UNIQA

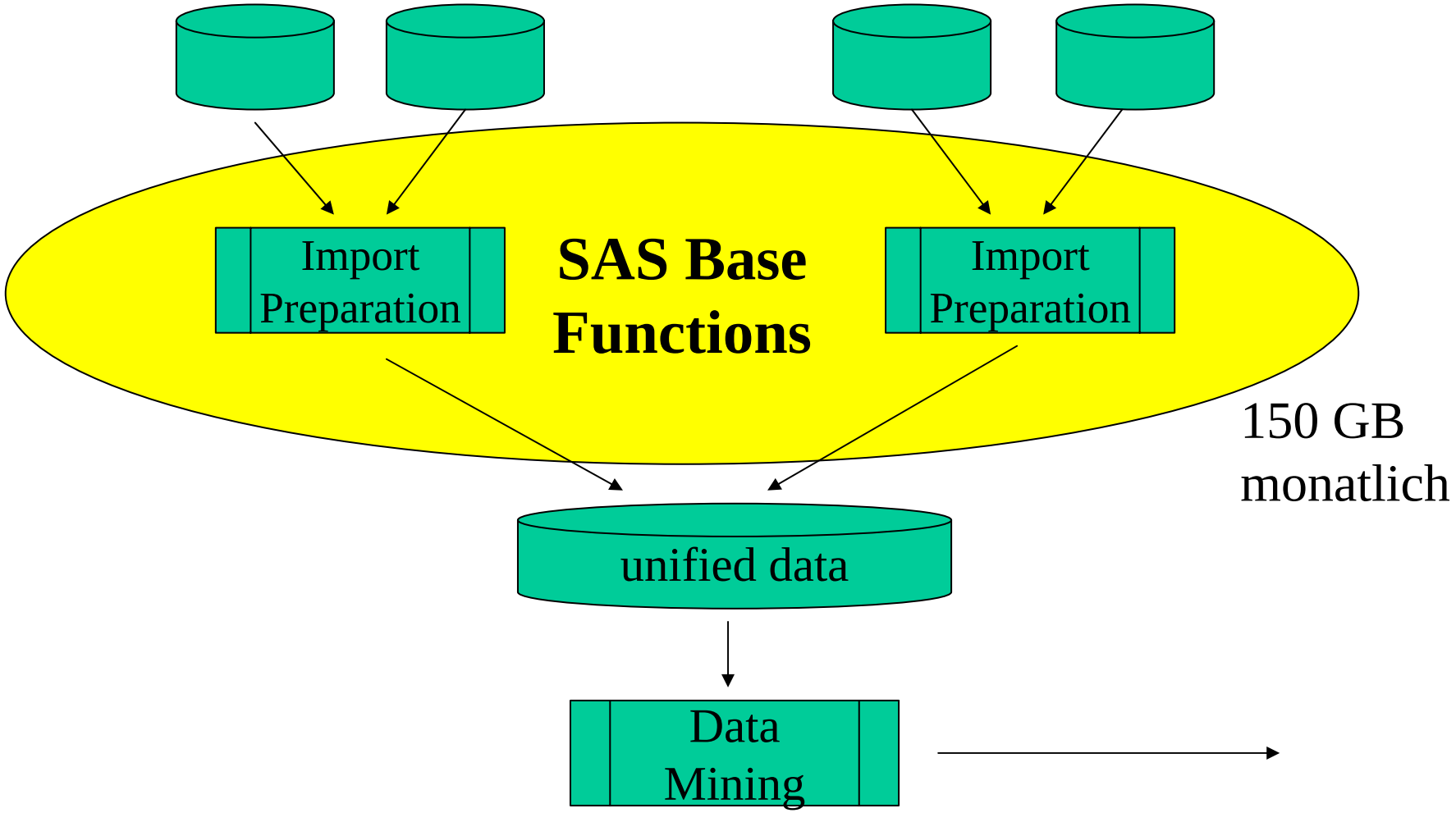
SAS Implementierungs-Erfahrungen beim Aufbau eines zentralen Kunden Data Marts für Data Mining

SAS Club, 22.11.2007

Helmut Zehetmayr

UNIQA Versicherungen AG

Datenszenario



UNIQA

Themenübersicht

- Optimierung SAS Programme
  - reduzieren Plattenspeicher
  - Laufzeitoptimierung
- nützliche Macros
  - LOGSEP/OUTSEP
  - Runstart
  - Runwait
- nützliche SAS Funktionen
  - SORT Optionen
  - ermitteln der Anzahl von Beobachtungen einer SAS Datei
  - SAS AutoCall

Optimierung SAS Programme

- reduzieren Plattenspeicher
  - Datenkomprimierung
- Laufzeitoptimierung
  - schlanke Dateien
  - INDEX oder SORT

UNIQA

reduzieren Plattenspeicher

- COMPRESS
  - alphanumerische Felder
  - **Option Compress=YES/NO;**
  - **Data Test (Compress=YES/NO);**
  - **Format Field $30.; Field='abcdef';**
    30 Bytes unkomprimiert
    8 Bytes komprimiert (6 Bytes Daten, 2 Bytes Längenfeld)

UNIQA

reduzieren Plattenspeicher

- COMPRESS
  - nicht empfehlenswert für kurze (unter $5.) oder für komplett ausgefüllte Datenenfelder
  - reduziert Platzbedarf aber erhöht CPU Aktivität

UNIQA

reduzieren Plattenspeicher

- COMPRESS
  - Report in LOG

```
Compressing data set WORK.TEST1 decreased size by 66.19
percent.
Compressed is 3624 pages; un-compressed would require
10720 pages.

Compressing data set WORK.TEST3 increased size by 40.00
percent.
Compressed is 7 pages; un-compressed would require 5
pages.

Compression was disabled for data set WORK.TEMP because
compression overhead would increase the size of the data
set.
```

reduzieren Plattenspeicher

- Format/Length Statements
  - numerische Felder
  - **Format Field 5.;** 8 Bytes
    Length Field 4; reduziert auf 4 Bytes
  - nicht empfehlenswert für Daten mit Dezimalstellen (Genauigkeitsverlust)

```
length      max. value
3                8.192
4            2.097.152
5          536.870.912
6      137.438.953.472
7   35.184.372.088.832
```

reduzieren Plattenspeicher

- YES/NO Daten alphanumerisch
  - numerische Felder (1/0) benötigen mindestens 3 Bytes
  - alphanumerisch ('1'/'0') nur 1 Byte
  - Achtung auf Datenkompression
    möglicherweise werden 3 Bytes benötigt
  - mehr "Schreibarbeit"

Laufzeitoptimierung

- schlanke Tabellen (nur wirklich benötigte Daten)
  - **Proc SORT**
    Data=Input (keep/drop=.....
    where=(.....))
    Out=Output; ← **sehr wichtig**

  - **Data Output (keep/drop=...);**
    Merge/Set Input (keep/drop=...);

UNIQA

Laufzeitoptimierung

- INDEX oder SORT
  - häufigste Zugriffsreihenfolge
    Ladesequenz
  - Vorselektion
    Index oder SORT mit where=
  - Datenabfrage (view table)
    Index

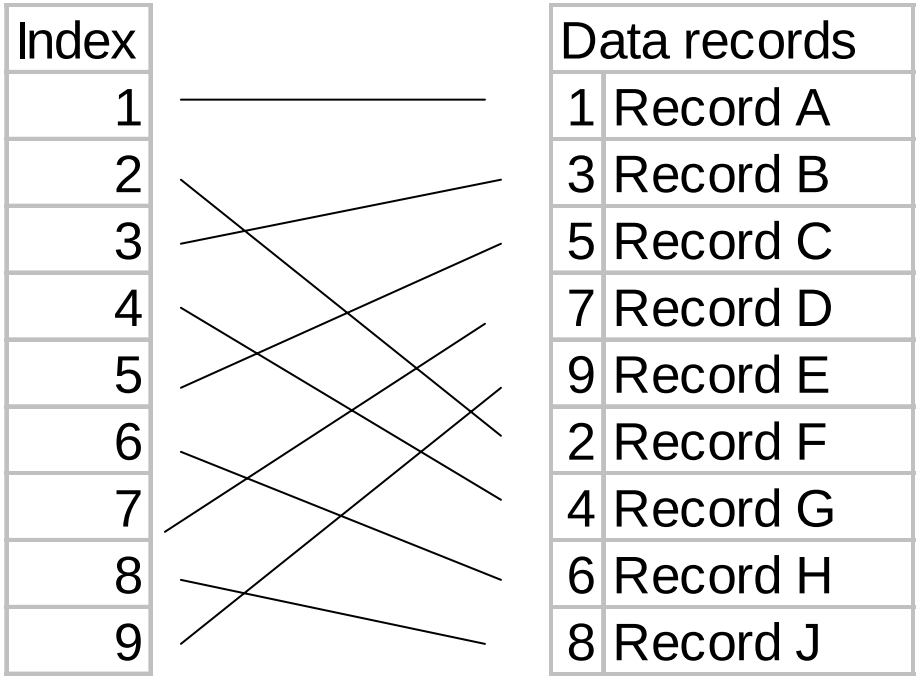
Laufzeitoptimierung

- Arbeitsweise Index-sequentieller Zugriff
  - Ladereihenfolge alphanumerisch

Data records	
1	Record A
3	Record B
5	Record C
7	Record D
9	Record E
2	Record F
4	Record G
6	Record H
8	Record J

Laufzeitoptimierung

- Arbeitsweise Index-sequentieller Zugriff
  - Aufbau des Index



Laufzeitoptimierung

- Arbeitsweise Index-sequentieller Zugriff
  - lesen erste Datenzeile (ID = 1)

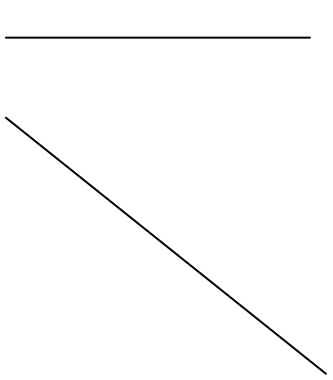
Index
1
2
3
4
5
6
7
8
9

Data records	
1	Record A
3	Record B
5	Record C
7	Record D
9	Record E
2	Record F
4	Record G
6	Record H
8	Record J

Laufzeitoptimierung

- Arbeitsweise Index-sequentieller Zugriff
  - weiterlesen (ID=2, sechste Datenzeile)

Index
1
2
3
4
5
6
7
8
9

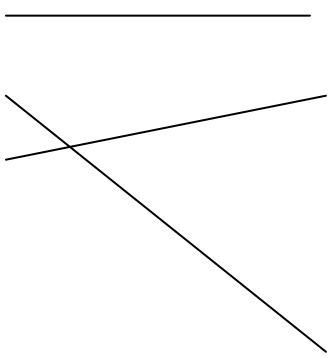


Data records	
1	Record A
3	Record B
5	Record C
7	Record D
9	Record E
2	Record F
4	Record G
6	Record H
8	Record J

Laufzeitoptimierung

- Arbeitsweise Index-sequentieller Zugriff
  - weiterlesen (ID=3, zweite Datenzeile)

Index
1
2
3
4
5
6
7
8
9

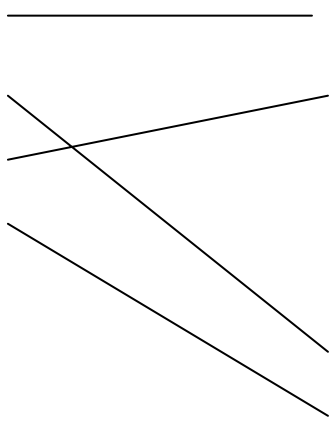


Data records	
1	Record A
3	Record B
5	Record C
7	Record D
9	Record E
2	Record F
4	Record G
6	Record H
8	Record J

Laufzeitoptimierung

- Arbeitsweise Index-sequentieller Zugriff
  - "butterfly" reading

Index
1
2
3
4
5
6
7
8
9

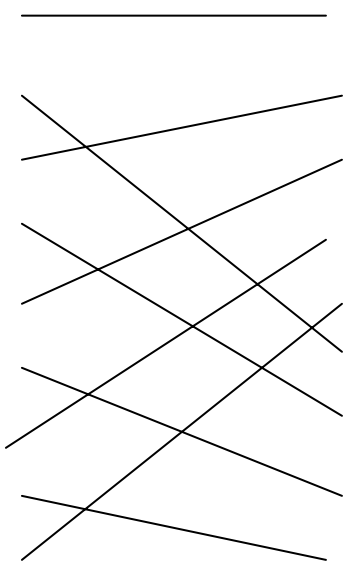


Data records	
1	Record A
3	Record B
5	Record C
7	Record D
9	Record E
2	Record F
4	Record G
6	Record H
8	Record J

Laufzeitoptimierung

- Arbeitsweise Index-sequentieller Zugriff
  - "butterfly" reading

Index
1
2
3
4
5
6
7
8
9



Data records	
1	Record A
3	Record B
5	Record C
7	Record D
9	Record E
2	Record F
4	Record G
6	Record H
8	Record J

UNIQA

nützliche Macros

- LOGSEP/OUTSEP
  - umschalten LOG-Ausgabe vom SAS-Fenster in Textdatei und retour
- RUNSTART
  - Beginn/Fortsetzung eines Programmes zu einer bestimmten Uhrzeit
- RUNWAIT
  - Unterbrechung der Durchführung für eine bestimmte Zeit

UNIQA

LOGSEP/OUTSEP

- umschalten LOG-Ausgabe vom SAS-Fenster in Textdatei
  - **%LOGSEP(C:\SASLog\ProgName);**
    %OUTSEP(C:\SASLog\ProgName);
    C:\SASLog\ProgName_yyyymmdd_hhmmss.ttt.TXT
  - keine Grössenbeschränkung
  - während der Laufzeit Kontrolle mit Texteditor
    Achtung: MS-Word nicht verwenden
  - weitere %LOGSEP/OUTSEP-Aufrufe erzeugen neue Dateien
  - Analyse der Textdateien bei Bedarf

LOGSEP/OUTSEP

- Zurückschalten zum SAS LOG-Fenster
  - **%LOGSEP();**
    %OUTSEP();

- jede LOG-Datei beinhaltet
  - Beginn LOG-Datei

```
------------------------------------------------------------------------------
Begin of this LOG at 09MAY2007 10:15:01
------------------------------------------------------------------------------
```

  - Ende LOG-Datei

```
------------------------------------------------------------------------------
End of this LOG at 10MAY2007 10:26:34
------------------------------------------------------------------------------
Further LOG messages will be written to C:\TEMP\LOG1_20070510_102634.289.TXT
------------------------------------------------------------------------------
```

Runstart

- **`%Runstart(hh:mm);`**

- SAS Sitzung wird für 60 Sekunden unterbrochen, bis die angegebene Zeit erreicht ist

- kann bis zu 23 h 59 min dauern

Runwait

- **`%Runwait(hh:mm);`**
  - SAS Sitzung wird für die angegebenen Stunden:Minuten unterbrochen
- **`%Runwait(sss);`**
  - SAS Sitzung wird für die angegebene Sekundenanzahl unterbrochen

UNIQA

Themenübersicht

- Optimierung SAS Programme
  - reduzieren Plattenspeicher
  - Laufzeitoptimierung
- nützliche Macros
  - LOGSEP/OUTSEP
  - Runstart
  - Runwait
- nützliche SAS Funktionen
  - SORT Optionen
  - ermitteln der Anzahl von Beobachtungen einer SAS Datei
  - SAS AutoCall

nützliche SAS Funktionen

- SORT Optionen
  - NODUPKEYS/NODUPRECS
  - Keep/Drop/Where
- Anzahl der Datensätze in einer SAS-Datei
  - zählen
  - CONTENTS procedure
  - SQL
- SAS AutoCall
  - Arbeitsweise
  - Aktivierung
  - Einschränkungen

SORT Optionen

- NODUPKEYS/NODUPRECS

field1	field2	field3
1	2	3
1	2	4
1	2	5
1	2	4
1	2	3
1	2	2

SORT Optionen

- NODUPKEYS/NODUPRECS

```
Proc SORT Data=test Out=test1 nodupkeys;
by field1 field2;
run;
```

field1	field2	field3
1	2	3
1	2	4
1	2	5
1	2	4
1	2	3
1	2	2

field1	field2	field3
1	2	3

SORT Optionen

- NODUPKEYS/NODUPRECS

```
Proc SORT Data=test Out=test2 noduprecs;
by field1 field2;
run;
```

field1	field2	field3
1	2	3
1	2	4
1	2	5
1	2	4
1	2	3
1	2	2

field1	field2	field3
1	2	3
1	2	4
1	2	5
1	2	4
1	2	3
1	2	2

UNIQA

SORT Optionen

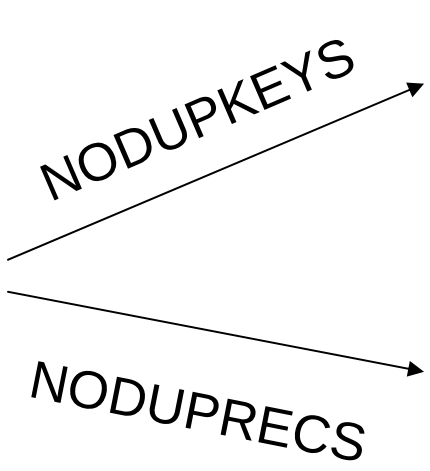
- NODUPKEYS/NODUPRECS
  mit vorsortierten Daten

field1	field2	field3
1	2	2
1	2	3
1	2	3
1	2	4
1	2	4
1	2	5

SORT Optionen

- NODUPKEYS/NODUPRECS mit vorsortierten Daten

field1	field2	field3
1	2	2
1	2	3
1	2	3
1	2	4
1	2	4
1	2	5



field1	field2	field3
1	2	2

field1	field2	field3
1	2	2
1	2	3
1	2	4
1	2	5

SORT Optionen

- Keep/Drop/Where
  - **OUT=** Option nicht vergessen!

  - ```
    Proc SORT Data=test (keep=.....
                         where=(...))
              Out=test1 (drop=.....)
    ```

  - Sortiervariablen mussen in **keep=** vorhanden sein, dürfen in **drop=** aber nicht angegeben werden
    ```

UNIQA Anzahl der Datensätze einer SAS-Datei

- Zählung durch Lesen der Datei

```
Data _null_;
Set WORK.Input1 end=EOF;
recs + 1;
if EOF then
   Call SYMPUT('Input1_Obs',
                TRIM(LEFT(recs)));
Run;
```

lange Laufzeit für grosse Dateien
nicht empfehlenswert

UNIQA Anzahl der Datensätze einer SAS-Datei

- Zählung beim MERGE mehrerer Dateien

```
Data _null_;
Merge WORK.Input1 (in=a)
      WORK.Input2 (in=b)
      end=EOF;
by Field1 Field2;
if a then recsa + 1;
if b then recsb + 1;
if EOF then do;
   Call SYMPUT('Input1_Obs',
               TRIM(LEFT(recsa)));
   Call SYMPUT('Input2_Obs',
               TRIM(LEFT(recsb)));

   end;
Run;
```

liefert oft falsche Ergebnisse

UNIQA Anzahl der Datensätze einer SAS-Datei

- CONTENTS procedure

```
Proc CONTENTS Data=WORK._ALL_ Out=Temp noprint;
Run;
Data _null_;
Set Temp;
by MemName;
if FIRST.MemName then
   Call SYMPUT(TRIM(MemName) || '_Obs',
               TRIM(LEFT(NObs)));
Run;
```

schnellste Methode für mehrere Dateien

Anzahl der Datensätze einer SAS-Datei

- SQL

```
%Let Input1_Obs = 0;
Proc SQL noprint;
Select COUNT(*) into :Input1_Obs
from WORK.Input1;
%Let Input1_Obs = &Input1_Obs;
Quit;
```

schnellste Mathode für eine einzelne Datei

UNIQA

SAS AutoCall

- Arbeitsweise
  - Sourcecode für Macros in unterschiedlichen Verzeichnissen
  - Vorkompilierung nicht notwendig automatisch beim erstmaligen Aufruf
  - vorgegebene Verzeichnisse werden in der Kompilierungsphase durchsucht
  - kompilierte Version in WORK.SASMACR

UNIQA

SAS AutoCall

- Aktivierung

```
Options MAUTOSOURCE
SASAUTOS=(
"C:\Data\SAS\Macros\AllUsers"
"C:\Data\SAS\Macros\International"
"C:\Data\SAS\Macros\International\BasisDB"
SASAUTOS);
```

- Angabe in **AUTOEXEC**

UNIQA

SAS AutoCall

- Einschränkungen
  - Makro- und Sourcecodename müssen identisch sein

    %Macro Test(P1); .....

    muss **TEST.SAS** heissen

  - Änderungen des Makros werden **nicht** automatisch nachgezogen, daher manuelles Rekompilieren notwendig

Themenübersicht

- Optimierung SAS Programme
  - reduzieren Plattenspeicher
  - Laufzeitoptimierung
- nützliche Macros
  - LOGSEP/OUTSEP
  - Runstart
  - Runwait
- nützliche SAS Funktionen
  - SORT Optionen
  - ermitteln der Anzahl von Beobachtungen einer SAS Datei
  - SAS AutoCall

→ keine weiteren Themen vorgesehen ☺