

Le RETAIN expliqué à ma fille

L'intrigante instruction RETAIN devient parfois indispensable dans une étape Data. Quelques mots pour vous aider à savoir si, comme l'ami Ricoré, elle vient toujours au bon moment.

Que se passe-t-il dans une étape Data ?

Lors de la lecture de données dans une étape Data (que ce soit avec une instruction SET, MERGE ou INFILE), les informations qui composent les observations sont stockées de manière temporaire dans un espace appelé Vecteur de Travail (ou PDV dans les documents en anglais).

Le traitement des données s'effectue ainsi :

1. copie des informations lues dans le vecteur de travail
2. opérations (calculs) demandés (création de nouvelles variables, modifications de variables existantes)
3. écriture du contenu du vecteur de travail dans la table créée (éventuellement sous condition, en présence de WHERE ou de IF ... THEN OUTPUT¹)
4. suppression des informations contenues dans le vecteur de travail
5. retour à l'étape 1.

Que fait l'instruction RETAIN ?

```
DATA tableCréee ;  
  SET|MERGE|INFILE ... ;  
  RETAIN nom(s)Variable(s) ;  
  ... ;  
RUN ;
```

RETAIN permet de « protéger » la valeur d'une variable dans le vecteur de travail, c'est-à-dire de la dispenser de l'étape 4 dans l'enchaînement ci-dessus. On peut donc accéder, d'une observation à l'autre, à la valeur de la variable telle qu'elle était pour l'observation précédente. RETAIN s'applique surtout à de nouvelles variables ; pour les données lues dans une table SAS, la fonction LAG permet de récupérer plus aisément la valeur à l'observation précédente.

Cas 1 : incrémentation

```
DATA work.class ;  
  SET sashelp.class ;  
  RETAIN numero_eleve ;  
  IF _N_ = 1 THEN numero_eleve = 0 ;  
  numero_eleve = numero_eleve + 1 ;  
RUN ;
```

Dans ce cas, on souhaite incrémenter une variable (NUMERO_ELEVE) à chaque observation. Sans RETAIN, on perdrait la valeur précédente de NUMERO_ELEVE en changeant d'observation. Le résultat serait donc, sans RETAIN : à la 1^{ère} observation, NUMERO_ELEVE

¹ Qui peut s'abrégier en IF condition ; avec le même effet que IF condition THEN OUTPUT ; — si la condition est vérifiée, l'observation est copiée dans la table en sortie

vaudrait 1, et à toutes les autres observations, cette variable serait manquante (car valeur manquante + 1 → valeur manquante).

On peut simplifier ce programme de deux manières : en premier lieu, l'instruction RETAIN permet également de proposer une valeur d'initialisation pour la variable.

```
DATA work.class ;
  SET sashelp.class ;
  RETAIN numero_eleve 0 ;
  numero_eleve = numero_eleve + 1 ;
RUN ;
```

On peut également, dans le cas d'une addition, utiliser la syntaxe SAS simplifiée suivante. Le RETAIN est alors implicite, de même que l'initialisation à zéro. Aucune autre opération que l'addition n'est supportée dans cette syntaxe résumée.

```
DATA work.class ;
  SET sashelp.class ;
  numero_eleve + 1 ;
RUN ;
```

Cas 2 : conserver la mémoire d'une valeur

Pour chaque client, on a l'historique de ses achats. On veut calculer, pour chaque client, l'ancienneté de ses achats par rapport au tout premier qu'il a effectué.

```
PROC SORT DATA = work.ventes ;
  BY numClient dateAchat ;
RUN ;
DATA work.ventes ;
  SET work.ventes ;
  BY numClient ;
  RETAIN premierAchat ;
  FORMAT premierAchat DDMMYY10. ;
  IF FIRST.numClient THEN premierAchat = dateAchat ;
  delai = dateAchat - premierAchat ;
RUN ;
```

Ici, RETAIN permet de conserver la mémoire de la première date d'achat par client (repéré par un booléen FIRST disponible grâce au SET ... BY) en la stockant dans une nouvelle variable protégée des nettoyages du vecteur de travail.

numClient	dateAchat	premierAchat	delai
1	14/06/1999	14/06/1999	0
1	25/07/1999	14/06/1999	41
1	25/10/1999	14/06/1999	133
1	06/07/2001	14/06/1999	753
2	01/06/2000	01/06/2000	0
2	19/12/2000	01/06/2000	201