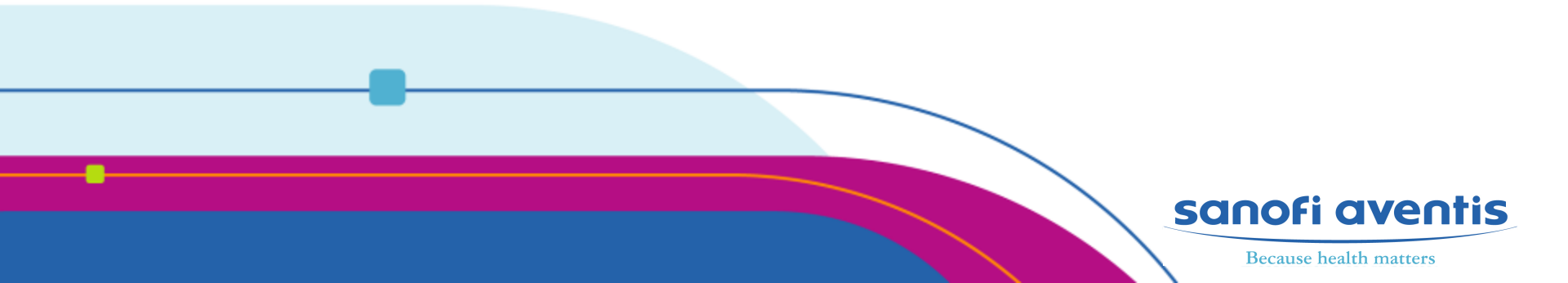




For a programming more efficient

Claude Guyot
PhUSE 2010 – Berlin
Paper CS05



sanofi aventis

Because health matters

Agenda

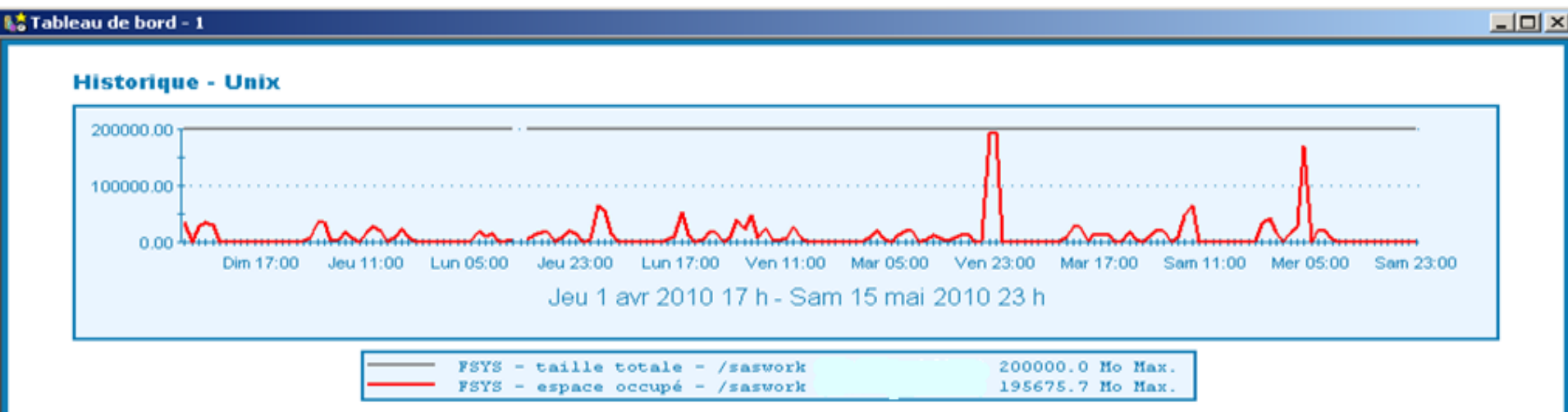
- Reasons for a programming more efficient
- Save the time of programming
- Save your time and the time of the others
- Save the space

Reasons for a programming more efficient

- Requirements of Health authorities lead to work with more and more data
 - Data pooling
 - Multiplication of new standards
 - Answers to Health Authorities on recent and old studies
- Deadlines often very short

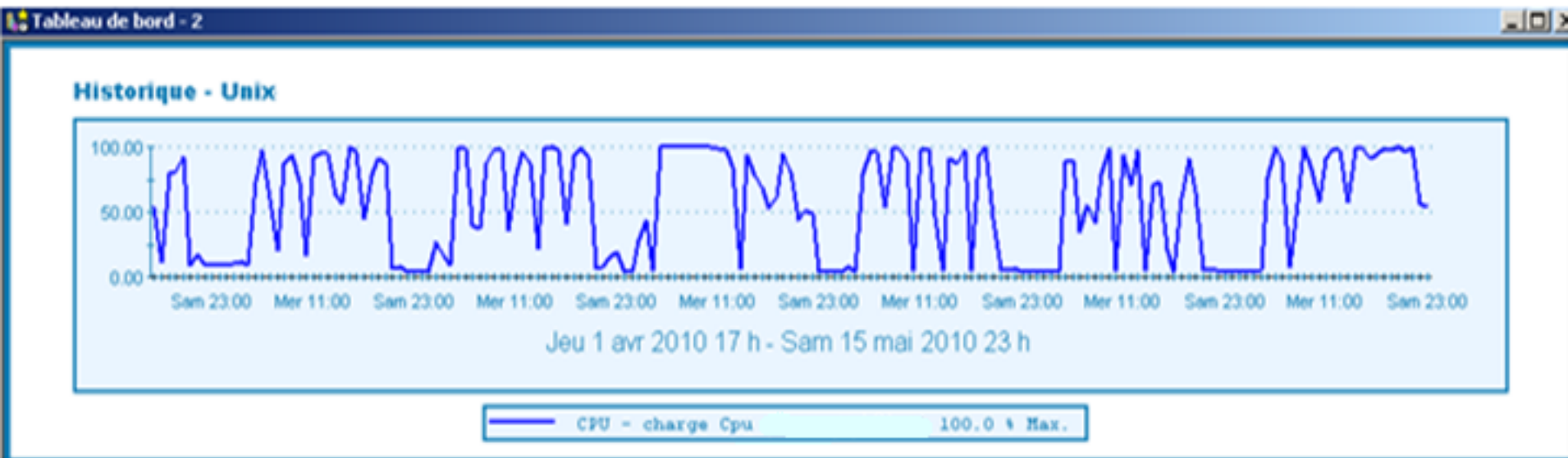
Reasons for a programming more efficient

■ Full SAS work



Reasons for a programming more efficient

■ Slowness of the network



Save the time of programming

■ Programs must be clear

- A program is not just a succession of statements, data steps or procedures in only one block
- All the different steps must be well dissociated by
 - changing lines for each new statement,
 - skipping lines to differentiate clearly all the parts,
 - using the indentation to see clearly the beginning and the end of a data step, of a procedure or of a DO loop...

Save the time of programming

- Programs must be well documented
 - Use a header at the beginning of the program to give:
 - all useful information to explain the goal of the program,
 - the pre-requisites necessary for the program,
 - the history of all the changes/updates of the program
 - the people who have worked on the program (author and all people who have brought updates)
 - Insert comments in the program to explain each major step

Save the time of programming

■ In Sanofi-Aventis

- all these rules are described in the Good Programming Practice (GPP)
- For some standard macro-programs, a document is written for each macro
 - same information as the header but more detailed
 - Explanation of the different macro-parameters
 - different concret examples
 - limits of the macro

■ In conclusion, think to facilitate the work of the others

Save your time and the time of the others

■ Simple rules

■ Select the variables

- Keep only what it is useful for the aim of the program in the data steps and in the procedures to reduce the quantity of data to manage
- Gain CPU time and work space
- Statement KEEP to keep only useful variables
- Statement DROP to drop unnecessary variables

```

5 proc sort data=rdata.PATIENT out=PATIENT1 ;
6   by PATID ;
7 run ;

```

NOTE: There were 17677 observations read from the data set RDATA.PATIENT.
 NOTE: The data set WORK.PATIENT1 has 17677 observations and 115 variables.
 NOTE: PROCEDURE SORT used (Total process time):

real time	5.32 seconds	
user cpu time	0.03 seconds	
system cpu time	0.26 seconds	
Memory		29253k
OS Memory		35440k
Timestamp	18/10/2010	16:06:47

```

5 proc sort data=rdata.PATIENT (keep=PATID PRACID SEX AGE DIABETE DRUG SUBJID) out=PATIENT2 ;
6   by PATID ;
7 run ;

```

NOTE: There were 17677 observations read from the data set RDATA.PATIENT.
 NOTE: The data set WORK.PATIENT2 has 17677 observations and 7 variables.
 NOTE: PROCEDURE SORT used (Total process time):

real time	0.04 seconds	
user cpu time	0.00 seconds	
system cpu time	0.04 seconds	
Memory		1563k
OS Memory		6256k
Timestamp	18/10/2010	16:08:35

```

5 proc sort data=rdata.PATIENT out=PATIENT3 (keep=PATID PRACID SEX AGE DIABETE DRUG SUBJID) ;
6   by PATID ;
7 run ;

```

NOTE: There were 17677 observations read from the data set RDATA.PATIENT.
 NOTE: The data set WORK.PATIENT3 has 17677 observations and 7 variables.
 NOTE: PROCEDURE SORT used (Total process time):

real time	0.07 seconds	
user cpu time	0.03 seconds	
system cpu time	0.04 seconds	
Memory		29169k
OS Memory		35696k
Timestamp	18/10/2010	16:09:45

Save your time and the time of the others

- The WHERE clause instead of IF statement
 - In a DATA step, the WHERE clause allows to select records without reading all the dataset
 - The IF condition reads all the dataset and then selects the records

```

52 data patient1 ;
53   set rdata.patient1 ;
54   if sexc="Female" ;
55 run ;

```

NOTE: There were 12617 observations read from the data set RDATA.PATIENT1.

NOTE: The data set WORK.PATIENT1 has 5799 observations and 111 variables.

NOTE: DATA statement used (Total process time):

real time	2.57 seconds	
user cpu time	0.03 seconds	
system cpu time	0.12 seconds	
Memory		430k
OS Memory		6000k
Timestamp	18/10/2010	17:16:38

```

56
57
58 data patient2 ;
59   set rdata.patient1 ;
60   where sexc="Female" ;
61 run ;

```

NOTE: There were 5799 observations read from the data set RDATA.PATIENT1.

WHERE sexc='Female';

NOTE: The data set WORK.PATIENT2 has 5799 observations and 111 variables.

NOTE: DATA statement used (Total process time):

real time	0.26 seconds	
user cpu time	0.00 seconds	
system cpu time	0.12 seconds	
Memory		450k
OS Memory		6000k
Timestamp	18/10/2010	17:16:38

Save your time and the time of the others

- Choice the better way to join 2 datasets –
Example: select the records of a dataset according to a selection done in a first dataset

PATIENT dataset

	PATID	SEX	expoc	PR
1	0001	2	Yes	c97
2	0002	2	No	h98
3	0005	2	Yes	b98
4	0007	1	Yes	a99
5	0009	2	Yes	a97

THER dataset

	Subject identifier	Start oftreatment	End oftreatment	DR
2587	0005	23/12/2003	03/05/2006	917
2588	0005	23/12/2003	03/05/2006	882
2589	0007	08/10/2007	17/02/2009	992
2590	0007	08/10/2007	17/02/2009	962
2591	0007	08/10/2007	17/02/2009	961

PAT_THER_joint dataset

	Subject identifier	Start oftreatment	End oftreatment	DR
1545	0007	16/03/2005	07/01/2009	971
1546	0007	16/03/2005	07/01/2009	898
1547	0007	16/03/2005	07/01/2009	987
1548	0007	06/03/2003	17/12/2008	995
1549	0007	06/03/2003	17/12/2008	995

Save your time and the time of the others

- Merge of both datasets
 - 2 PROC SORT
 - 1 data step to merge

```
/* Joint with a MERGE */  
❏ proc sort data=rdata.PATIENT1 (keep=PATID SEX) out=PATIENT2 (drop=SEX) ;  
    by PATID ;  
    where SEX='1' ;  
run ;  
  
❏ proc sort data=rdata.THER out=THER1 ;  
    by PATID ;  
run ;  
  
❏ data PATTHER_MERGE ;  
    merge PATIENT2 (in=a) THER1 (in=b) ;  
    by PATID ;  
    if a and b ;  
run ;
```

Save your time and the time of the others

■ Joint with SQL

```
/* Joint with SQL */  
❏ proc sql ;  
    create table PATTHER_SQL (drop=_PATID)  
    as select a.PATID, b.*  
    from rdata.PATIENT1 (keep=PATID SEX) as a, rdata.THER (rename=(PATID=_PATID)) as b  
    where a.PATID=b._PATID and a.SEX='1' ;  
quit ;
```

■ Joint with macro-variable

```
/* Joint with Macro-variable */  
❏ proc sql noprint ;  
    select quote(PATID) into :patid_cod separated by ','  
    from rdata.PATIENT1  
    where sex="1" ;  
quit ;  
  
❏ data PATTHER_MACROVAR ;  
    set rdata.ther ;  
    where PATID in (&patid_cod) ;  
run ;
```

Save your time and the time of the others

■ Joint with a format

```
/* Joint with format */  
□ data PATID_COD (keep=fmtname start end label) ;  
  set rdata.PATIENT1 end=fin ;  
  
  if sex='1' then do ;  
    fmtname="$pat_cod" ;  
    start=PATID ;  
    end=PATID ;  
    label=PATID ;  
    output ;  
  end ;  
  
  else do ;  
    fmtname="$pat_cod" ;  
    start=PATID ;  
    end=PATID ;  
    label="Other" ;  
    output ;  
  end ;  
run ;  
  
□ proc format library=work.formats cntlin=work.PATID_COD ;  
  quit ;  
  
□ data PATTER_FORMAT ;  
  set rdata.ther ;  
  where put(PATID, $pat_cod.) ne "Other" ;  
run ;
```

Save your time and the time of the others

■ Joint with Hash method

```
/* Joint with Hash method */  
proc sql noprint ;  
    select NAME into :var_pat separated by ' '  
    from sashelp.vcolumn  
    where LIBNAME="RDATA" and MEMNAME="PATIENT1" ;  
  
    select quote(strip(NAME)) into :var_the separated by ','  
    from sashelp.vcolumn  
    where LIBNAME="RDATA" and MEMNAME="THER" ;  
quit ;  
  
data PATTHER_HASH ;  
    declare hash tab_hash() ;  
    rc=tab_hash.DefineKey("PATID") ;  
    rc =tab_hash.DefineData (&var_the);  
    rc=tab_hash.DefineDone() ;  
  
    do until (eof1) ;  
        set rdata.PATIENT1 (where=(SEX='1')) end=eof1 ;  
        rc=tab_hash.add() ;  
        drop &var_pat ;  
    end ;  
  
    do until (eof2) ;  
        set rdata.THER end=eof2 ;  
        if tab_hash.find()=0 then output;  
    end ;  
run ;
```

Save your time and the time of the others

■ Joint with Index

```
❏ proc sql ;  
    create index PATID on rdata.PATIENT1 (PATID);  
    create index PATID on rdata.THER (PATID);  
  
    create table PATTHER_INDEX (drop=SEX _PATID)  
    as select a.*, b.*  
    from rdata.PATIENT1 (keep=PATID SEX) as a, rdata.THER (rename=(PATID=_PATID)) as b  
    where a.PATID=b._PATID and a.SEX='1' ;  
quit ;
```

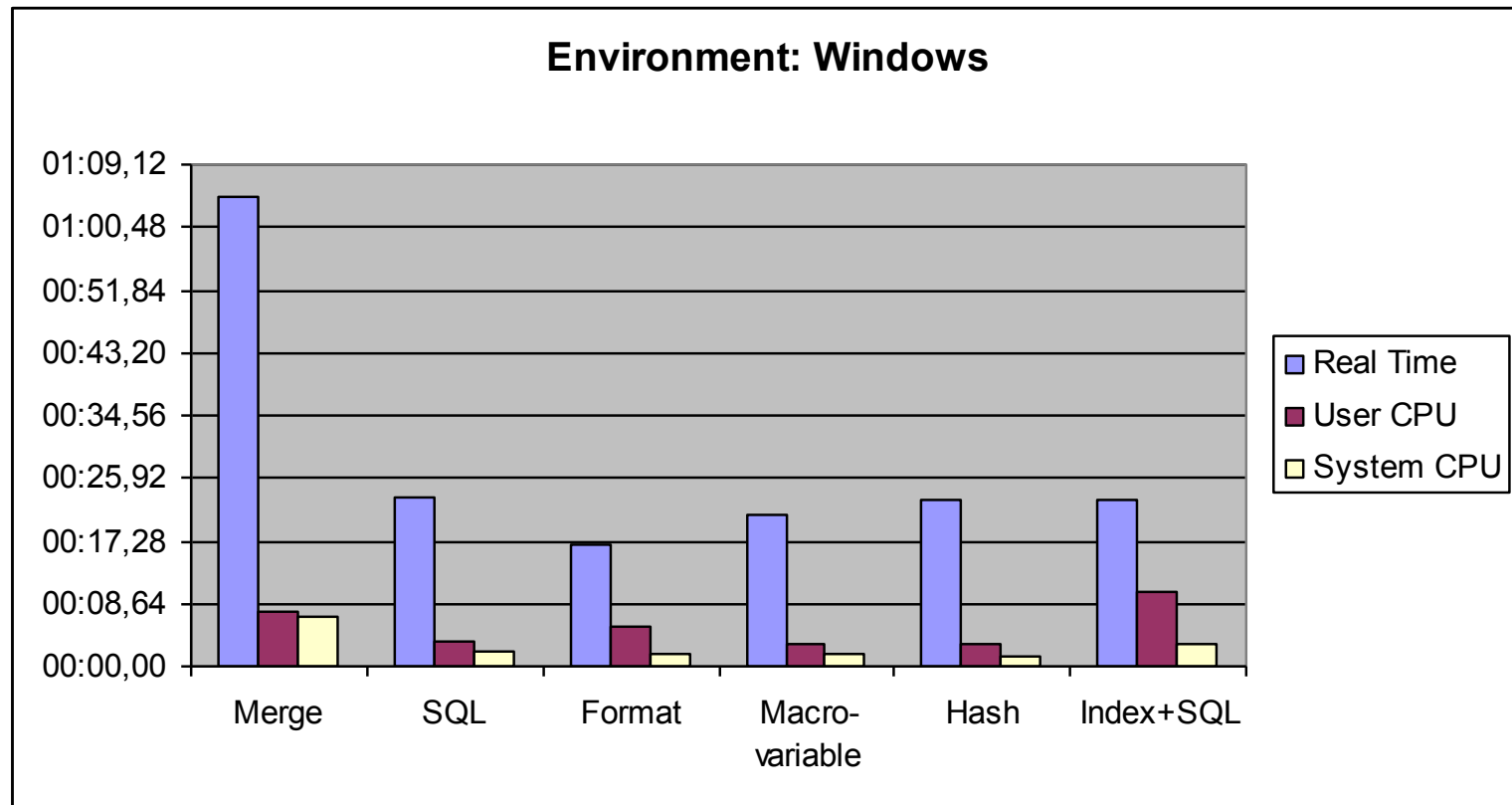
Save your time and the time of the others

■ Performances of all methods: Windows

Environment: WINDOWS	Steps	Real Time	User CPU	System CPU
Merge	1st SORT	00:01,75	00:00,00	00:00,04
	2nd SORT	00:52,60	00:05,21	00:05,31
	Merge	00:10,21	00:02,16	00:01,62
	Total	01:04,56	00:07,37	00:06,97
SQL		00:23,27	00:03,32	00:01,95
Format	Data step	00:00,06	00:00,01	00:00,03
	PROC FORMAT	00:00,36	00:00,03	00:00,03
	Data step	00:16,35	00:05,56	00:01,55
	Total	00:16,73	00:05,61	00:01,60
Macro-variable	PROC SQL	00:02,32	00:00,01	00:00,01
	Data step	00:18,44	00:03,06	00:01,83
	Total	00:20,76	00:03,07	00:01,84
Hash	PROC SQL	00:00,51	00:00,00	00:00,01
	Data step	00:22,52	00:03,18	00:01,49
	Total	00:23,03	00:03,18	00:01,50
Index+SQL		00:22,88	00:10,40	00:03,15

Save your time and the time of the others

■ Performances of all methods: Windows



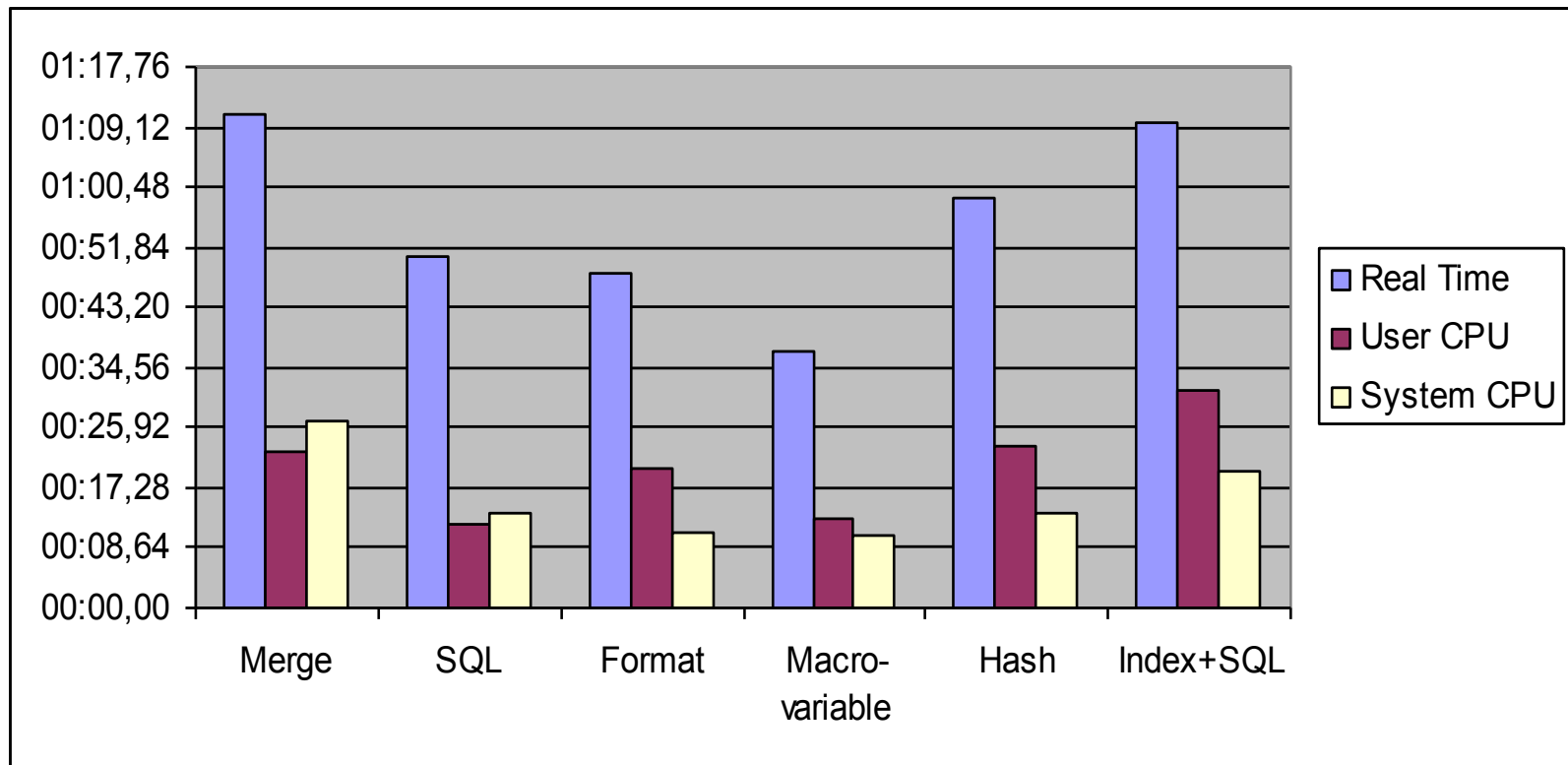
Save your time and the time of the others

■ Performances of all methods: Unix

Environment: Unix	Steps	Real Time	User CPU	System CPU
Merge	1st SORT	00:01,27	00:00,74	00:00,25
	2nd SORT	00:36,34	00:13,26	00:17,09
	Merge	00:33,20	00:08,25	00:09,43
	Total	01:10,81	00:22,25	00:26,77
SQL		00:50,33	00:12,01	00:13,55
Format	Data step	00:01,25	00:00,02	00:00,25
	PROC FORMAT	00:00,23	00:00,09	00:00,04
	Data step	00:46,54	00:19,91	00:10,50
	Total	00:48,01	00:20,02	00:10,79
Macro-variable	PROC SQL	00:01,16	00:00,03	00:00,18
	Data step	00:35,89	00:12,73	00:10,18
	Total	00:37,05	00:12,76	00:10,36
Hash	PROC SQL	00:00,03	00:00,00	00:00,01
	Data step	00:58,83	00:23,20	00:13,51
	Total	00:58,86	00:23,20	00:13,51
Index+SQL		01:09,64	00:31,42	00:19,45

Save your time and the time of the others

■ Performances of all methods: Unix



Save the space

- To fit the length of the variables as soon as possible (according to the CDISC recommendations)
 - Not too small to not truncate the information
 - But not too large to not take space unnecessarily
 - Default length of the numeric variables is 8 but, according to the kind of information, can be less

Save the space

```
/* Length of numeric variables=3 */  
data rdata.patient2 ;  
  length PRSCQTY PRSCDAYS REGSTAT prsc_nbt prsc_qtyt exposure statusb status1 statusbeq status1eq  
        statusb2 status2 status2eq statusb2eq statusb3 status3 statusb3eq status3eq 3 ;  
set rdata.patient1 ;  
run ;
```

Windows

	Patient1	19.7MB	Table
	Patient2	19.7MB	Table

Unix

	Patient1	19.3MB	Table
	Patient2	18.0MB	Table

Save the space

■ Compress the datasets

■ Compression done on the character variables

- Important gain of space if lot of character variables
- But reversed effect if number of character variables not important enough due to creation of a flag at each record

■ Decompression at each use of the dataset

- Decompression takes time
- To assess the benefit of the gain of space versus the time for decompression

Save the space

```
data rdata.patient_comp (compress=Yes reuse=yes) ;  
  set rdata.patient1 ;  
run ;
```

 Patient1	19.7MB	Table
 Patient2	19.7MB	Table
 Patient_comp	8.7MB	Table

Save the space

```
data rdata.ther_comp (compress=Yes reuse=yes) ;  
    set rdata.ther ;  
run ;
```

A PATID	Text	6	
A DRUGCODE	Text	8	
A BNF	Text	8	
12>. firstdrug	Number	8	DDMMYY10.
12>. lastdrug	Number	8	DDMMYY10.



Ther

388.3MB Table



Ther_comp

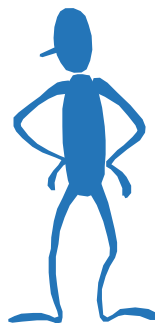
434.3MB Table

Conclusion

- A lot of tricks exist to optimize the programs
- But, if each trick improves an aspect (CPU time or Space), it can have negative effect on the other aspect
- An assessment must be done prior to the choice in accordance with the priorities of the bussiness and with technology already in place

Thank you for your attention !!

Questions ? ? ?



Contact:

claudio.guyot@sanofi-aventis.com

(33) 1 60 49 72 72