

Merge versus Update Statement

Park, Hun Myoung
Indiana University

1. Introduction

SAS has two statements to join observations from two or more SAS data sets into single observations. The MERGE statement supports one-to-one merging and match-merging. One-to-one merging combines observations by overwriting the master data set with secondary data sets. Match-merging combines observations in a new data set according to the values of a common variable. The UPDATE statement updates a master data set by applying a transaction data set.

The MERGE automatically replaces existing values in the master data set with missing values in the second data set, whereas the UPDATE by default does not. The former has the BY statement as an option, whereas the latter requires the BY statement.

	BY Statement	# 2 nd data sets	Missing in the 2 nd data sets
One-to-One merging	No	Multiple	Overwriting
Match-merging	Required	Multiple	Overwriting
Update	Required	One	NOMISSINGCHECK

2. Sample Data Sets

Here are two sample data sets to illustrate the differences between the two statements. Note that some observations have missing values in some variables.

```
DATA master;
INPUT name $ id history math $ physics @@ ;
DATALINES;
Park 87 89 master 98
Koo 87 94 master .
Kim 85 95 master 91
Choi 89 . master 100
; RUN;
```

```
PROC SORT; BY name id; RUN;
PROC PRINT; RUN;
```

Obs	name	id	history	math	physics
1	Choi	89	.	master	100
2	Kim	85	95	master	91
3	Koo	87	94	master	.
4	Park	87	89	master	98

For the sake of this study, the second data set has a duplicacy, Koo 87.

```

DATA trans;
INPUT  name $ id history math $ stat @@ ;
DATALINES;
Won  87  96 slave  83
Park 87  95 slave  .
Koo  87  99      . 89
Kim  85  . slave  97
Koo  87  77 slave 100
; RUN;

```

```

PROC SORT; BY name id; RUN;
PROC PRINT; RUN;

```

Obs	name	id	history	math	stat
1	Kim	85	.	slave	97
2	Koo	87	99		89
3	Koo	87	77	slave	100
4	Park	87	95	slave	.
5	Won	87	96	slave	83

3. One-to-one Merging

One-to-one merging, the MERGE statement without the BY statement, ignores the uniqueness and order of an observation. It just puts data sets together in the mechanical sense. Consequently, this type of merging may end up with a messy result, especially when data sets are not well organized.

The following examples illustrate how master and secondary data sets play roles in merging. The red texts represent information overwritten by the secondary data sets.

```

DATA merge1;
MERGE master trans;
RUN;
PROC PRINT; RUN;

```

Obs	name	id	history	math	physics	stat
1	Kim	85	.	slave	100	97
2	Koo	87	99		91	89
3	Koo	87	77	slave	.	100
4	Park	87	95	slave	98	.
5	Won	87	96	slave	.	83

```

DATA merge2;
MERGE trans master;
RUN;
PROC PRINT; RUN;

```

Obs	name	id	history	math	stat	physics
1	Choi	89	.	master	97	100
2	Kim	85	95	master	89	91
3	Koo	87	94	master	100	.
4	Park	87	89	master	.	98
5	Won	87	96	slave	83	.

4. Match-Merging

Match-merging requires common variables in order to distinguish one observation from others. The variables are listed in the BY statement. Of course, the data sets must be sorted by the variables that are shared by data sets.

```
DATA merge3;
  MERGE master trans;
  BY name id;
RUN;
PROC PRINT; RUN;
```

Obs	name	id	history	math	physics	stat
1	Choi	89	.	master	100	.
2	Kim	85	.	slave	91	97
3	Koo	87	99	.	.	89
4	Koo	87	77	slave	.	100
5	Park	87	95	slave	98	.
6	Won	87	96	slave	.	83

The history value for Kim (85) was replaced by missing in the secondary data set. Note that the duplicacy in the secondary data set is listed twice.

```
DATA merge4;
  MERGE trans master;
  BY name id;
RUN;
PROC PRINT; RUN;
```

Obs	name	id	history	math	stat	physics
1	Choi	89	.	master	.	100
2	Kim	85	95	master	97	91
3	Koo	87	94	master	89	.
4	Koo	87	77	slave	100	.
5	Park	87	89	master	.	98
6	Won	87	96	slave	83	.

Note that the second Koo (87) remains unchanged; only first duplicate observation is overwritten by the secondary data set.

5. Update a Data Set

The UPDATE requires common variables that are specified in the BY statement. The following output has one more observation than that of the MERGE statement.

```
DATA update1;
  UPDATE master trans;
  BY name id;
RUN;
PROC PRINT; RUN;
```

Obs	name	id	history	math	physics	stat
1	Choi	89	.	master	100	.
2	Kim	85	95	slave	91	97

3	Koo	87	77	slave	.	100
4	Park	87	95	slave	98	.
5	Won	87	96	slave	.	83

Kim's history remains unchanged since the secondary data set has missing for that observation. Koo's information is updated with his last observation in the second data set.

What if you turn on the UPDATEMODE=NOMISSINGCHECK option? Look at the Kim's history score that was replaced by missing, a value of the secondary data set.

```
DATA update2;
  UPDATE master trans UPDATEMODE=NOMISSINGCHECK;
  BY name id;
RUN;
PROC PRINT; RUN;
```

Obs	name	id	history	math	physics	stat
1	Choi	89	.	master	100	.
2	Kim	85	.	slave	91	97
3	Koo	87	77	slave	.	100
4	Park	87	95	slave	98	.
5	Won	87	96	slave	.	83

6. Append Observations

The SET statement appends observations in the secondary data set to the master data set. It stacks up the master and secondary data sets.

```
DATA append;
  SET master trans;
RUN;
PROC PRINT; RUN;
```

Obs	name	id	history	math	physics	stat
1	Choi	89	.	master	100	.
2	Kim	85	95	master	91	.
3	Koo	87	94	master	.	.
4	Park	87	89	master	98	.
5	Kim	85	.	slave	.	97
6	Koo	87	99	.	.	89
7	Koo	87	77	slave	.	100
8	Park	87	95	slave	.	.
9	Won	87	96	slave	.	83