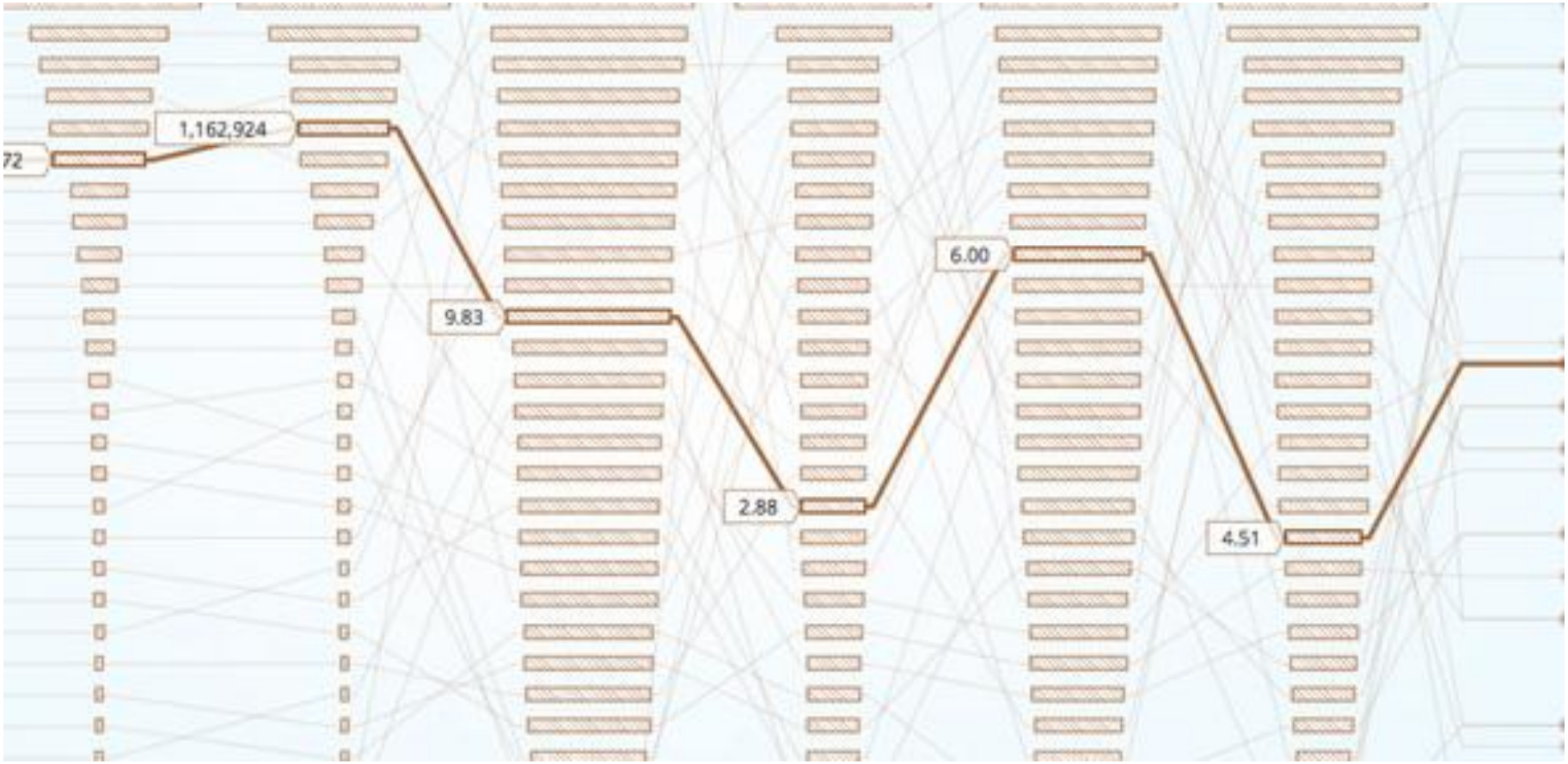
# Tips & Tricks for seamless integration with web applications and file system

Alexander Khorunzhiy
Stein Arve Finnestad
Nithya Mohan

sas

+ Web Applications

= ?


72
1,162,924
9.83
2.88
6.00
4.51

# Challenge

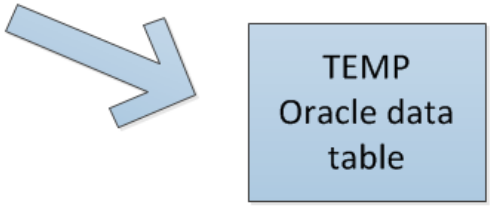
- Oracle table with 40 000 000 of rows should be updated by SAS with 200 000 rows
- all the manuals & whitepapers regarding SAS 9.2 are studied

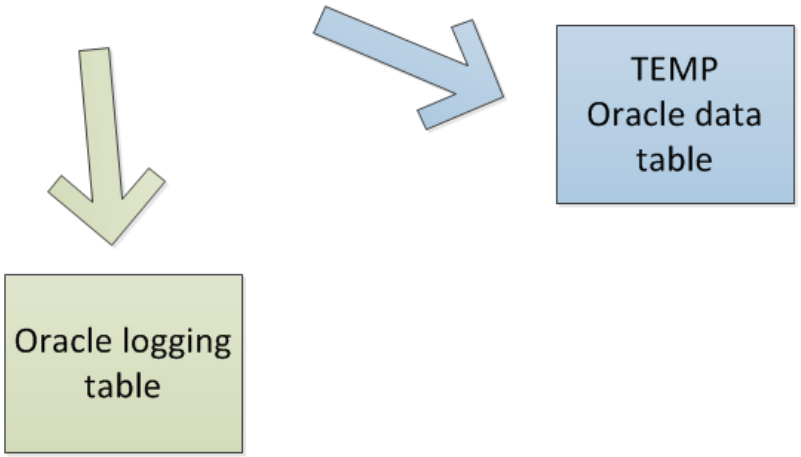
# 1st approach which doesn't work

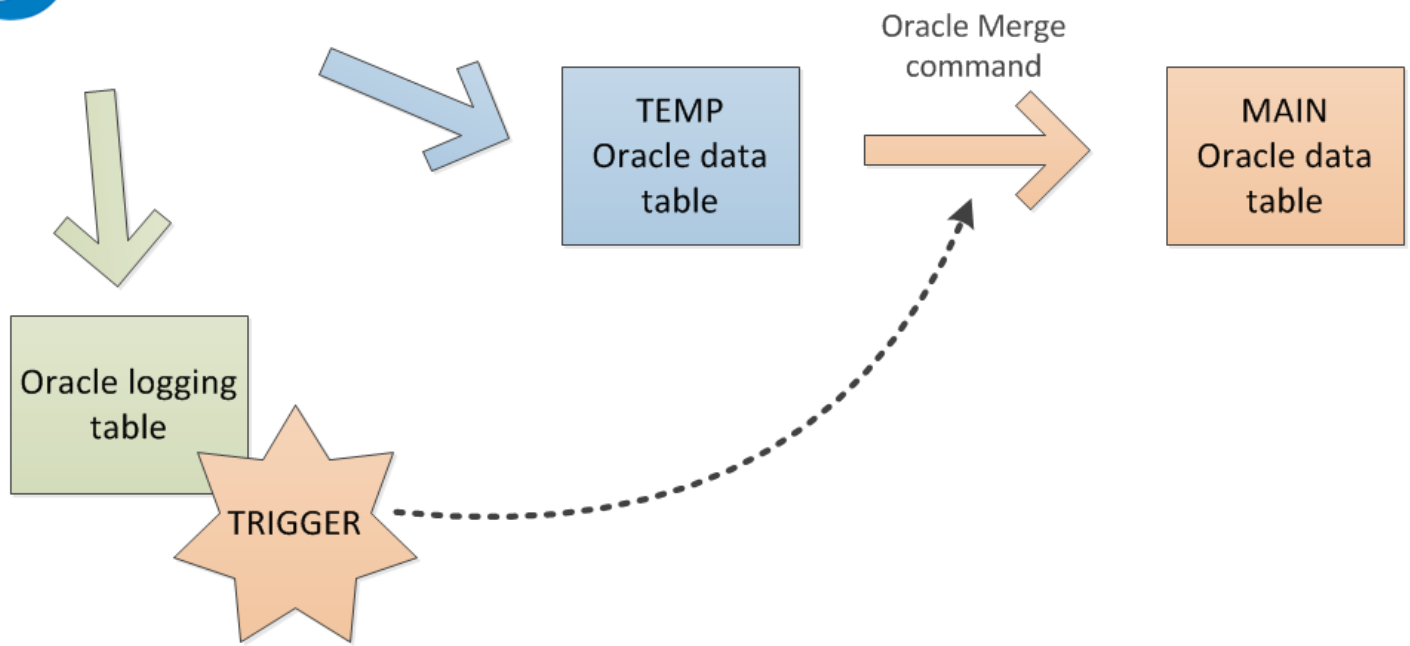
- SAS Table Loader transformation with Update/Insert load style + any type of "magic"
- Result:  it takes forever

# 2nd approach which doesn't work

- Use SAS Table Loader transformation with Replace load style + any type of "magic"
- Result: it causes application hiccups visible to users


TEMP
Oracle data
table


TEMP
Oracle data
table
Oracle logging
table


Oracle Merge command
TEMP Oracle data table
MAIN Oracle data table
Oracle logging table
TRIGGER

```
Create or replace trigger X
After insert on Logging_table

Begin
    MERGE INTO Main_table t1
       Using Temp_table t2
       On (t1.ID = t2.ID)
    When  Matched then
       Update Set
         t1.colum1 = ...
    When Not Matched then
       Insert ( column 1, ...) Values (...);

    Truncate table Temp_table;

End;
```

sas

+ Web Applications

= 

# Result

- working good (loading time = 1 minute)
- a lot of data for web applications is prepared like that in SAS 9.2 DWH now

sas

+ DOS Commands

= ?

challenge:

- Need to import data from ascii files into SAS staging table (csv or fixed field)
- Structure within each file is known
- File names and number of files to import is unknown

approach:

- Append data from all input files together to form the staging table
- Need the filenames to import
- How 2 loop?

DIR /B <folder>


Temporary table with file names

DIR /B <folder>

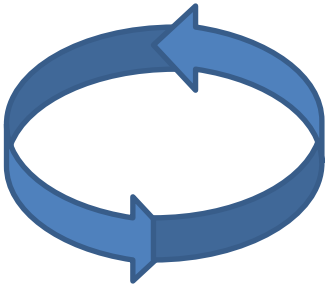

Temporary table with file names

```
%let import_folder=C:\Windows\Temp;
%let dircmd="dir /B /ON &import_folder.";
filename read pipe &dircmd.;

 data all_files;
  infile read;
  input;
  filename = _infile_;
 run;
```

DIR /B <folder>


Temporary table with file names

Macro looped import



Appended data table

```
/* 1. Empty table to hold the
appended samples */
DATA OUT_TABLE;
  LENGTH
    Id 8
    Name $ 50
FORMAT
    Id 8 BEST.
    Name $ 50.
run;
```

```
/* 2. Macro variable with number of
files to import */
proc sql noprint;
  select
    count(*)
  into
    :NObs
  from
    FILE_NAME_TABLE;
quit;
```

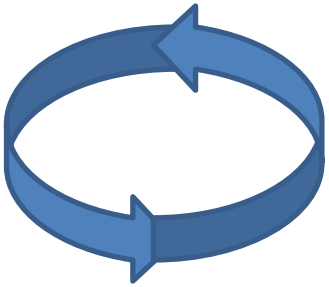
```
/* 3. Macro variables with file names */
proc sql noprint;
  select
    filename
    into :name1-:name%trim(&NObs.)
  from
    FILE_NAME_TABLE;
quit;
```

```
/* 4.  Macro to loop through each file and import, and then append data to target table */
%macro readFiles;
 %do i=1 %to &NObs;
  %let fileName="&folder.\&&name&i.";

  data OUT_TABLE_&i.;
   LENGTH
    Id 8
    Name $ 50
   FORMAT
    Id 8 BEST.
    Name $ 50.

   INFILE &filename.
    DSD FIRSTOBS=2;

   INPUT
    Id
    Name $ ;
  run;

  proc append base=OUT_TABLE DATA=OUT_TABLE_&i.;
  run;

 %end; /*do i=1 to NObs*/

%mend;

%readFiles;
```

DIR /B <folder>


Temporary table with file names

Macro looped import



Appended data table

sas

+ DOS Commands

= 

sas

+ XML

= ?

problem:

- A web application that handles XML documents and certain sections of the document need to be populated with data from SAS datawarehouse
- The default SAS XML generated from tables are not simple and easy for the application to parse and merge into the document

1st approach which doesn't work:

- XMLMap approach did not work well for exports.. We do use it for imports

2nd approach which doesn't work:

- After discussion with developer it was decided that SAS automatic XML format was too much to parse by application and table and format changes can be handled only by the change in application

# SAS XML output for a table


```
<?xml version="1.0" encoding="iso-8859-1" ?>
- <odsxml>
  - <head>
      <meta operator="sassrv" />
    </head>
  - <body>
    - <proc name="Print">
        <label name="IDX" />
      - <branch name="Print" label="The Print Procedure" class="ContentProcName" toc-level="1">
        - <leaf name="Print" label="Data Set WORK.CARS" class="ContentItem" toc-level="2">
          - <output name="Print" label="Data Set WORK.CARS" clabel="Data Set WORK.CARS">
            - <output-object type="table" class="Table">
              - <style>
                  <border spacing="1" padding="7" rules="groups" frame="box" />
                </style>
              - <colspecs columns="3">
                - <colgroup>
                    <colspec column="1" width="8" name="Make" type="string" class="Header" precision="0" scale="0" unformatted_type="string" unformatted_width="8" />
                    <colspec column="2" width="14" name="Model" type="string" class="Header" precision="0" scale="0" unformatted_type="string" unformatted_width="8" />
                    <colspec column="3" width="8" name="table_name" type="string" class="Header" precision="0" scale="0" unformatted_type="string" unformatted_width="8" />
                  </colgroup>
                </colspecs>
              - <output-head>
                - <row>
                  - <header name="Make" label="Make" type="string" class="Header" unformatted_type="string" unformatted_width="4" row="1" column="1">
                      <value>Make</value>
                    </header>
                  - <header name="Model" label="Model" type="string" class="Header" unformatted_type="string" unformatted_width="5" row="1" column="2">
                      <value>Model</value>
                    </header>
                  - <header name="table_name" label="table_name" type="string" class="Header" unformatted_type="string" unformatted_width="10" row="1" column="3">
                      <value>table_name</value>
                    </header>
                  </row>
                </output-head>
              - <output-body>
                - <row>
                  - <data name="Make" label="Make" type="string" class="Data" sasformat="$F" precision="0" scale="0" unformatted_type="string" unformatted_width="5" row="2" column="1">
                      <value>Acura</value>
                    </data>
                  - <data name="Model" label="Model" type="string" class="Data" sasformat="$F" precision="0" scale="0" unformatted_type="string" unformatted_width="3" row="2" column="2">
                      <value>MDX</value>
                    </data>
                  - <data name="table_name" label="table_name" type="string" class="Data" sasformat="$F" precision="0" scale="0" unformatted_type="string" unformatted_width="4" row="2" column="3">
                      <value>CARS</value>
                    </data>
                  </row>
                - <row>
                  - <data name="Make" label="Make" type="string" class="Data" sasformat="$F" precision="0" scale="0" unformatted_type="string" unformatted_width="5" row="3" column="1">
                      <value>Acura</value>
                    </data>
                  - <data name="Model" label="Model" type="string" class="Data" sasformat="$F" precision="0" scale="0" unformatted_type="string" unformatted_width="14" row="3" column="2">
                      <value>RSX Type S 2dr</value>
                    </data>
                  - <data name="table_name" label="table_name" type="string" class="Data" sasformat="$F" precision="0" scale="0" unformatted_type="string" unformatted_width="4" row="3" column="3">
                      <value>CARS</value>
                    </data>
                  </row>
                </output-body>
              </output-object>
            </output>
          </leaf>
        </branch>
      </proc>
    </body>
  </odsxml>
```

# workaround

- Select the table to be output in XML
- Do a proc contents on the table and change certain formats to generic formats

```
proc contents data=cars_test out=colm noprint;
QUIT;
```

```
data colm;
set colm;
if type=1 and (format='' or format='BEST') then format='number';
if type=1 and format='DATE' then format='date';
if type=1 and format='TIME' then format='time';
if type=2 and (format='' or format='$') then format='text';
table_name='CARS';
rename format=Column_type;
rename label=Column_label;
format Column_display_order best.;
Column_display_order=varnum;
rename name=column_name;
run;
```

```
PROC SQL;
   CREATE TABLE WORK.QUERY_FOR_CARS AS
   SELECT t1.Make,
          t1.Model,
          t1.table_name,
          t2.column_name,
          t2.Column_display_order,
          t2.Column_type
      FROM WORK.CARS t1 LEFT JOIN WORK.COLM t2 ON (t1.table_name = t2.table_name)
      ORDER BY t2.Column_display_order;
QUIT;
```

# Workaround

- Join the proc contents table with the data and then output in a nested loop

```
filename xmlout4 "/sas/test_custom.xml" encoding="utf-8";
data _null_ ;
file xmlout4;
set QUERY_FOR_CARS end=thatsit ;
by Table_Name Column_display_order;

if _n_ eq 1 then do;
    put '<?xml version="1.0" encoding="UTF-8"?>' ;
    end;
if first.Table_Name then do;
    put '<table name="' Table_Name '">' ;
end;
if first.Column_display_order then do;
    put '<column display_order="' Column_display_order '" type="' Column_Type '" name="' Column_name '">' ;
    put '</column>' ;
end;
if Make ne '' then do;
    put '<row>' ;
    put '<Make>' Make '</Make>' ;
    put '<Model>' Model '</Model>' ;
    put '</row>';
end;
if last.Table_Name then do;
    put '<last_updated when="' lastUpdated '" username="SAS" comments="data updated by SAS">';
    put '</last_updated>';
    put '</table>' ;
end;
run;
```

# SAS XML after workaround


```
<?xml version="1.0" encoding="UTF-8" ?>
- <table name="CARS">
  <column display_order="1" type="text" name="Make" />
  <column display_order="2" type="text" name="Model" />
  - <row>
    <Make>Acura</Make>
    <Model>MDX</Model>
  </row>
  - <row>
    <Make>Acura</Make>
    <Model>RSX Type S 2dr</Model>
  </row>
  - <row>
    <Make>Acura</Make>
    <Model>MDX</Model>
  </row>
  - <row>
    <Make>Acura</Make>
    <Model>RSX Type S 2dr</Model>
  </row>
  <last_updated when="." username="SAS" comments="data updated by SAS" />
</table>
```

sas

+ XML

= ♡

result:

- XML generated with the format as required by application.
- This workaround makes it easier to add more tables and columns to the existing tables without altering the application