entimo

Geschwindigkeit ist nicht alles

–

wozu Hash-Objekte wirklich gut sind

Arne Leißner

19. KSFE 2015 in Hannover

Hash-Objekt und PDV

```
DATA lab_pat;
  SET lab;
  IF _N_ = 1 THEN DO;
    IF flag THEN dsn = "patinfo_enh";
    ELSE dsn = "patinfo";
    DCL HASH h (DATASET: dsn);

    h.DEFINEKEY("subjid");

    h.DEFINEDATA("age", "sex");
    IF flag THEN h.DEFINEDATA("weight", "height");

    h.DEFINEDONE();
  END; /* _N_ = 1 */
...
  IF  measure > 75 THEN rc = h.FIND();
...
RU
```

Bedingungsabhängiges Anlegen zur Execution Time

Ohne Bedingung: Bei jeder Iteration würde ein (weiteres) Hash-Objekt angelegt und gefüllt werden …

Datenabhängiges Festlegen der Datenquelle zur Execution Time

Minimum: Eine Schlüsselspalte

Schrittweises und datenabhängiges Festlegen der Struktur

Gefülltes Hash-Objekt beinhaltet:
- 1 Schlüsselspalte
- 2 oder 4 Datenspalten
- # Einträge wie # Sätze in Quelldatei

Auch wenn bei _N_ = 1 angelegt: Hash-Objekt in jeder Iteration referenzierbar

Lookup: Key Value aus aktueller Beobachtung verwendet; Daten automatisch in PDV übertragen, wenn subjid gefunden

Hash-Objekt: Benefits

- Zusätzliche dynamische Datenstruktur(en) im Direktzugriff des 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)
- Freie Navigation (via Schlüssel od. sequentiell)
  -> Direktadressierung
  -> Multiadressierung
  -> schrittweise vorwärts / rückwärts mittels Iterator

Hash-Objekt: Benefits

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

Hash-Objekt: Techniken & Use Cases

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

Sortierung

Aufgabe: Sortiere sashelp.class na

Features: attribute-tags DATASET
+ OUTPUT Methode

```
DATA _null_;
   IF 0 THEN SET sashelp.class;

   DCL HASH h(DATASET: "sashelp.class",
              ORDERED: "A", MULTIDATA: "Y");
   h.DEFINEKEY("age", "sex");
   h.DEFINEDATA(ALL: "Y");
   h.DEFINEDONE();

   h.OUTPUT(DATASET "work.class_sorted");
RUN;

NOTE: There were 19 observations read from th
NOTE: The data set WORK.CLASS_SORTED has 19 o
```

class_sorted

Beob.	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Thomas	M	11	57.5	85.0
3	Jane	F	12	59.8	84.5
4	Louise	F	12	56.3	77.0
5	James	M	12	57.3	83.0
6	John	M	12	59.0	99.5
7	Robert	M	12	64.8	128.0
8	Alice	F	13	56.5	84.0
9	Barbara	F	13	65.3	98.0
10	Jeffrey	M	13	62.5	84.0
11	Carol	F	14	62.8	102.5
12	Judy	F	14	64.3	90.0
13	Alfred	M	14	69.0	112.5
14	Henry	M	14	63.5	102.5
15	Janet	F	15	62.5	112.5
16	Mary	F	15	66.5	112.0
17	Ronald	M	15	67.0	133.0
18	William	M	15	66.5	112.0
19	Philip	M	16	72.0	150.0

De-Duplizierung (1)

Aufgabe: Reduziere sashelp.class
Alter und Geschlecht

Features: attribute-tag MULTIDAT

class_dedup

Beob.	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Thomas	M	11	57.5	85.0
3	Jane	F	12	59.8	84.5
4	James	M	12	57.3	83.0
5	[illegible]	F	13	56.5	84.0
6	Jeffrey	M	13	62.5	84.0
7	Carol	F	14	62.8	102.5
8	Alfred	M	14	69.0	112.5
9	Janet	F	15	62.5	112.5
10	Ronald	M	15	67.0	133.0
11	Philip	M	16	72.0	150.0

```
DATA _null_;
   IF 0 THEN SET sashelp.class;

   DCL HASH h(DATASET: "sashelp.class",
              ORDERED: "A", MULTIDATA: "N");
   h.DEFINEKEY("age", "sex");
   h.DEFINEDATA(ALL: "Y");
   h.DEFINEDONE();

   h.OUTPUT(DATASET "work.class_sorted");
RUN;

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.CLASS_SORTED has 11 observations and 5 variables.
```

De-Duplizierung (2)

Aufgabe: Reduziere sashelp.class
Alter und Geschlecht

Features: attribute-tags MULTIDA

```
DATA _null_;
   IF 0 THEN SET sashelp.class;

   DCL HASH h(DATASET: "sashelp.class",
              ORDERED: "A", DUPLICATE: "R", M
   h.DEFINEKEY("age", "sex");
   h.DEFINEDATA(ALL: "Y");
   h.DEFINEDONE();

   h.OUTPUT(DATASET "work.class_sorted");
RUN;

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.CLASS_SORTED has 11 observations and 5 variables.
```

class_dedup

Beob.	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Thomas	M	11	57.5	85.0
3	Louise	F	12	56.3	77.0
4	Robert	M	12	64.8	128.0
5	Barbara	F	13	65.3	98.0
6	Jeffrey	M	13	62.5	84.0
7	Judy	F	14	64.3	90.0
8	Henry	M	14	63.5	102.5
9	Mary	F	15	66.5	112.0
10	William	M	15	66.5	112.0
11	Philip	M	16	72.0	150.0

Zählung

Aufgabe: Zähle die Sätze
Kombination au

Features: attribute-tag SU

```
satz=1    Age=14 Sex=M count=1
satz=2    Age=13 Sex=F count=1
satz=3    Age=13 Sex=F count=2
satz=4    Age=14 Sex=F count=1
satz=5    Age=14 Sex=M count=2
satz=6    Age=12 Sex=M count=1
satz=7    Age=12 Sex=F count=1
satz=8    Age=15 Sex=F count=1
satz=9    Age=13 Sex=M count=1
satz=10   Age=12 Sex=M count=2
satz=11   Age=11 Sex=F count=1
satz=12   Age=14 Sex=F count=2
satz=13   Age=12 Sex=F count=2
satz=14   Age=15 Sex=F count=2
satz=15   Age=16 Sex=M count=1
satz=16   Age=12 Sex=M count=3
satz=17   Age=15 Sex=M count=1
satz=18   Age=11 Sex=M count=1
satz=19   Age=15 Sex=M count=2
```

```
DATA _null_;
   DCL HASH h(SUMINC: "increment");
   h.DEFINEKEY("age", "sex");
   h.DEFINEDONE();           /* Kein D

   increment = 1;
   DO UNTIL (eof);
      SET sashelp.class(keep = age sex) END = eof;
      rc = h.REF();          /* REF() = CHECK() + ADD() */
      rc = h.SUM(SUM: count);
      satz + 1;
      PUT @5 satz= @15 age= sex= count=;
   END;
RUN;

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
```

work.monatsumsatz (obs = 16)

Beob.	produkt	jahr	monat	umsatz
1	A	2010	1	€1.644,7
2	B	2010	1	€486,2
3	C	2010	1	€941,9
4	A	2010	2	€1.809,4
5	B	2010	2	€487,5
6	C	2010	2	€994,9
7	A	2010	3	€1.762,7
8	B	2010	3	€493,0
9	C	2010	3	€1.073,5
10	A	2010	4	€1.773,3
11	B	2010	4	€542,2
12	C	2010	4	€1.047,1
13	A	2010	5	€1.914,5
14	B	2010	5	€567,8
15	C	2010	5	€1.129,1
16	A	2010	6	€1.984,4

Produkt den Ja
ätzen
KEYSUM + FI

```
tz", KEYSUM: "ums
jahr");
"jahr");

= jahr produkt u
_sum.ADD();

sumsatz");

ns read from the
MSATZ has 15 obse
```

work.jahresumsatz

Beob.	produkt	jahr	umsatz
1	A	2010	€269.433,5
2	A	2011	€282.708,8
3	A	2012	€300.262,3
4	A	2013	€308.316,1
5	A	2014	€325.509,2
6	B	2010	€62.823,6
7	B	2011	€65.066,0
8	B	2012	€67.258,4
9	B	2013	€70.900,6
10	B	2014	€73.997,5
11	C	2010	€155.348,9
12	C	2011	€164.830,6
13	C	2012	€172.118,2
14	C	2013	€176.715,6
15	C	2014	€190.105,5

Data Set Split, Durchwanderung, Mehrfachlesen

Aufgabe: Erzeuge aus sashelp.class je Alter und Geschlecht eine neue SAS-Datei ohne vorher Zahl und Inhalt der Kombinationen zu kennen

Features:
- Mehrere Hash-Objekte
- Mehrfachlesen einer Datei
- Hash-Iterator
- FIRST, NEXT und OUTPUT Methode

Idee:
- ➔ 1. Hash-Objekt: Datei komplett einlesen
- ➔ 2. Hash-Objekt: Nur die Schlüssel einlesen
- ➔ 2. Hash Objekt mit Iterator versehen
- ➔ 2. Hashobjekt durchwandern und je Schlüsselkombination Ausgabedatei aus dem 1. Hash-Objekt schreiben

Data Set Split, Durchwanderung, Mehrfachlesen

```
DATA _null_;
   IF 0 THEN SET sashelp.class;

   DCL HASH h(DATASET: "sashelp.class", MULTIDATA: "Y");
   h.DEFINEKEY("age", "sex");
   h.DEFINEDATA(ALL: "Y");
   h.DEFINEDONE();

   DCL HASH h_key(DATASET: "sashelp.class(keep = age sex)");
   h_key.DEFINEKEY("age", "sex");
   h_key.DEFINEDONE();

   DCL HITER h_key_iter("h_key");  /* Iterator */

   rc = h_key_iter.FIRST();
   DO UNTIL(h_key_iter.NEXT());
      h.OUTPUT(DATASET: "class_"!!strip(sex)!!"_"!!strip(put(age, 3.)) !!
                        "(WHERE = (sex='"!!strip(sex)!!"' and age=" !!
                        strip(put(age, 3.))!!"))");
   END;
RUN;

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.CLASS_F_11 has 1 observations and 5 variables.
NOTE: The data set WORK.CLASS_M_11 has 1 observations and 5 variables.
NOTE: The data set WORK.CLASS_F_12 has 2 observations and 5 variables
...
```

age, sex + MULTIDATA

... ohne MULTIDATA

Key-Iterator

FIRST
NEXT
OUTPUT

Fuzzy-Merge

work.patients (obs = 15)

Beob.	subjid	age	sex	height	weight
1	PAT_000001	86	F	168	66.0
2	PAT_000002	59	F	168	85.0
3	PAT_000003	86	F	168	55.0
4	PAT_000004	53	M	187	80.1
5	PAT_000005	87	F	173	62.0
6	PAT_000006	86	M	180	82.8
7	PAT_000007	91	F	150	78.0
8	PAT_000008	86	F	180	50.0
9	PAT_000009	40	M	184	50.9
10	PAT_000010	52	M	170	96.3
11	PAT_000011	32	M	186	76.5
12	PAT_000012	58	F	183	65.0
13	PAT_000013	32	M	173	58.8
14	PAT_000014	30	F	161	89.0
15	PAT_000015	76	F	174	55.0

-Werte in Kategorien (Intervalle) ein

-Objekte

egorien (obere Grenze ist Schlüssel)

daten ergänzt um BMI-Infos

or für K

XT, RE

-Objekt:

nt

tz lesen

MI ermit

Hash-O

Hash O

work.bmi_kategorie

Beob.	Kategorie	bis_bmi
1	0.1 - Starkes Untergewicht (< 16)	16.0
2	0.2 - Mäßiges Untergewicht (16 - < 17)	17.0
3	0.3 - Leichtes Untergewicht (17 - < 18,5)	18.5
4	1 - Normalgewicht (18,5 - < 25)	25.0
5	2 - Übergewicht (25 -< 30)	30.0
6	3.1 - Fettleibigkeit Grad I (30 - < 35)	35.0
7	3.2 - Fettleibigkeit Grad II (35 - < 40)	40.0
8	3.3 - Fettleibigkeit Grad III (>= 40)	999.0

Fuzzy-Merge

work.patients (obs = 15)

Beob.	subjid	age	sex	height	weight
1	PAT_000001	86	F	168	66.0
2	PAT_000002	59	F	168	85.0
3	PAT_000003	86	F	168	55.0
4	PAT_000004	53	M	187	80.1
5	PAT_000005	87	F	173	62.0
6	PAT_000006	86	M	180	82.8
7	PAT_000007	91	F	150	78.0
8	PAT_000008	86	F	180	50.0
9	PAT_000009	40	M	184	50.9
10	PAT_000010	52	M	170	96.3
11	PAT_000011	32	M	186	76.5
12	PAT_000012	58	F	183	65.0
13	PAT_000013	32	M	173	58.8
14	PAT_000014	30	F	161	89.0
15	PAT_000015	76	F	174	55.0

work.patients_bmi_cat (obs = 15)

Beob.	subjid	sex	age	bmi	Kategorie
1	PAT_000188	F	49	13.2	0.1 - Starkes Untergewicht (< 16)
2	PAT_000471	M	53	38.5	3.2 - Fettleibigkeit Grad II (35 - < 40)
3	PAT_000544	F	96	18.9	1 - Normalgewicht (18,5 - < 25)
4	PAT_000617	M	81	33.5	3.1 - Fettleibigkeit Grad I (30 - < 35)
5	PAT_000900	M	21	31.0	3.1 - Fettleibigkeit Grad I (30 - < 35)
6	PAT_001273	F	32	15.6	0.1 - Starkes Untergewicht (< 16)
7	PAT_001346	F	77	30.1	3.1 - Fettleibigkeit Grad I (30 - < 35)
8	PAT_001419	F	81	26.8	2 - Übergewicht (25 -< 30)
9	PAT_001702	F	67	28.1	2 - Übergewicht (25 -< 30)
10	PAT_001999	M	50	23.0	1 - Normalgewicht (18,5 - < 25)
11	PAT_002075	F	54	14.3	0.1 - Starkes Untergewicht (< 16)
12	PAT_002148	M	54	27.5	2 - Übergewicht (25 -< 30)
13	PAT_002431	F	98	33.8	3.1 - Fettleibigkeit Grad I (30 - < 35)
14	PAT_002504	F	40	32.8	3.1 - Fettleibigkeit Grad I (30 - < 35)
15	PAT_003160	F	81	21.8	1 - Normalgewicht (18,5 - < 25)

TOP-<n>%

Aufgabe: Die 5% der Patienten mit den höchsten BMI ermitteln

Features: attribute-tag ORDERED
Attribut NUM_ITEMS
Methoden FIRST, LAST, NEXT, PREV,
DELETE, REMOVE

Idee:

- ➔ Vorherigen Data Step erweitern
- ➔ 3. Hash-Objekt (absteigend nach BMI)
- ➔ Umkopieren aller Patienten und BMI-Infos auf das neue Hash-Objekt
- ➔ Von hinten (mit dem kleinsten BMI beginnend) 95% der Einträge entfernen

TOP-<n>%

```
...
DCL HASH h_top (ORDERED:
h_top.DEFINEKEY("bmi", "s
h_top.DEFINEDATA("bmi", "
h_top.DEFINEDONE();

DCL HITER h_iter("h");
DCL HITER h_top_iter("h_t

anz = h.NUM_ITEMS;
h_iter.FIRST();
DO i = 1 TO anz;
   rc = h_top.ADD();
   rc = h_iter.NEXT();
END;
rc = h.DELETE(); rc = h_i

h_top_iter.LAST();
DO i = 1 TO INT(anz * 0.9
   bmi_del = bmi; subjid_
   rc = h_top_iter.PREV()
   rc = h_top.REMOVE(key:
END;
h_top.OUTPUT(DATASET: "pa
RUN;

NOTE: The data set WORK.PATIENTS_BMI_TOP5PCT has 2500 observations ...
```

work.patients_bmi_top5pct (obs = 15)

Beob.	bmi	subjid	age	sex	Kategorie
1	42.2	PAT_035625	56	M	3.3 - Fettleibigkeit Grad III (>= 40)
2	42.2	PAT_025228	29	M	3.3 - Fettleibigkeit Grad III (>= 40)
3	42.2	PAT_004461	32	M	3.3 - Fettleibigkeit Grad III (>= 40)
4	42.1	PAT_010372	37	M	3.3 - Fettleibigkeit Grad III (>= 40)
5	42.0	PAT_047896	89	M	3.3 - Fettleibigkeit Grad III (>= 40)
6	41.9	PAT_046186	27	M	3.3 - Fettleibigkeit Grad III (>= 40)
7	41.9	PAT_036580	89	M	3.3 - Fettleibigkeit Grad III (>= 40)
8	41.9	PAT_034605	57	M	3.3 - Fettleibigkeit Grad III (>= 40)
9	41.9	PAT_003347	34	M	3.3 - Fettleibigkeit Grad III (>= 40)
10	41.7	PAT_017262	95	M	3.3 - Fettleibigkeit Grad III (>= 40)
11	41.7	PAT_014023	70	M	3.3 - Fettleibigkeit Grad III (>= 40)
12	41.6	PAT_024777	55	M	3.3 - Fettleibigkeit Grad III (>= 40)
13	41.6	PAT_007975	37	M	3.3 - Fettleibigkeit Grad III (>= 40)
14	41.5	PAT_028157	93	M	3.3 - Fettleibigkeit Grad III (>= 40)
15	41.5	PAT_014799	19	M	3.3 - Fettleibigkeit Grad III (>= 40)

Gruppenstatistiken

Aufgabe: Aus den TOP-5% je Geschlecht die Zahl der Patienten und deren BMI-Durchschnitt bestimmen

Features: attribute-tag MULTIDATA + DO_OVER Methode

Idee:
- ➔ Vorherigen Data Step erweitern
- ➔ 4. Hash-Objekt (KEY: sex)
- ➔ Multidata DO_OVER-Verarbeitung je KEY-Value

Gruppenstatistiken

```
...
DCL HASH h_top_sex (MULTIDATA: "Y");
h_top_sex.DEFINEKEY("sex");
h_top_sex.DEFINEDATA("sex", "bmi");
h_top_sex.DEFINEDONE();

h_top_iter.FIRST();
DO i = 1 to h_top.NUM_ITEMS;                    /* Umkopieren */
   rc = h_top_sex.ADD();
   rc = h_top_iter.NEXT();
END;

DO sex = "F", "M";
   anz_subj = 0;
   summe_bmi = 0;
   DO WHILE (h_top_sex.DO_OVER(KEY: sex) = 0);    /* Alle mit gleichem KEY */
               anz_subj = anz_subj + 1;
               summe_bmi = summe_bmi + bmi;
   END;
   mean_bmi = summe_bmi / anz_subj;
   PUT "***** " sex= anz_subj= COMMAX5. @28 mean_bmi= 4.1;
END;
RUN;

***** sex=F anz_subj=349   mean_bmi=37.8
***** sex=M anz_subj=2.151 mean_bmi=38.4
```

sex + MULTIDATA

DO_OVER KEY:

Hash-Objekt: Résumé

- Beispielaufgaben auch auf anderem Wege lösbar - erfordert ggf. zusätzliche Verarbeitungsschritte
- Erweitert die Möglichkeiten des DATA Steps
- Insbesondere interessant, wenn die Techniken in einem DATA Step kombiniert werden sollen
- Relativ schlanke Programmierung (Definitionsteil manchmal länger als Logikteil)
- Key-bezogene und MULTIDATA-bezogene Methoden -> Aufpassen !
- Handling teilweise etwas „sperrig“
- *Aber: Anschauen lohnt sich (nicht nur wegen der Geschwindigkeit)*