

Understanding Macro Variable Scopes

Eric Wang, Programmer Analyst
Health Quality Council, SK, Canada

www.hqc.sk.ca

SASKATCHEWAN
**HEALTH
QUALITY**
COUNCIL



MEASURING AND
REPORTING FOR
LEARNING AND
IMPROVEMENT



Topics

- ❖ **Introducing the ways to create macro variables**
- ❖ **Understanding the macro variable scopes**
- ❖ **Learning some good programming practice**



Introducing Macro Variables

In term of scopes, there are two types:

- **Global macro variables**



- **Local macro variables**



Introducing Macro Variables

- **Global macro variables**
 - Stored in global symbol table
 - Always available during your SAS session

Global Symbol Table	
SYSDATE	04OCT10
SYSTIME	11:16
city	Saskatoon

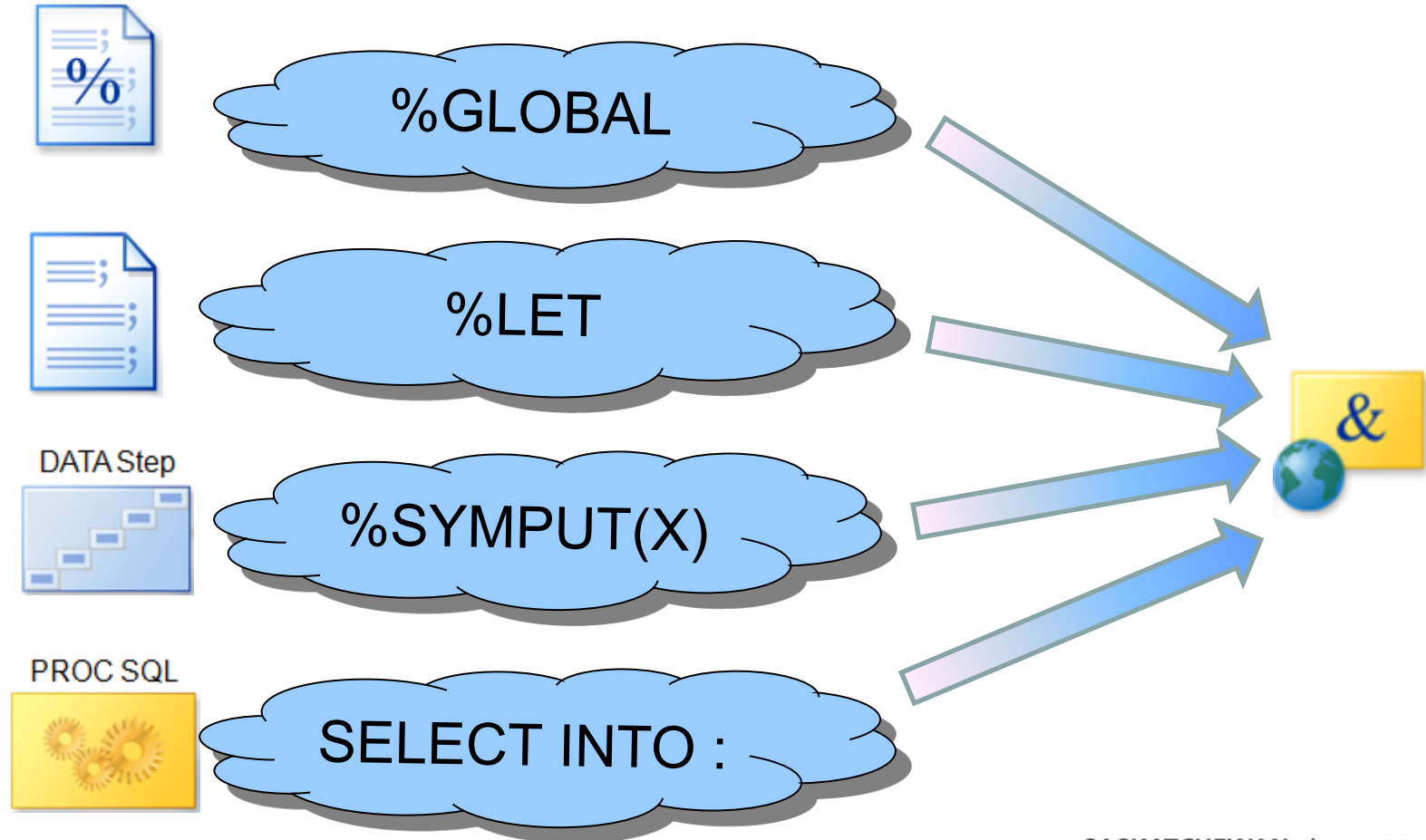


Introducing Macro Variables

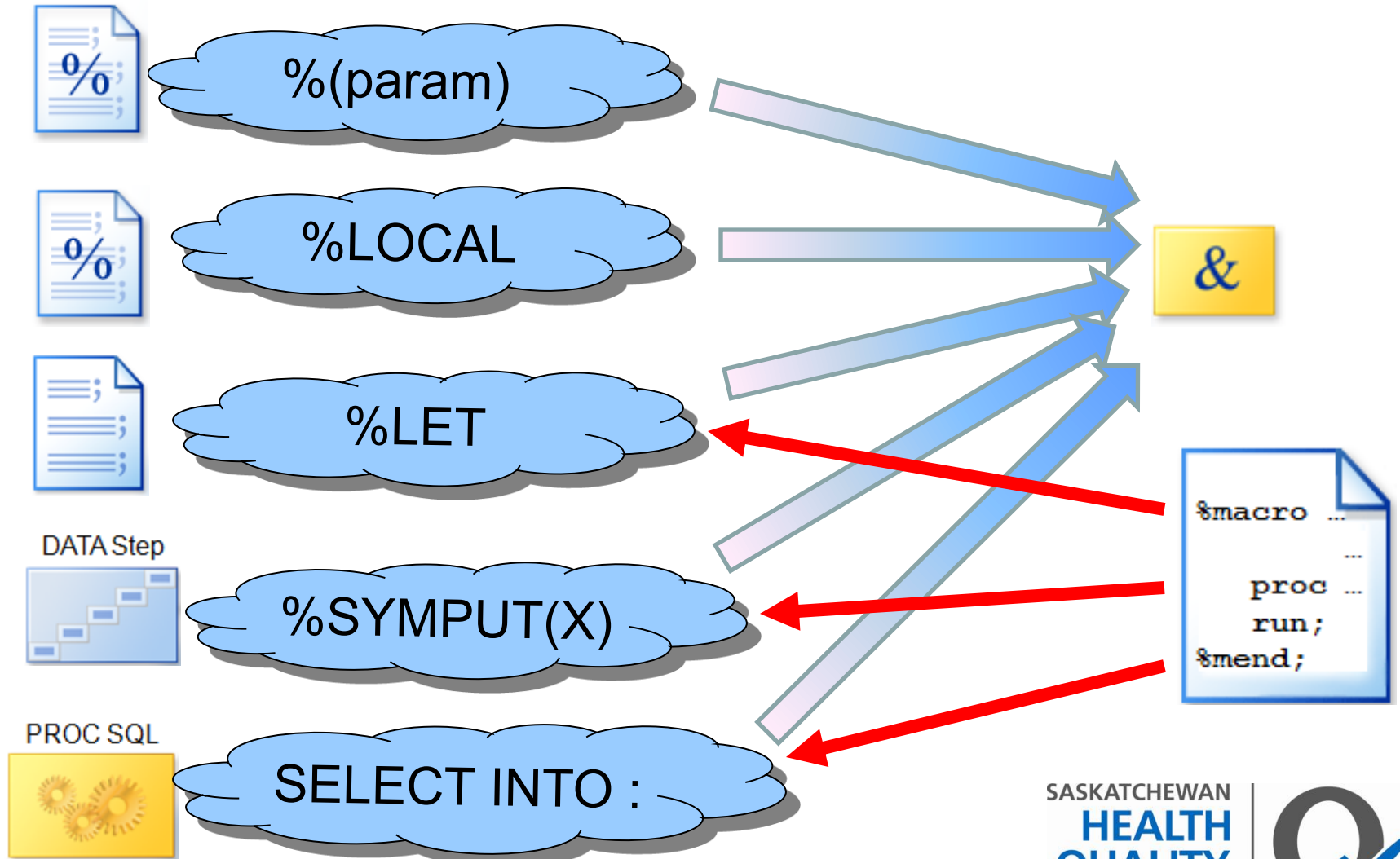
- **Local macro variables**
 - Stored in local symbol table
 - Only available during macro execution

Local Symbol Table	
dsn	work.temp
vars	age sex

Creating Global Macro Variables



Creating Local Macro Variables



General Rules

- **Macro processor always attempts to update the value for an existing macro variable rather than create a new one**
- **CALL SYMPUT (SYMPUTX) is special:**
It creates macro variables only if the current local symbol table is not empty, otherwise, it usually creates the variable in the closest nonempty symbol table



Examples

→ `%let new=inventory;`

→ `%macro name1;`

→ `%let new=report;`

`%mend name1;`

Global Symbol Table	
SYSDATE	04OCT10
new	report

Local Symbol Table	

→ `%name1`

→ `data &new;`



Examples

➔ `%let new=inventory;`

➔ `%macro name2;`

➔ `%let new=report;`

➔ `%let old=warehouse;`

`%mend name2;`

Global Symbol Table	
SYSDATE	04OCT10
new	report

Local Symbol Table	
old	warehouse

➔ `%name2`

➔ `data &new;`

➔ `set &old;`

`run;`

```
1463 data &new;
1464 set &old;
-
22
200
```

WARNING: Apparent symbolic reference OLD not resolved.
ERROR: File WORK.OLD.DATA does not exist.

Understanding Macro Variable Scopes

Same rules applies to multi-
level nesting!



Examples

```
→ %macro condition;  
→   %let old=sales;  
→   %let cond=cases>0;  
→ %mend condition;
```

Local Symbol Table	
cond	Cases>0

```
→ %macro name3;  
→   %let old=warehouse;  
→   %condition  
→   %put '&old' = &old, '&cond' = &cond;  
→ %mend name3;
```

Local Symbol Table	
old	sales

```
→ %name3
```

```
1500 %name3  
WARNING: Apparent symbolic reference COND not resolved.  
'&old' = sales, '&cond' = &cond
```



Be Careful to CALL SYMPUT(X)

It creates macro variables only if the current local symbol table is not empty, otherwise, it usually creates the variable in the closest nonempty symbol table



Examples

```
%macro sample1;
```

```
data _null_;
```

```
x='a token';
```

```
call symput('myvar1', x);
```

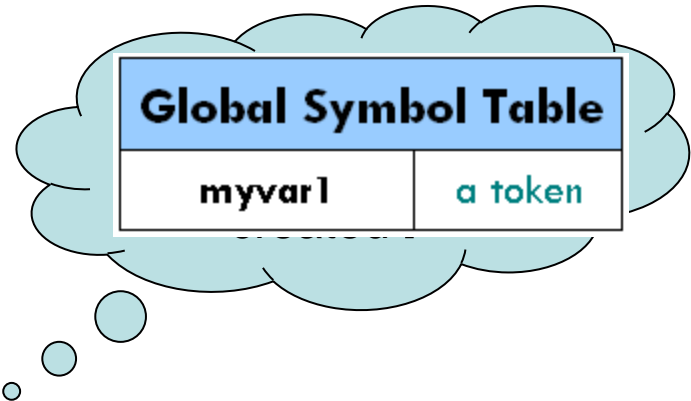
```
run;
```

```
%mend sample1;
```

```
1533 %put '&myvar1'=&myvar1;  
      '&myvar1'=a token
```

```
%sample1
```

```
➔ %put '&myvar1'=&myvar1;
```



Examples

- `%macro sample2(param1);`
- `data _null_;`
- `x='a token';`
- `call symput('myvar1', x);`
- `run;`
- `%mend sample2;`

Local Symbol Table	
param1	10
<u>Myvar1</u>	<u>a token</u>

→ `%sample2(10)`

```
WARNING: Apparent symbolic reference MYVAR1 not resolved.  
1552 %put '&myvar1'=&myvar1;  
      '&myvar1'=&myvar1
```

→ `%put '&myvar1'=&myvar1;`

Special Cases for CALL SYMPTUT(X)

Please see reference for more details.



Good Programming Practice

❖ Always try to create local variables

✓ %local

✓ CALL SYMPUT('myvar', 'myval', 'L')



❖ Delete global variables when you don't need them in your program

✓ %sysdel myvar

Reference

<http://www.otago.ac.nz/sas/macro/z1072111.htm>



Questions?