PRÉSENTATION DU LANGAGE DS2 ET DE SES APPLICATIONS

Caractéristiques :
Catégories : Base SAS®, SAS ® Embedded Processes
OS : Windows, Unix
Version : SAS® 9.2 TS2M3, SAS® 9.3
Vérifié en avril 2013

Depuis SAS 9.2 TS2M3, SAS propose un nouveau langage de programmation permettant de créer et gérer des tables SAS : le DS2 (« Data Step 2 »).
Ces nouveautés permettent aux développeurs spécialistes des langages « orientés objet » et aux administrateurs de bases de données de développer des programmes SAS rapidement grâce à une syntaxe adaptée à leurs profils (langage SQL, syntaxe Objet…). Elles se veulent être un complément aux fonctionnalités existantes de SAS concernant le SQL et le Java.

Le langage DS2, dont l'un des atouts essentiels est la capacité à exécuter des traitements massivement parallélisés, est aujourd'hui utilisé dans les logiciels SAS dédiés à l'architecture « High Performance Analytics » tels que SAS Visual Analytics, les logiciels SAS In-Database (processus embarqués dans les bases de données comme Teradata, DB2, Oracle), Hadoop ainsi que dans Dataflux Federation Server.
Ce langage se veut plus proche des données des SGBDs accédés, notamment par la fourniture de types de données proches de ceux qu'on retrouve dans ces bases.

Table des matières

Introduction 1
Une nouvelle syntaxe 2
Une étape DATA au sein d'une procédure 2
Implémentation de méthodes par défaut au sein de l'étape DATA 2
Variables et périmètre des variables 3
Déclaration des variables 3
Périmètre des variables et des objets 4
Accès à des variables d'un périmètre différent 4
Nouveautés de DS2 par rapport au langage SAS/Base habituel 5
Gestion des NULL / Valeurs manquantes 5
Packages 5
Langage SQL 6
Threads 7
Conclusion 7

Introduction

Les éléments que vous trouverez dans cet article vous permettront de découvrir simplement les fonctionnalités offertes par le langage DS2 dans le cadre d'exemples concrets.

Le langage DS2 est l'intersection entre la syntaxe de l'étape DATA et d'autres types de syntaxe tels que le SQL ANSI, l'utilisation de syntaxes « orientées objet » (utilisation d'objets nommés « méthodes » et « packages »).

Du fait que le DS2 s'adresse à un public plus large que celui des développeurs SAS, le vocabulaire relatif aux données s'aligne sur celui que nous trouvons dans le monde des SGBD ainsi,

- un dataset sera une table,
- une observation sera une ligne
- et une variable sera une colonne.

Le langage DS2 s'appuie principalement sur deux procédures SAS :
- Proc DS2
- Proc DSTRANS

La procédure DS2 permet d'implémenter des programmes SAS ayant pour objectif de manipuler des données avec cette nouvelle syntaxe.
La procédure DSTRANS est, elle, essentiellement dédiée à la traduction de langage Base standard (étapes DATA) en syntaxe DS2. Toutes les fonctionnalités de l'étape DATA ne sont pas supportées car la procédure DSTRANS est essentiellement consacrée à la génération de code dédié au scoring dans SAS® Enterprise Miner™.

Une nouvelle syntaxe

Une étape DATA au sein d'une procédure

Une étape DATA doit être encapsulée dans une procédure DS2 dont la syntaxe est la suivante :

```
Proc ds2 ;
Data nomtable ;
…
Enddata;
Run ;
Quit;
```

Il est important de noter la présence d'un nouveau mot clé terminant l'étape data : « ENDDATA; » qui précède le traditionnel « run ; ».
La fin de la procédure DS2 est assurée par l'instruction « quit ; ».

Implémentation de méthodes par défaut au sein de l'étape DATA

Dans une étape Data DS2, il est nécessaire de créer 3 méthodes permettant de déclarer des variables, d'écrire des programmes SAS ou des requêtes SQL. Le principe se rapproche de celui des fonctions.

- La méthode INIT()
  Elle permet de définir des variables ou du code SAS pour préparer la création de la table SAS.

- La méthode RUN()
  Elle permet d'écrire le corps du programme DS2 nécessaire à la création de la table de données.
  Important : c'est dans la méthode RUN() que la boucle implicite est réalisée sur la ou les lignes de la (ou des) table(s) de données en entrée.

- La méthode TERM()
  Cette méthode permet d'exécuter du code post traitement sur la table, si cela est nécessaire.

Si les méthodes ne sont pas explicitement écrites par le développeur, le compilateur va les ajouter de manière implicite (vides) comme ceci :

```
Proc ds2 ;
Data test;
        Method INIT() ;
        End ;
        Method RUN() ;
        End ;
        Method TERM() ;
        End ;
Enddata ;
Run;
Quit ;
```

Si aucun nom de table n'est spécifié, alors la PROC DS2 ne génèrera pas de table de sortie. Il en résultera donc une sortie vers l'output au même titre qu'une PROC PRINT sur le périphérique ODS par défaut.

Exemple :

```
Proc ds2 ;
Data;  /* cette étape Data enverra seulement les données sur l'output et pas dans une table SAS */
        Method INIT() ;
        End ;
        Method RUN() ;
        End ;
        Method TERM() ;
        End ;
Enddata ;
Run;
Quit ;
```

Variables et périmètre des variables

Le langage DS2 permet de déclarer des variables de plusieurs types, ce qui est nouveau dans SAS. Dans l'étape Data originelle, les variables sont de type numérique ou alphanumérique (caractères). La syntaxe DS2 permet de créer des types similaires à ceux rencontrés dans les systèmes actuels de gestion des bases de données : char(n), varchar(n), double, bigint, binary, date, float, date, timestamp…
Leur manipulation est implicitement traduite en format SAS lorsque les tables SAS natives sont créées.

- Les types basés sur du caractère sont stockés avec la forme « char(N) » (N étant le nombre de caractères).
- Les variables DS2 numériques sont traduites dans le format « double » (nombre codé sur 64 bits) dans la table SAS résultat.

Déclaration des variables

Le langage DS2 diffère de l'étape DATA en ce qui concerne la syntaxe de déclaration des variables. En DS2, la déclaration des variables se fait de la manière suivante :

```
dcl type nomvariable1 nomvariable2 ;
        ou
declare type nomvariable1 nomvariable2 ;
```

Toutefois, il est toujours possible de déclarer des variables à la volée.
Les variables peuvent être déclarées dans une méthode ou dans le corps de l'étape DATA de la procédure DS2.

Exemple :

```
method init() ;
        var = 'test' ;
end ;
```

Le DS2 identifiera la variable « var » par rapport à son contenu et sera implicitement typée en char(4). Une variable déclarée sans type et non initialisée par une valeur sera de type DOUBLE (numérique).

Important : au sein d'une méthode, une variable non explicitement déclarée par « dcl » ou « declare » sera automatiquement GLOBALE !
La possibilité de déclarer des variables dans différentes sections du programme DS2 implique une gestion du périmètre des variables et de leur durée de vie ainsi que de leur contenu.

Périmètre des variables et des objets

Le périmètre des variables dépend de l'endroit où elles sont déclarées. Si la déclaration survient avant la "méthode init() ;", alors la variable sera globale et aura une durée de vie équivalente à celle du programme.
Si la déclaration survient dans une méthode "(init(), run(), term())", alors la variable sera locale à la méthode et aura la durée de vie de la méthode.
Seules les variables globales sont utilisées pour l'output dans les tables SAS. Les variables locales aux méthodes sont des variables temporaires et de travail. Il est donc inutile de les supprimer de l'output car cela est fait automatiquement.

Note : si la variable est initialisée dans une méthode, le contenu de cette variable ne sera accessible par une autre méthode qu'avec l'emploi d'une expression « THIS » expliquée ci-après.

Accès à des variables d'un périmètre différent

L'expression « This » permet d'accéder à une variable globale au sein d'une méthode si la variable globale existe. Cette variable globale peut être déclarée en amont dans le programme DS2 ou dans une autre méthode. Si la variable appelée par l'expression « This » n'a pas été préalablement déclarée, cette variable est déclarée à la volée et devient donc une variable globale.

Exemple d'un programme réalisant la somme des valeurs des variables a et b :

```
proc ds2;
data _null_ ;
        method init();
                a=1; /* a est une variable globale car non déclarée avec l'instruction dcl */
                b=2; /* b est une variable globale car non déclarée avec l'instruction dcl */
        end;
        method somme(double x, double y) returns double;
                        return  x+ y;
        end;
        method run();
                dcl int a; /* a est une variable locale à la méthode run() */
                dcl int b; /* a est une variable locale à la méthode run() */
                a= 13;
                b= 3;
                S=somme(this.a, this.b);  /* "this.a" et "this.b" référencent les variables a et b globales
declarées dans la method init() et non les variables a et b déclarées dans la méthode courante run() */
                put S=;
        end;
enddata;
run;
quit;
```

L'expression "**S=somme(this.a, this.b);"** de l'exemple précédent appelle les valeurs des variables globales **a** (dont la valeur =1) et **b** (dont la valeur =2). Le résultat de cette somme est S=3.

Si nous remplaçons :

S=somme(this.a, this.b);

par

S=somme(a, b);

alors le résultat sera **S=16** car les valeurs prises sont locales à la méthode dans laquelle est passé l'appel à la fonction «somme ».

Différences de DS2 par rapport au langage SAS/Base habituel

Gestion des NULL / Valeurs manquantes

La gestion des valeurs manquantes par SAS n'est pas conforme aux standards ANSI SQL par défaut. Il existe des différences importantes entre la gestion des valeurs manquantes (SAS) et celle des valeurs NULL (SGBD).

De ce fait, il a donc été implémenté au niveau de la procédure DS2 la possibilité de choisir le mode de gestion des valeurs inexistantes :

- Mode SAS : les valeurs absentes sont des valeurs manquantes (typées mais sans valeur)
- Mode ANSI : les valeurs absentes ne sont pas typées

Le mode ANSI est adapté aux SGBD implémentant le standard ANSI SQL tels qu'Oracle et DB2. Par défaut le mode SAS est actif. Le mode ANSI est une option de la PROC DS2.

Une confusion entre ces modes peut conduire à des résultats de requêtes erronés, voire des pertes de données.

Dans une clause WHERE ou HAVING par exemple, le mode « SAS » interprète les NULL d'une base de données comme des valeurs manquantes.
Le mode « SAS » considère également les blancs (espaces seuls) comme des valeurs manquantes.
En mode « ANSI », un blanc (espace seul) est considéré comme un espace.

Selon le mode utilisé, la comparaison de valeurs manquantes/nulles peut donc produire des résultats erronés.

Packages

Un package est une collection de variables et méthodes dans des programmes DS2 ou des threads (description ci-après).
Un package peut être stocké dans un catalogue SAS.
Il est chargé en mémoire lors de sa déclaration (instanciation).
Il s'agit d'un objet qui peut être réutilisé et paramétré de manière différente dans un autre programme DS2.
Les packages stockés sont sécurisés avec un cryptage AES (Advanced Encryption Standard).

Il existe deux familles de packages : les packages définis par l'utilisateur ou les packages prédéfinis dans SAS.
Parmi les packages prédéfinis, voici une présentation du package prédéfini : HASH.
Ce type de package permet de stocker aisément des données et d'y accéder simplement à l'aide de clés uniques. Selon le nombre de clés et la taille de la table de données à parcourir, les temps d'accès à la donnée (processus d'exploration) sont généralement bien plus rapides que l'accès à la même donnée stockée dans un Array (tableau).

Dans l'exemple suivant, l'objectif est de montrer comment extraire des données du package Hash avec DS2 sur la table d'exemple "sashelp.cars".
Pour cela, il est nécessaire de créer au préalable une petite table de données qui liste les véhicules européens et de calculer le pourcentage de remise par modèle.
L'objectif est de générer un listing de véhicules communs entre la table d'origine sashelp.cars et ceux de la liste de véhicules européens. Ce listing contiendra également le pourcentage de remise de chaque véhicule.

A noter que dans cet exemple, la « petite » table (véhicules européens) est extraite de la table « sashelp.cars ». Cela induit que tous les modèles vont correspondre et la table finale aura le même nombre de lignes que la « petite table » (véhicules européens).

```
/* copie de la table sashelp.cars */
data cars ;
set sashelp.cars;
run;

/* voitures européennes par marque avec pourcentage de remise */
data europe (keep=make model horsepower discount) ;
set work.cars;
where origin='Europe' ;
format discount 3.2;
discount=(1-(invoice / MSRP))*100;
run;

proc ds2;
package hash /ext='hash'; run;
data highdisc (overwrite=yes);
        dcl package hash allcars(0, 'cars');
        dcl char(13) Make;
        dcl char(40) Model;
        dcl double Horsepower  ;
method init();
        allcars.defineKey('Model');
        allcars.defineData('Make');
        allcars.defineData('Model');
        allcars.defineData('Horsepower');
        allcars.defineDone();
end;
method run();
        /* itération sur la petite table issue de la requête SQL
   pour trouver les véhicules communs avec la table "cars" et les classer par pourcentage de
remise */
        set {select model,max(discount) as discount from europe  group by discount, model
order by discount desc};
        if allcars.find()=0  then output; /* comme la table Europe est un sous-ensemble de cars,
toutes les lignes de “Europe” correspondront aux lignes du package hash, donc la table de sortie
fera le même nombre de lignes que la table europe */
end;
enddata;
run;
quit;
```

Langage SQL

L’exemple ci-dessous montre qu’il est possible d’utiliser du langage SQL au sein de la procédure DS2. Cette requête liste les véhicules de la table cars (préalablement copiée dans la work) ayant la chaîne « erc » dans leur marque.

Note : la requête SQL doit être positionnée entre accolades {…} et ne peut pas contenir le nom de la table incluant le *libref* si celui-ci est sashelp (sashelp.table). Cette syntaxe est notamment dédiée à la soumission de code SQL vers des systèmes de base de données tiers.

```
proc ds2;
data sql ;
method run();
        set { select make, horsepower, model from cars
        where make like '%erc%'};
end;
enddata;
run;
quit;
```

Threads

A l'instar du package, un thread est une liste globale d'instructions et de méthodes balisées entre les mots clés : « thread ; » et « endthread ; ».
Il peut être composé d'autres threads.

A la différence d'un package, un thread est compilé et stocké dans une table. Il est chargé en mémoire lors de sa déclaration (instanciation). Il s'agit également d'un objet qui peut être réutilisé et paramétré de manière différente dans un autre programme DS2.

Le langage DS2 donne la possibilité au développeur de créer et de gérer plusieurs processus simultanés. Ce mode est dit « multi-threadé ».
L'intérêt majeur de cet aspect est de pouvoir manipuler les données en mémoire dans plusieurs processus parallèles indépendants afin d'accélérer la conception de la table résultat.

Un programme DS2 peut être lancé en tant que thread ou peut lancer des threads, contrairement à l'étape DATA qui fonctionne de manière séquentielle. Cette fonctionnalité peut être utilisée pour améliorer les performances de constitution d'une table de résultat par différents threads. Il est à noter que les performances dépendent de la synchronisation de ces threads et du temps d'exécution du thread le plus lent.

Enfin, l'utilisation des threads peut conduire à une sortie dans le désordre, cela dépend du temps d'exécution de chaque thread. Le résultat peut donc être entrelacé et impliquer un risque sur la cohérence des données.

Conclusion

DS2 est un nouveau langage dans SAS qui mêle la syntaxe du langage SAS usuel avec des langages complémentaires (SQL, orienté objet).
Il permet notamment aux développeurs de pouvoir interagir avec les nœuds SAS installés au sein des bases de données (Oracle, DB2, Teradata) et ainsi de bénéficier d'excellentes performances dans la manipulation de volumes importants de données (Big Data).

DS2 permet également d'utiliser le langage macro (Macro-variables) pour faire de la substitution de texte contextuelle et ainsi de rendre les programmes plus génériques.
Pour éviter les confusions et l'usage de guillemets, le compilateur DS2 supporte la syntaxe :%TSLIT(&charvariable). La fonction macro TSLIT permet de gérer simplement les macro-variables sans se soucier d'ajouter les guillemets.

Si vous souhaitez aller plus loin, vous pouvez vous référer aux documentations publiées sur le site web de SAS :

- Guide de démarrage
- Guide complet

Marc Avezac
Consultant Support Clients SAS France