23 observations read from the data set SASHELP.VCOLUMN.
      WHERE (libname='D_LIB') and (memname='AE');
```

Bsp. 1: Einwürfe

- Prüfen, was geladen wurde

```
...
   h_data.OUTPUT(DATASET "work.data");
   h_meta.OUTPUT(DATASET "work.meta");
...

NOTE: The data set WORK.DATA has 23 observations and 6 variables.
NOTE: The data set WORK.META has 25 observations and 6 variables.
```

- Unerwartete Log-Ausgabe wenn Hash-Objekt aus View geladen wird (unabhängig, ob mit oder ohne WHERE Data Set Option):

  NOTE: DATA STEP stopped due to looping.

  -> Vermeintliches EOF-Problem (EOF nicht erreicht)

Bsp. 1: Schritt 2 (Vergleich)

```
...
   /* 1.1 Vollständige Übereinstimmung mit deklarierter Struktur */
   h_meta.EQUALS(HASH: "h_data", RESULT: eq);
   IF eq THEN PUT "Alles OK, alle deklarierten Variablen in Datendatei";
...
```

Für den EQUALS-Vergleich herangezogen werden:

- Die Speichergröße im Hauptspeicher
- Die Zahl der Einträge
- Die definierten Strukturen bzgl. Schlüssel- und Datenvariablen
- Die Schlüssel- und Datenwerte an den entsprechenden Positionen

Bsp. 1: Schritt 3 (opt. Variable)

```
...
   /* 1.2 Übereinstimmung, falls einzelne opt. Var. nicht in Datendatei */
   IF NOT eq THEN DO;
      /* 1.2.1 Optionale Attribute in weiteres Hash Object einlesen */
      DCL HASH h_opt (DATASET: "m_lib.meta_ae(WHERE = (mand = 'N')
                                KEEP = name mand seq)");
      h_opt.DEFINEKEY("name");
      h_opt.DEFINEDATA("name", "seq");
      h_opt.DEFINEDONE();

      /* 1.2.2 Iteratoren definieren */
      DCL HITER h_opt_iter("h_opt");

      /* 1.2.3 Metadaten um nicht vorhandene opt. Spalten reduzieren */
      IF h_opt.NUM_ITEMS > 0 THEN DO;
         rc = h_opt_iter.FIRST();
         DO WHILE (rc = 0);
            IF h_data.FIND() THEN DO;
               h_meta.REMOVE();
            END;
            rc = h_opt_iter.NEXT();
         END;
         h_meta.EQUALS(HASH: "h_data", RESULT: eq);
         IF eq THEN PUT "Alles OK, einige opt. Var. nicht in Datendatei";
      END;
   END;
...
```

Bsp. 1: Schritt 3a (SEQ-Nummern)

```
...
   /* 1.2.2 Iteratoren definieren */
   ...
      DCL HITER h_meta_iter("h_meta");

      /* 1.2.3 Metadaten um nicht vorhandene opt. Spalten reduzieren */
      IF h_opt.NUM_ITEMS > 0 THEN DO;
         ...
            IF h_data.FIND() THEN DO;
               h_meta.REMOVE();
               /* SEQ-Nummern anpassen */
               rem_seq = seq;
               rc = h_meta_iter.FIRST();
               DO WHILE (rc = 0);
                  IF seq > rem_seq THEN DO;
                     seq = seq - 1;
                     rc = h_meta.REPLACE();
                  END;
                  rc = h_meta_iter.NEXT();
               END;
...
```

Bsp. 1: Schritt 4a (Anzahl Variablen)

```
...
   /* 1.3 Abweichungen im Detail */
   IF NOT eq THEN DO;
      /* 1.3.1 Anzahl der Variablen unterschiedlich*/
      IF h_meta.NUM_ITEMS ne h_data.NUM_ITEMS THEN
         PUT "Fehler: Variablenanzahl unterschiedlich";
...
```

Bsp. 1: Schritt 4b (unterschiedl. Var.)

```
...
      /* 1.3.2 Variablen aus Metadaten nicht in Datendatei */
      rc = h_meta_iter.FIRST();
      DO WHILE (rc = 0);
         IF h_data.FIND() THEN not_in_data = 1;
         rc = h_meta_iter.NEXT();
      END;
      IF not_in_data THEN PUT "Fehler: Variable(n) fehlen in Datendatei";
...
```

```
...
      /* 1.3.3 Variablen aus Datendatei nicht in Metadaten */
      DCL HITER h_data_iter("h_data");

      rc = h_data_iter.FIRST();
      DO WHILE (rc = 0);
         IF h_meta.FIND() THEN not_in_metadata = 1;
         rc = h_data_iter.NEXT();
      END;
      IF not_in_meta THEN PUT "Fehler: Undeklarierte Variable(n)";
...
```

Bsp. 1: Schritt 4c (unterschiedl. Props.)

```
...
      /* 1.3.4 Var.infos unterschiedlich (Typ|Länge|Label|Format|Seq) */
      rc = h_meta_iter.FIRST();
      DO WHILE (rc = 0);
         type_m = type; length_m = length; label_m = label;
         format_m = format; seq_m = seq;
         IF NOT h_data.FIND() THEN DO;
            if type ne type_m or length ne length_m or label ne label_m or
               format ne format_m or seq ne seq_m THEN diff_values = 1;
         END;
         rc = h_meta_iter.NEXT();
      END;
      IF diff_values THEN PUT "Fehler: Unterschiede in den Properties";
   END;
RUN;
```

Programmerweiterungen:

- *Protokollierungen des Strukturvergleichs im Detail*
- *Ergänzung um inhaltliche Konsistenzprüfungen (Schlüsselkriterien, Nullable, Passfähigkeit der Formate, ….) ebenfalls via Hash-Objekt*

Bsp. 2: Rekursive Verarbeitung

Aufgabe:	Ermittle alle Dateien in einem Verzeichnisbaum ohne rekursive Programmierung
Fallbeispiel:	Auswertung aller Unterverzeichnisse ab einem Startverzeichnis auf dem lokalen PC-Laufwerk
Features:	attribute-tag: ORDERED Hash Iterator
	Methoden FIRST() LAST() ADD() REMOVE()
	Attribut NUM_ITEMS

Bsp. 2: Der Verzeichnisbaum

Name	Ordner +	Elementtyp
Root	D:\	Dateiordner
A	D:\Root\	Dateiordner
AA	D:\Root\A\	Dateiordner
AAA	D:\Root\A\AA\	Dateiordner
Datei AAA1	D:\Root\A\AA\AAA\	Datei
AAB	D:\Root\A\AA\	Dateiordner
Datei AAB1	D:\Root\A\AA\AAB\	Datei
Datei AA1	D:\Root\A\AA\	Datei
AB	D:\Root\A\	Dateiordner
ABA	D:\Root\A\AB\	Dateiordner
ABB	D:\Root\A\AB\	Dateiordner
ABC	D:\Root\A\AB\	Dateiordner
Datei AB1	D:\Root\A\AB\	Datei
Datei AB2	D:\Root\A\AB\	Datei
Datei AB3	D:\Root\A\AB\	Datei
Datei AB4	D:\Root\A\AB\	Datei
Datei A1	D:\Root\A\	Datei
Datei A2	D:\Root\A\	Datei
B	D:\Root\	Dateiordner
C	D:\Root\	Dateiordner
Datei 1	D:\Root\	Datei
Datei 2	D:\Root\	Datei
Datei 3	D:\Root\	Datei
Datei 4	D:\Root\	Datei

25 Dateien
in 19 Ordnern

Bsp. 2: Lösungsidee und Schritte

(1) Hash-Objekt zur Verwaltung von Verzeichnispfaden anlegen und mit dem Root-Pfad als zunächst einzigem Eintrag füllen.

(2) Schleife über das Hash-Objekt ausführen:
Solange noch Einträge (Verzeichnispfade) vorhanden sind:
- den letzten Verzeichniseintrag auswählen
- alle in dem Verzeichnis gefundenen Dateien in SAS-Datei schreiben
- alle in dem Verzeichnis gefundenen Verzeichnisse (Unterverzeichnisse) dem Hash-Objekt hinzufügen
- den aktuell bearbeiteten Verzeichniseintrag im Hash-Objekt löschen

Bsp. 2: Schritt 1 (Initialisierung)

```
%LET root_dir = d:\root;

data work.files(keep = path member);
   /* 1. Initialisierung */
   /* 1.1 Hash-Objekt und Iterator anlegen */
   LENGTH no_dir 8 path $ 500;

   DCL HASH h_dir(ORDERED: "A");
   h_dir.DEFINEKEY("no_dir");
   h_dir.DEFINEDATA("no_dir", "path");
   h_dir.DEFINEDONE();
   DCL HITER h_dir_iter("h_dir");

   /* 1.2 Startverzeichnis setzen*/
   path = "&root_dir";
   anz_dir + 1;
   no_dir = 1;
   h_dir.ADD();
...
```

Bsp. 2: Schritt 2 (Schleife)

```
...
   /* 2. Verzeichnisanalyse: Arbeite Verzeichnisliste von hinten ab */
   stop = 0;
   DO WHILE (NOT stop);
      /* 2.1 Adressiere den letzten Verzeichniseintrag */
      ...
      /* 2.2 Ermittle und behandle die Member aus dem Verzeichnis */
      ...
      /* 2.2.1 Füge Verzeichnisse dem Hash-Objekt hinzu */
      ...
      /* 2.2.2 Gebe Dateien in SAS-Datei aus */
      ...
      /* 2.3 Lösche aktuellen Eintrag aus Hash-Objekt oder setze stop */
      ...
   END;
...
run;
```

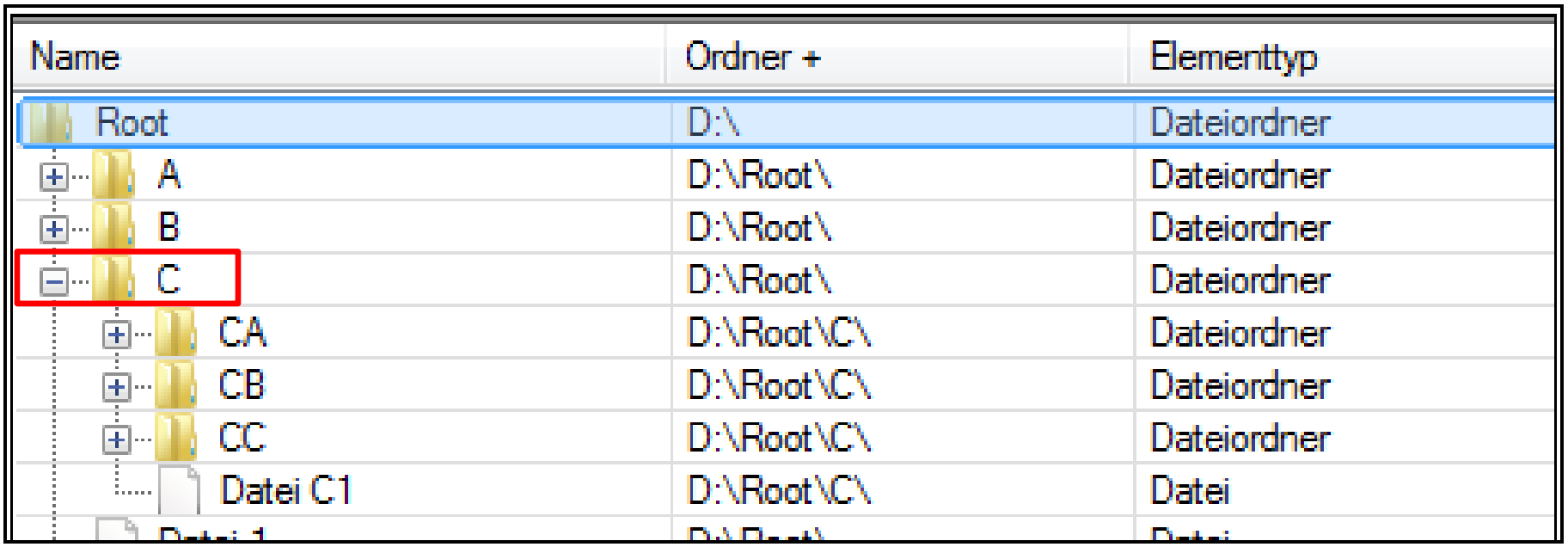
Bsp. 2: Schritt 2 (1. Durchlauf)

Name	Ordner +	Elementtyp
Root	D:\	Dateiordner
A	D:\Root\	Dateiordner
B	D:\Root\	Dateiordner
C	D:\Root\	Dateiordner
Datei 1	D:\Root\	Datei
Datei 2	D:\Root\	Datei
Datei 3	D:\Root\	Datei
Datei 4	D:\Root\	Datei

Vor Durchlauf 1	
Hash-Objekt	**SAS-Datei**
[1] D:\root	

Nach Durchlauf 1	
Hash-Objekt	**SAS-Datei**
[2] d:\root\A	Datei 1
[3] d:\root\B	Datei 2
[4] d:\root\C	Datei 3
	Datei 4

Bsp. 2: Schritt 2 (2. Durchlauf)

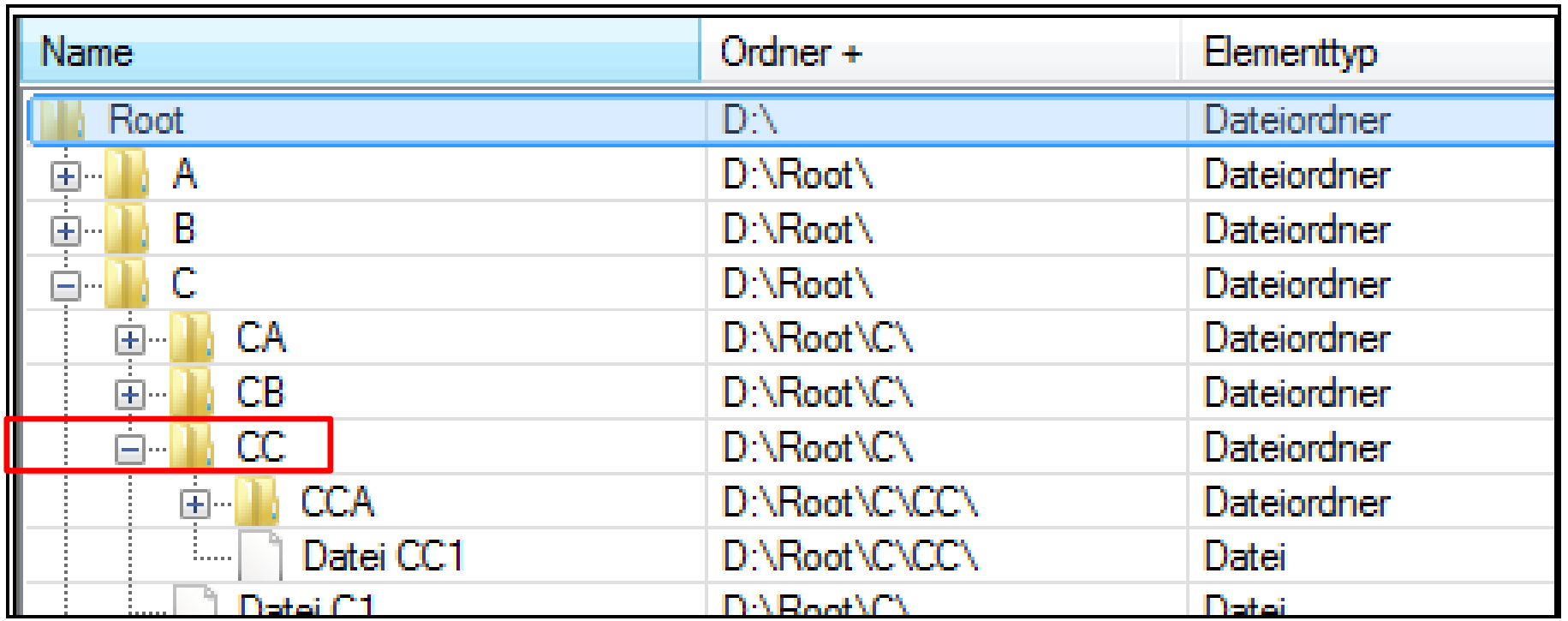


Name	Ordner +	Elementtyp
Root	D:\	Dateiordner
A	D:\Root\	Dateiordner
B	D:\Root\	Dateiordner
C	D:\Root\	Dateiordner
CA	D:\Root\C\	Dateiordner
CB	D:\Root\C\	Dateiordner
CC	D:\Root\C\	Dateiordner
Datei C1	D:\Root\C\	Datei

Nach Durchlauf 1	
Hash-Objekt	**SAS-Datei**
[2] d:\root\A	Datei 1
[3] d:\root\B	Datei 2
[4] d:\root\C	Datei 3
	Datei 4

Nach Durchlauf 2	
Hash-Objekt	**SAS-Datei**
[2] d:\root\A	Datei 1
[3] d:\root\B	Datei 2
[5] d:\root\C\CA	Datei 3
[6] d:\root\C\CB	Datei 4
[7] d:\root\C\CC	Datei C1

Bsp. 2: Schritt 2 (3. Durchlauf)



Name	Ordner +	Elementtyp
Root	D:\	Dateiordner
A	D:\Root\	Dateiordner
B	D:\Root\	Dateiordner
C	D:\Root\	Dateiordner
CA	D:\Root\C\	Dateiordner
CB	D:\Root\C\	Dateiordner
CC	D:\Root\C\	Dateiordner
CCA	D:\Root\C\CC\	Dateiordner
Datei CC1	D:\Root\C\CC\	Datei
Datei C1	D:\Root\C\	Datei

Nach Durchlauf 2	
Hash-Objekt	**SAS-Datei**
[2] d:\root\A	Datei 1
[3] d:\root\B	Datei 2
[5] d:\root\C\CA	Datei 3
[6] d:\root\C\CB	Datei 4
[7] d:\root\C\CC	Datei C1

Nach Durchlauf 3	
Hash-Objekt	**SAS-Datei**
[2] d:\root\A	Datei 1
[3] d:\root\B	Datei 2
[4] d:\root\C\CA	Datei 3
[5] d:\root\C\CB	Datei 4
[8] d:\root\C\CC\CCA	Datei C1
	Datei CC1

Bsp. 2: Schritt 2 (letzter Eintrag)

```
...
   /* 2. Verzeichnisanalyse: Arbeite Verzeichnisliste von hinten ab */
   stop = 0;
   DO WHILE (NOT stop);
      /* 2.1 Adressiere den letzten Verzeichniseintrag */
      h_dir_iter.LAST();
      remove_no_dir = no_dir;

      rc = FILENAME("indir", STRIP(path)!!"/");
      d_id = DOPEN("indir");
```

Bsp. 2: Schritt 2 (Member handling)

```
/* 2.2 Ermittle und behandle die Member aus dem Verzeichnis */
DO _j = 1 TO DNUM(d_id);
   member = DREAD(d_id, _j);
   member_path = STRIP(path)||"\"!!STRIP(member);

   /* Directory oder File ? */
   rc = FILENAME("infile", member_path);
   IF rc = 0 THEN DO;
      m_id = DOPEN("infile");
      /* 2.2.1 Füge Verzeichnisse dem Hash-Objekt hinzu */
      IF m_id > 0 THEN DO;    /* -> DIR */
         anz_dir + 1;
         h_dir.ADD(KEY: anz_dir, DATA: anz_dir, DATA: member_path);
         rc = DCLOSE(m_id);
      END;
      /* 2.2.1 Gebe Dateien in SAS-Datei aus */
      ELSE DO;
         m_id = FOPEN("infile");
         IF m_id > 0 THEN DO;       /* -> FILE */
            anz_file + 1;
            OUTPUT;
            rc = FCLOSE(m_id);
         END;
      END;
   END;
END;
rc = DCLOSE(d_id);
```

Bsp. 2: Schritt 2 (Schleifenende)

```
      /* 2.3 Lösche Eintrag aus Hash-Objekt oder setze stop */
      IF h_dir.NUM_ITEMS = 1 THEN stop = 1;
      ELSE DO;
         h_dir_iter.FIRST();
         IF no_dir = remove_no_dir THEN h_dir_iter.LAST();
         h_dir.REMOVE(KEY: remove_no_dir);
      END;
   END;

   PUT "Summary: #Files: " anz_file COMMAX. " #Dirs: " anz_dir COMMAX.;
RUN;
```

```
Summary: #Files:       25      #Dirs:        19
```

Programmerweiterungen:

- *Mehrere Startverzeichnisse (Laufwerke) und Exclude-Verzeichnisse*
- *Einlesen weiterer Dateiattribute (Size, Creation Date, Last Modified, …)*
- *Dateianalyse (potentielle Dubletten, Versionsstände, Backups …)*

Infos

- Dr. Arne Leißner
  Entimo AG
  arne.leissner@entimo.de

- KSFE 2015

  „Geschwindigkeit ist nicht alles – wozu Hash Objekte wirklich gut sind“

  - Proceedings der 19. Konferenz der SAS-Anwender in Forschung und Entwicklung (KSFE); S. 189 – 207
  - de.saswiki.org
    -> KSFE -> KSFE 2015 -> Performance