

## АНОТАЦІЯ

Робота описує метод для оптимізації процесу оновлення специфікації SDTM (Модель табулювання даних дослідження) на протязі життєвого циклу дослідження. Зазвичай ці специфікації зберігаються в робочій книзі MS Excel і служать керівництвом для статистичних програмістів під час розробки програм SAS для створення наборів даних. Проте часті зміни цих специфікацій, часто внесені головним програмістом без повідомлення інших, можуть призвести до розходжень між атрибутами змінних набору даних та їх специфікаціями.

Запропоноване рішення включає використання Proc SQL та домену PE як приклади для оновлення атрибутів змінних, таких як LABEL, TYPE та FORMAT, кожен раз при запуску програми створення набору даних. Це досягається шляхом вбудовування макропідстановок в код Proc SQL, що дозволяє динамічно оновлювати ключові слова коду. Техніка використовує DOSUBL та %SYSFUNC для віртуального "введення" ключових слів коду, забезпечуючи миттєве оновлення SDTM програм при кожному їх запуску.

Впровадження цієї техніки динамічного оновлення коду може призвести до значної економії часу шляхом зменшення необхідності вручну оновлювати та уникати розходжень між атрибутами змінних набору даних та їх специфікаціями. Робота містить 23 сторінок.

## **ABSTRACT**

The abstract describes a method to streamline the process of updating SDTM (Study Data Tabulation Model) specification throughout the study lifecycle. Typically, these specifications are stored in an MS Excel workbook and serve as a guide for statistical programmers when developing dataset SAS programs. However, frequent changes to these specifications, often made by lead programmers without notifying others, can lead to discrepancies between dataset variable attributes and their specifications.

The proposed solution involves using Proc SQL and the PE domain as examples to update variable attributes such as LABEL, TYPE, and FORMAT whenever the dataset creation program is run. This is achieved by embedding macro calls in the Proc SQL code, allowing for dynamic updates to code keywords. The technique utilizes DOSUBL and %SYSFUNC to perform virtual "typing" of code keywords, ensuring that SDTM programs are instantly updated whenever they are run.

Implementing this dynamic code update technique can result in significant time savings by reducing the need for manual updates and preventing mismatches between dataset variable attributes and their specifications.. The note contains 23 pages.

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
Харківський національний університет імені В.Н.Каразіна  
Факультет математики і інформатики  
Кафедра теоретичної та прикладної інформатики

**Кваліфікаційна робота**  
**магістр**

на тему «Автоматичне оновлення SAS програми SDTM при внесенні змін до специфікації»

Виконав: студент 2 курсу, групи МФ-61  
спеціальність 122 «Комп'ютерні науки»  
освітньо-наукова програма  
«Інформатика»

Адаменко В.О.

---

(прізвище та ініціали)

Керівник Мовчун Т.О.

---

(прізвище та ініціали)

Рецензент

---

(прізвище та ініціали)

Харків – 2024 року

## **ЗМІСТ**

<b>ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....</b>	<b>3</b>
<b>ВСТУП.....</b>	<b>4</b>
<b>1.1 Мета і завдання дослідження.....</b>	<b>5</b>
<b>1.2 Методи дослідження та практичне значення одержаних результатів</b> <b>.....</b>	<b>6</b>
<b>ОСНОВНА ЧАСТИНА .....</b>	<b>7</b>
<b>2.1. Отримання інформації зі специфікацій.....</b>	<b>7</b>
<b>2.2. Цифровий робот 1 .....</b>	<b>9</b>
<b>2.2. Цифровий робот 2 .....</b>	<b>11</b>
<b>2.3 Повний код proc sql .....</b>	<b>14</b>
<b>ВИСНОВОК .....</b>	<b>22</b>
<b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....</b>	<b>23</b>

## **ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ**

SAS — Statistical Analysis System

SDTM — Study Data Tabulation Model

XML — Extensible Markup Language

SQL — Structured Query Language

PE — Physical Examinations.

## ВСТУП

У роботі описується метод для оптимізації процесу оновлення специфікацій SDTM (Модель табулювання даних дослідження) на протязі життєвого циклу клінічного дослідження. Зазвичай ці специфікації зберігаються в робочій книзі MS Excel і служать керівництвом для статистичних програмістів під час розробки програм SAS для створення наборів даних. Проте часті зміни цих специфікацій, часто внесені головним програмістом без повідомлення інших, можуть призвести до розходжень між атрибутами змінних набору даних та їх специфікаціями.

Для уникнення цього:

- (а) представлено концепцію "цифрових роботів" - малих макросів SAS, які використовують утиліти DOSUBL/SYSFUNC для виконання коду в межах операторів програм;
- б) демонструється, як такі цифрові макроси, вбудовані в Proc SQL, можуть зробити програми створення SDTM самоновачаючимися відповідно до змін у специфікаціях SDTM (зазвичай відомих як "специфікації").

## 1.1 Мета і завдання дослідження

Головна ідея полягає в тому, щоб отримати позицію рядка змінної у специфікаціях SDTM, використовувати цей номер позиції для читання значень атрибутів і використовувати ці значення для керування "цифровими роботами", які потім набирають відповідні рядки коду SQL. У цій роботі акцентується увага лише на чотирьох атрибутах змінної: LENGTH, TYPE, LABEL та FORMAT. Для LENGTH та LABEL ми просто використовуємо невеликий макрос %GET\_ATTR ("Цифровий робот 1"), який виконує заміну тексту за допомогою макрових змінних, дуже базову техніку використання макросів. Але для атрибута TYPE, від якого залежить FORMAT, підхід з використанням макрових змінних не є достатнім. Потрібно визначити, чи була змінена змінна з символьного на числовий тип або навпаки, а потім використати відповідну функцію PUT або INPUT для оновлення атрибута змінної. Це потребує додаткової обробки і, отже, більш вдосконаленого макросу ("цифровий робот 2"). Обидва макроси називаються "цифровими роботами", оскільки вони присутні у коді Proc SQL як виклики макросу, а не як макрові змінні.

## **1.2 Методи дослідження та практичне значення одержаних результатів**

Запропоноване рішення включає використання Proc SQL та домену PE як приклади для оновлення атрибутів змінних, таких як LABEL, TYPE та FORMAT, кожен раз при запуску програми створення набору даних. Це досягається шляхом вбудовування макropідстановок в код Proc SQL, що дозволяє динамічно оновлювати ключові слова коду. Техніка використовує DOSUBL та %SYSFUNC для віртуального "введення" ключових слів коду, забезпечуючи миттєве оновлення SDTM програм при кожному їх запуску.

Впровадження цієї техніки динамічного оновлення коду може призвести до значної економії часу шляхом зменшення необхідності вручну оновлювати та уникати розходжень між атрибутами змінних набору даних та їх специфікаціями.



## ОСНОВНА ЧАСТИНА

### 2.1. Отримання інформації зі специфікацій

Перший крок у процесі - прочитати файл Excel із специфікаціями SDTM та створити макро-змінні: одну, що містить повний список назв змінних, як вони відображені в документі Excel, іншу - відповідний список міток змінних, третю - відповідний список довжин, і четверту - список типів змінних:

```
%let specification_path = %str(&path.\StudySpecifications);
```

```
%let input_specs_file = SDTM_Specs.xls;
```

```
%let sheet_name = PE;
```

```
%let specs_range = "&sheet_name.$A2:E100"n;
```

```
%let output_specs = sdtm_specs;
```

```
* Імпорт даних із Excel;
```

```
proc import datafile="&specification_path.\&input_specs_file."
```

```
    out=&output_specs
```

```
    dbms=xls replace;
```

```
    getnames=YES;
```

```
    mixed=YES;
```

```
    range=&specs_range;
```

```
run;
```

```
*Створення макрозмінних для списків атрибутів;
```

```
proc sql noprint;
```

```
    select variable_name
```

```
into :variable_list separated by ","  
  
from sdtm_specs;  
  
select length  
  
into :len1 - :len&sysmaxlong  
  
from sdtm_specs;  
  
select type  
  
into :type1 - :type&sysmaxlong  
  
from sdtm_specs;  
  
select label  
  
into :label1 - :label&sysmaxlong  
  
from sdtm_specs;  
  
quit;
```

Для LENGTH, LABEL та TYPE ми просто використовуємо невеликий макрос %GET\_ATTR, який називається "Цифровий робот 1".

## 2.2. Цифровий робот 1

Перший макрос, %get\_attr, отримує номер рядка змінної у списку змінних, витягнутому з робочого аркуша Excel зі специфікаціями SDTM для певного домену. Номер рядка потім використовується для зчитування LENGTH, LABEL або TYPE в макрову змінну &value, значення якої потім "вводиться" в рядок коду SQL:

```
%macro %get_attr (var, attr);  
  
    %global attr_val delim amp n value;  
  
    %let attr_val = %str(&attr);  
  
    %let delim = %str();  
  
    %let amp = %nrstr(&);  
  
    %let n = %sysfunc(whichc(&var, &varnamelist));  
  
    %let value = %sysfunc(catx(&delim, &amp, &attr_val, &n));  
  
    &value.  
  
%mend %get_attr;
```

Реалізація у коді sql:

```
proc sql noprint;  
  
    create table PE as  
  
    select  
  
        /* DOMAIN */  
  
        strip(domain) as DOMAIN
```

```
length = % get_attr (DOMAIN, len)

%sysfunc(ifc((%get_attr (DOMAIN, typ) = 'Char'),

format = $% get_attr (DOMAIN, len).,

format = % get_attr (DOMAIN, len).))

label = "% get_attr (DOMAIN, lab)",
```

Функція IFC використовується для вибору відповідного формату на основі типу змінної. Проте це буде вірно лише у випадку, якщо тип змінної точно відповідає типу, що вже пов'язаний з змінною в програмі. Якщо специфікації були оновлені таким чином, що тип змінної змінився, заміна тексту за допомогою макро-змінної не запобігає виникненню помилки SAS. Потрібно повторно встановити тип змінної в програмі, використовуючи функції конвертації типу PUT або INPUT. Це роль, яку відіграє Цифровий робот 2.

## 2.2. Цифровий робот 2

Цей цифровий робот просто є невеликим макросом SAS, %DETERMINE\_TYPE, який може виконати функцію PUT або INPUT у коді SQL на основі атрибуту TYPE змінної, використовуючи функцію DOSUBL та утиліту %SYSFUNC, яка змушує виконуватися DOSUBL. (Цей макрос %DETERMINE\_TYPE також використовує макрос %GET\_ATTR для отримання атрибуту TYPE).

```
%let input_dataset = PE;
```

```
%macro determine_type(variable);
```

```
  %let result = %sysfunc(dosubl('
```

```
    data _null_;
```

```
      length var_type $5 var_text $50;
```

```
      ds_id = open("&input_dataset");
```

```
      num_obs = attrn(ds_id, "NOBS");
```

```
      num_vars = attrn(ds_id, "NVAR");
```

```
      rc = fetchobs(ds_id, 1);
```

```
      var_num = varnum(ds_id, "&variable");
```

```
      var_type = vartype(ds_id, var_num);
```

```
      if (var_type = "N") and (%retrieve_attr(&variable, type) = %str(Char)) then
```

```
        var_text = "put(&variable., 8.) as &variable. ";
```

```
      else if (var_type = "C") and (%retrieve_attr(&variable, type) = %str(Num))
```

```
then
```

```

        var_text = "input(&variable., 8.) as &variable. ";

    else

        var_text = "&variable.";

    rc = close(ds_id);

    call symputx("var_text_out", var_text, "g");

run;'

));

%str(&var_text_out)

%mend determine_type;

```

Реалізація у коді sql:

```
/*SUBJID*/
```

```
proc sql noprint;
```

```
    create table PE as
```

```
    select
```

```
/* Приклад використання макросу DETERMINE_TYPE для змінної SUBJID*/
```

```
    %determine_type(SUBJID)
```

```
    length = % get_attr(SUBJID, length)
```

```
    %sysfunc(ifc((%get_attr(SUBJID, type) = 'Char'),
```

```
        format = $%get_attr (SUBJID, length).,
```

```
        format = %get_attr (SUBJID, length).))
```

```
    label = "%get_attr (SUBJID, label)"
```

У цьому випадку макрос %DETERMINE\_TYPE також викликається для виконання функції PUT або INPUT для конвертації типу змінної.

Всі змінні можуть мати виклик макросу %DETERMINE\_TYPE у коді SQL, але в цій рооті використання обмежується лише змінною SUBJID, яка може мати як числовий, так і символний тип змінної.

Легко можна продемонструвати, що незалежно від того, яким чином тип змінної в специфікаціях SDTM позначений як "Num" або "Char", PE створюється без будь-яких помилок SAS, і змінна SUBJID виглядає як символна або числова, як вказано в специфікаціях.

Зразок специфікацій SDTM

Dataset	Variable Name	Label	Type	Length
PE	STUDYID	Study Identifier	Char	15
PE	DOMAIN	Domain Abbreviation	Char	2
PE	SUBJID	Subject Identifier	Char	10
PE	USUBJID	Unique Subject Identifier	Char	25
PE	PESEQ	Sequence Number	Num	8
PE	PETESTCD	Body System Examined Short Name	Char	40
PE	PETEST	Body System Examined	Char	40
PE	PEORRES	Verbatim Examination Finding	Char	200
PE	PESTRESC	Character Result/Finding in Standard Format	Char	200
PE	PESTAT	Completion Status	Char	8
PE	VISITNUM	Visit Number	Num	8
PE	VISIT	Visit Name	Char	60
PE	PEDTC	Date/Time of Examination	Char	19
PE	PEDY	Study Day of Examination	Num	8

Для ясності не всі стовпці специфікацій відображені. У цій статті акцентується лише на трьох атрибутах змінних: LABEL, TYPE та LENGTH. Атрибут TYPE також дозволяє призначити FORMAT у коді SQL.

## 2.3 Повний код proc sql

```
data testPE;
infile

datalines;
input

Prot $ Domain $ Subjid $ seq ExmSite $ Result $ Status $ VisitName $
ExmDt date9. ExamDay;
datalines;
BPxxx pe 1112 1 Skin Acne Abnormal Done Baseline 24Sep2013 -8
BPxxx pe 1112 2 Head Abnormal Done Baseline 24Sep2013 -8
BPxxx pe 1112 3 Neck Normal Done Baseline 24Sep2013 -8
BPxxx pe 1113 1 Skin Rash Normal Done Baseline 24Sep2013 -8
BPxxx pe 1113 2 Head Done Baseline 24Sep2013 -8
BPxxx pe 1113 3 Skin Rash Abnormal Done Baseline 24Sep2013 -8
BPxxx pe 1114 1 Head Abnormal Done Baseline 24Sep2013 -8
BPxxx pe 1114 2 Skin Rash Done Baseline 24Sep2013 -8
BPxxx pe 1114 3 Neck Normal Done Baseline 24Sep2013 -8
BPxxx pe 1115 1 Skin Acne Normal Done Baseline 24Sep2013 -8
BPxxx pe 1115 2 Neck Abnormal Done Baseline 24Sep2013 -8
BPxxx pe 1115 3 Neck Normal Done Baseline 24Sep2013 -8
;
run;

%macro GenerateSDTM();

    /***** Підключення специфікацій SDTM із Excel*****/

    %let specs_path = %str(C:\M);
```



```

%let input_specs = SDTM_Specs.xlsx;

%let sheet_name = PE;

%let specs_range = "&sheet_name.$A1:E20"n;

%let output_specs = sdtm_specs;


proc import datafile="&specs_path.\&input_specs."

    out=&output_specs

    dbms=xlsx replace;

    getnames = YES;

    mixed = YES;

    range = &specs_range;

run;


/***** Визначення макрозмінних для атрибутів *****/

proc sql noprint;

    select variable_name into :var_list separated by ","

    from &output_specs;


    select length into :lengths1 - :lengths&sysmaxlong

    from &output_specs;


    select type into :types1 - :types&sysmaxlong

```

```

from &output_specs;

select label into :labels1 - :labels&sysmaxlong

from &output_specs;

quit;

/***** Макрос для отримання атрибутів змінної *****/

%macro get_attr(var, attr);

    %global attr_val;

    %let n = %sysfunc(whichc(&var, &var_list));

    %let attr_val = %sysfunc(catx(, &attr, &n));

    &attr_val.

%mend get_attr;

/***** Макрос для визначення типу змінної *****/

%let dataset_name = testPE;

%macro determine_type(variable);

    %let ret_value = %sysfunc(dosubl('

        data _null_;

            length vtype $5 var_text_out $50;

            dsid = open("&dataset_name");

            vnum = varnum(dsid, "& variable ");

```

```

vtype = vartype(dsid, vnum);

if (vtype = "N" and %get_attr(&variable, types) = "Char") then

    var_text_out = "put(&variable, 8.) as &variable ";

else if (vtype = "C" and %get_attr(&variable, types) = "Num") then

    var_text_out = "input(&variable, 8.) as & variable";

else

    var_text_out = "&variable";

call symputx("var_text", var_text_out, "g");

rc = close(dsid);

run;

));

%str(&var_text.)

%mend determine_type;

/***** ОСНОВНИЙ розділ створення SDTM *****/

proc sql noprint;

    create table PE as

    select

        /* STUDYID */

        strip(prot) as STUDYID

        length = %get_attr(STUDYID, lengths)

```

```

        format = %sysfunc(ifc(%get_attr(STUDYID, types) = 'Char',
$%get_attr(STUDYID, lengths)., %get_attr(STUDYID, lengths).))

        label = "%get_attr(STUDYID, labels)",

/* DOMAIN */

DOMAIN

        length = %get_attr(DOMAIN, lengths)

        format = %sysfunc(ifc(%get_attr(DOMAIN, types) = 'Char',
$%get_attr(DOMAIN, lengths)., %get_attr(DOMAIN, lengths).))

        label = "%get_attr(DOMAIN, labels)",

/* SUBJID */

%determine_type(SUBJID)

        length = %get_attr(SUBJID, lengths)

        format = %sysfunc(ifc(%get_attr(SUBJID, types) = 'Char',
$%get_attr(SUBJID, lengths)., %get_attr(SUBJID, lengths).))

        label = "%get_attr(SUBJID, labels)",

/* USUBJID */

compress(strip(prot) || '-' || strip(SUBJID)) as USUBJID

        length = %get_attr(USUBJID, lengths)

        format = %sysfunc(ifc(%get_attr(USUBJID, types) = 'Char',
$%get_attr(USUBJID, lengths)., %get_attr(USUBJID, lengths).))

        label = "%get_attr(USUBJID, labels)",

/* PESEQ */

seq as PESEQ

```

```

length = %get_attr(PESEQ, lengths)

format = %sysfunc(ifc(%get_attr(PESEQ, types) = 'Char',
$%get_attr(PESEQ, lengths)., %get_attr(PESEQ, lengths).))

label = "%get_attr(PESEQ, labels)",

/* PETESTCD */

substrn(strip(uppercase(ExamSite)), 1, 8) as PETESTCD

length = %get_attr(PETESTCD, lengths)

format = %sysfunc(ifc(%get_attr(PETESTCD, types) = 'Char',
$%get_attr(PETESTCD, lengths)., %get_attr(PETESTCD, lengths).))

label = "%get_attr(PETESTCD, labels)",

/* PETEST */

strip(ExamSite) as PETEST

length = %get_attr(PETEST, lengths)

format = %sysfunc(ifc(%get_attr(PETEST, types) = 'Char',
$%get_attr(PETEST, lengths)., %get_attr(PETEST, lengths).))

label = "%get_attr(PETEST, labels)",

/* PEORRES */

strip(result) as PEORRES

length = %get_attr(PEORRES, lengths)

format = %sysfunc(ifc(%get_attr(PEORRES, types) = 'Char',
$%get_attr(PEORRES, lengths)., %get_attr(PEORRES, lengths).))

label = "%get_attr(PEORRES, labels)",

/* PESTRESC */

```

```

strip(result) as PESTRESC

length = %get_attr(PESTRESC, lengths)

format = %sysfunc(ifc(%get_attr(PESTRESC, types) = 'Char',
$%get_attr(PESTRESC, lengths)., %get_attr(PESTRESC, lengths).))

label = "%get_attr(PESTRESC, labels)",

/* PESTAT */

status as PESTAT

length = %get_attr(PESTAT, lengths)

format = %sysfunc(ifc(%get_attr(PESTAT, types) = 'Char',
$%get_attr(PESTAT, lengths)., %get_attr(PESTAT, lengths).))

label = "%get_attr(PESTAT, labels)",

/* VISITNUM */

whichc(visitname, "Baseline", "Week1", "Week2") as VISITNUM

length = %get_attr(VISITNUM, lengths)

format = %sysfunc(ifc(%get_attr(VISITNUM, types) = 'Char',
$%get_attr(VISITNUM, lengths)., %get_attr(VISITNUM, lengths).))

label = "%get_attr(VISITNUM, labels)",

/* VISIT */

VisitName as VISIT

length = %get_attr(VISIT, lengths)

format = %sysfunc(ifc(%get_attr(VISIT, types) = 'Char',
$%get_attr(VISIT, lengths)., %get_attr(VISIT, lengths).))

label = "%get_attr(VISIT, labels)",

```

```

/* PEDTC */

case

    when not missing(ExmDt) then put(ExmDt, IS8601DA.)

    else "

end as PEDTC

length = %get_attr(PEDTC, lengths)

format = %sysfunc(ifc(%get_attr(PEDTC, types) = 'Char',
$%get_attr(PEDTC, lengths)., %get_attr(PEDTC, lengths).))

label = "%get_attr(PEDTC, labels)",

/* PEDY */

ExamDay as PEDY

length = %get_attr(PEDY, lengths)

format = %sysfunc(ifc(%get_attr(PEDY, types) = 'Char',
$%get_attr(PEDY, lengths)., %get_attr(PEDY, lengths).))

label = "%get_attr(PEDY, labels)"

from testPE

order by STUDYID, USUBJID, PETESTCD, VISITNUM, PEDTC;

quit;

%mend GenerateSDTM;

%GenerateSDTM();

```

## ВИСНОВОК

У роботі демонструється можливість самоновачання програми SAS на основі змін зовнішнього джерела. Це стосується того, що виклики макросів можуть бути вбудовані в оператори програмування. Складні алгоритми можуть бути виконані як виклики макросів за допомогою утиліт %SYSFUNC та DOSUBL SAS, що робить такі виклики макросів "цифровими роботами". Техніка була застосована для створення набору даних з фізичного обстеження, PE, на основі специфікацій, поданих на робочому аркуші Excel. Оскільки ці цифрові роботи завжди читають зовнішні специфікації, будь-які оновлення, внесені в специфікації, автоматично відображаються в віртуальному наборі даних, створеному цифровими роботами. У майбутньому цілком можливо розширити техніку на стовпець похідних у робочому аркуші специфікацій SDTM.



## **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. SAS Institute Inc. and World Programming Limited (England and Wales High Court (Chancery Division) July 23, 2010).
2. Delwiche, Lora D.; Susan J. Slaughter (2012). The Little SAS Book: A Primer: a Programming Approach. SAS Institute.
3. Li, Arthur (10 April 2013). Handbook of SAS DATA Step Programming. CRC Press.
4. Buck, Debbie. "A Hands-On Introduction to SAS DATA Step Programming" (PDF). SUGI 30: SAS Institute. Retrieved October 2, 2013.
5. Bass, N. Jyoti; K. Madhavi Lata & Kogent Solutions (1 September 2007). Base Sas Programming Black Book, 2007 Ed. Dreamtech Press
6. Tolbert, William (December 1, 2010). "How to Win Friends and Influence People with the SAS Output Delivery System". Clinical Medicine & Research.