



SOMMAIRE

Sommaire	1
News	2
Les derniers correctifs	2
Le code hash	3
Présentation de SAS® AppDev Studio 3.2	7
Les nouveautes SPDS 4.4	9

NEWS

Quel est le support de Microsoft Windows Vista™ ? <http://support.sas.com/techsup/pcn/vista.html>

SAS Forum Tech

SAS France a pris en compte les demandes de ses utilisateurs et propose un journée technique Mardi 5 juin 2007 à Grégy-sur-Yerres (77).

<http://www.sas.com/offices/europe/france/sasforum/sasftech.html>

Le bundle E9BX01 est disponible pour le module SAS/Base de SAS 9.1.3 service pack 4. Il regroupe les correctifs mis à disposition entre septembre 2006 et janvier 2007.

<http://ftp.sas.com/techsup/download/hotfix/ve9bx.html>

LES DERNIERS CORRECTIFS

- [Liste des correctifs](#) mis en ligne depuis le mois de novembre sur le Customer Support Center.
- [Le site des correctifs US](#)
- [Les correctifs spécifiques aux produits traduits en Français](#)

LE CODE HASH

Le code hash ou le « hashing » est disponible depuis SAS®9. Il propose des méthodes rapides et efficaces pour stocker, chercher et plus généralement manipuler des données dans des tables basées sur des clés d'identification.



Caractéristiques :

Catégories : SAS/Base

OS : Tous

Version : SAS®9

Vérifié le 01/03/2007

Le principe :

A première vue, le code hash semble plus compliqué à appréhender que le langage SAS. Cependant cela n'est qu'une illusion. Un moyen simple de comprendre la logique du code hash est de visualiser l'objet hash comme étant une table contenue en mémoire avec des lignes et des colonnes (ou des variables et des enregistrements). Pour la programmation, les seules connaissances dont vous aurez besoin sont de savoir créer une table, la définir, la remplir et y accéder. Des commandes sont prédéfinies à cet effet. Généralement la syntaxe des commandes se construit de la manière suivante :

Nom_objet_Hash.Nom_commande(<paramètre>) ;

Nous allons voir au travers d'un exemple les quatre commandes principales nécessaires pour utiliser un objet hash.

Le but de cet exemple est d'extraire certaines données d'une table (*table1*) en fonction d'une seconde (*table2*) en utilisant un objet hash.

table1

	keyvar	var1	var2	var3	var4	var5	var6	var7	var8
1	1	11	.	11	11	11	11	11	11
2	2	22	22	22	22	22	22	22	22
3	3	.	33	.	33	33	33	.	33
4	4	.	.	44	.	.	44	.	44
5	5	55	55	55	55	55	55	55	.
6	6	.	66

table2

	keyvar
1	1
2	3
3	6
4	7

Nous souhaitons obtenir une troisième table (*extraction*) qui contiendra uniquement les enregistrements des variables *keyvar*, *var1*, *var2*, *var3* et *var4* de la *table1* relatifs à ceux contenus dans la *table2*.

table extraction
souhaitée

	keyvar	var1	var2	var3	var4
1	1	11	.	11	11
2	3	.	33	.	33
3	6	.	66	.	.
4	7

L'utilisation du code hash n'est possible qu'au sein d'une étape data. Cela implique certaines limitations qui seront abordées dans la fin de cet article.

1. Créer une table en mémoire (objet hash)

La commande `.Declare()` crée la table, mais à ce stade aucune structure (variables / enregistrements) n'existe encore.

```
data extraction (drop = Cr i);  
    Declare hash mon_objet_hash ( ) ;
```

2. Définir la structure de la table

Dans cet étape nous allons définir la variable identifiante de l'objet hash (*keyvar*) et les variables de données (*var1*, *var2*, *var3* et *var4*) en utilisant les commandes `.DefineKey()` et `.DefineData()`.

```
/*Définir la taille de toutes les variables de la table en mémoire*/  
Length keyvar var1-var4 8 ;  
array var(4);  
/*Définir la variable identifiante*/  
Cr = mon_objet_hash.DefineKey ("keyvar");  
/*Définir les autres variables*/  
Cr = Mon_objet_hash.DefineData ("var1", "var2", "var3", "var4");  
/*Ecriture de la structure de la table*/  
Cr = Mon_objet_hash.DefineDone ( ) ;
```

3. Remplir la table en mémoire

On initialise ensuite les enregistrements de la table en mémoire à partir de la *table1*. Les enregistrements sont lus et ajoutés un par un dans l'objet hash grâce à la commande `.add()`.

```
do until (eof_table1) ;  
    set work.table1 (keep=keyvar var1-var4) end =eof_table1;  
    rc = Mon_objet_hash.add() ;  
end ;
```

4. Accéder aux données de la table

La variable *keyvar* est lue enregistrement par enregistrement à partir de la *table2*. Ensuite grâce à la commande `.find()`, on recherche dans la table en mémoire si celle-ci contient un enregistrement correspondant à la valeur de la variable *keyvar*. Si oui les valeurs correspondantes sont écrites dans le PDV (Program Data Vector).

```
do until (eof_table2) ;  
    set work.table2 end =eof_table2;  
    do i=lbound(var) to hbound(var);  
        var(i) = . ;  
    end;  
    rc = Mon_objet_hash.find() ;  
    output ;  
end ;  
run ;
```

L'instruction `output` écrit le contenu du PDV dans la table en sortie mais ne le vide pas. Il est donc nécessaire à chaque itération de réinitialiser les variables `var(i)` à manquantes.

Note : vous trouverez le programme de cet exemple dans le fichier `exemple.sas`.

Dans quel cas utiliser le code hash ?

- Chercher et trouver des enregistrements à partir d'une clé d'identification
- Trier des tables
- Fusionner des tables

Le but premier du code hash n'est pas de proposer de nouvelles fonctions de traitement mais de les réaliser plus vite. En effet pour retrouver une donnée à partir d'une clé dans une table, les temps d'exécution peuvent être de 2 à 6 fois inférieurs à ceux correspondant à d'autres méthodes de recherche plus classiques. C'est pour quoi il devient d'autant plus intéressant et performant d'utiliser le code hash si l'on travaille avec des tables de taille importante.

Etude de cas : Fusion de table

Dans cet exemple de fusion de deux tables, nous allons comparer les performances entre la méthode classique de fusion (MERGE) et celles utilisant des tables en mémoire grâce au code hash. Les tests sont réalisés sur une machine PC Windows XP pro SP2 à 2Ghz et 2Go de RAM.

Données d'exemple

Dans un premier temps nous allons créer des données d'exemple :

- Une première table work.small d'environ 60 000 enregistrements avec 21 variables (10 Mo)
- Une seconde table work.large d'environ 315 000 enregistrements avec 683 variables (1.8 Go)

Note : Vous trouverez le programme de génération des données d'exemple ainsi que tous les autres programmes utilisés dans le fichier attaché à cet article : comparaison.sas.

Méthode classique

Dans la majorité des cas, pour réaliser une fusion entre 2 tables, on utilise un programme du type suivant :

```
proc sort data=work.small; by keyvar; run;
proc sort data=work.large; by keyvar; run;
data work.match_merge;
    merge work.large (in=a)
          work.small (in=b);
    by keyvar;
    if a;
run;
```

Vous noterez que dans ce cas, il est indispensable de réaliser un tri sur les deux tables à fusionner par rapport à la clé d'identification.

Résultats:

33 min 30 sec en temps réel soit 3 min en temps processeur =
5 min (tri de la table work.small) + 28 min 42 sec (tri de la table work.large) + 5 min 43 sec (étape data avec l'instruction merge)

Méthode avec le « hashing »

Nous allons réaliser la même opération en utilisant les objets de hashing :

```
data work.hash_merge (drop=rc i);

    /* Création d'un objet hash (table) */
    declare hash h_small ();
    /* Définition de l'objet */
    length keyvar smallvar1-smallvar20 8; /*taille des variables*/
    array smallvar(20);
    rc = h_small.DefineKey ( "keyvar" ); /*définition de la clé
d'identification*/
    /* Définition des données*/
    rc = h_small.DefineData ( "smallvar1","smallvar2","smallvar3","smallvar4",
        "smallvar5","smallvar6","smallvar7","smallvar8",
        "smallvar9","smallvar10","smallvar11","smallvar12",
        "smallvar13","smallvar14","smallvar15","smallvar16"
        ,
        "smallvar17","smallvar18","smallvar19","smallvar20"
    );
    rc = h_small.DefineDone ();

    /* Initialisation de la table*/
    do until ( eof_small );
```

```

        set work.small end = eof_small;
        rc = h_small.add ();
end;

/* Fusion des tables */
do until ( eof_large );
    set work.large end = eof_large;
    /* initialisation des variables avant qu'elles soient fusionnées à
partir de h_small */
    do i=lbound(smallvar) to hbound(smallvar);
        smallvar(i) = .;
    end;
    rc = h_small.find ();
    output;
end;
run;

```

Dans ce cas, aucun tri préalable n'est nécessaire sur les tables.

Résultats :

5 min 30 sec (26 sec en temps processeur)

Synthèse :

Fusion de table	Temps Réel	Temps UC
Méthode classique (1)	33 min 30s	2 min 59.73 sec
Hashing (2)	5 min 30s	26.18 sec
Rapport 1/2	6.09	6.87

Limitations :

Le code hash se base sur l'utilisation de la mémoire. Deux conséquences sont à noter :

1. Les objets hash existent et sont accessibles uniquement au sein de l'étape DATA dans laquelle ils ont été créés. Ceux-ci sont supprimés à la fin de l'exécution de cette étape DATA.
2. Les objets hash sont directement stockés dans la RAM. Leur taille est donc limitée par la quantité de mémoire disponible sur la machine. Il est préférable de faire une estimation de la taille qu'ils peuvent atteindre au préalable.

Estimation de la taille d'un objet hash :

Taille de l'objet hash = $\left(\sum_{i=1}^n \text{taille var}(i)\right) \times \text{nombre_d'enregistrements}$

Conclusion :

Tel que nous l'avons démontré dans cet article, le code hash est très puissant pour fusionner des tables, et plus généralement pour manipuler des données. Il ne s'agit pas là pour autant de dire que le code hash est la solution la plus adéquate à toutes les situations. L'utilisation des objets hash donne une alternative intéressante et significative à la programmation des étapes DATA, pour l'optimisation des performances sur de gros traitements.

Pour plus d'informations sur le code hash, vous pouvez consulter:

- les présentations du SAS Forum International :
 - <http://www2.sas.com/proceedings/sugi27/p012-27.pdf>
 - <http://www2.sas.com/proceedings/sugi31/244-31.pdf>
- L'aide en ligne :
 - <http://support.sas.com/onlinedoc/913/getDoc/fr/lrcon.hlp/a002586295.htm>

Pascal Lemetayer
Consultant Support Clients
SAS France

PRESENTATION DE SAS® APPDEV STUDIO 3.2

Découvrez en quelques points les nouveautés du produit.



Caractéristiques :

Catégories : Technologies web, SAS
AppDev Studio
OS : Windows
Version : SAS AppDev Studio 3.2
Vérifié le 01/01/0007

Nouvel environnement de développement

La nouvelle version de SAS AppDev Studio est disponible. Dans les versions précédentes, l'atelier de développement se nommait webAF. Avec cette nouvelle mouture, webAF a laissé place à un environnement de développement intégré à l'atelier OpenSource : Eclipse. SAS AppDev Studio est devenu SAS AppDev Studio Eclipse Plug-ins. Toutes les fonctionnalités intégrées à Eclipse sont disponibles, comme l'accès à l'aide en ligne, l'aide au développement, l'usage des 'snippets' (extrait de code), etc.

Nouveaux types de projet

Deux nouveaux types de projet apparaissent avec cette version :

- ✚ Projet Java SAS : environnement pour l'utilisation de SAS via une application, applet ou autres développements JAVA.
- ✚ Projet d'application Web SAS : environnement de construction d'application web JAVA (servlet, jsp) intégré à la plate-forme SAS.

Référentiel de classes SAS

Le référentiel est une collection de classes JAVA réutilisables et partageables entre les différents projets en développement. Ce référentiel permettra facilement de gérer des dépendances spécifiques souhaitées pour un projet en cours. Les classes sont gérées par numéro de version. Une gestion autonome des classes nécessaires à votre projet sera proposée par l'interface lors de la création de nouveau projet de développement.

Nouveaux modèles de développement

Le plugin amène de nouveaux modèles de développement afin de répondre à tous les développements pouvant être nécessaires dans le cadre de la solution SAS®9.

- Portlet pour l'environnement du SAS Information Delivery Portal,
- Plugin pour l'interface SAS® Management Console,
- Plugin pour l'interface SAS® Data Integration Studio,
- Création de rapports web (modèle SAS® Web Report Studio),
- Rapports OLAP.

Importation, exportation et migration

Une fonction d'importation des projets SAS AppDev Studio 3.0 permettra de migrer vos projets SAS AppDev Studio dans l'environnement Eclipse.

La fonction d'export est présente afin de générer une application WAR (Web ARchive), ou de créer un nouveau package de ressources JAVA (fichier JAR) à partir de votre projet.

SAS AppDev Studio Server-Side Catalogs est toujours fourni avec cette version pour les développements existants. Cette couche permet de continuer à faire fonctionner vos applications webAF/webEIS existantes avec SAS.

Un document spécifique sur la migration de vos développements existants et n'utilisant pas la nouvelle plate-forme SAS9 est disponible sur le web à cette adresse :

<http://support.sas.com/rnd/appdev/doc/ADS32Migration.pdf>

SAS® webEIS™

Le produit n'est plus distribué avec cette version. Il est remplacé par SAS Web Report Studio et SAS® Web OLAP Viewer for Java.

Documentation et pré-requis

Cette version du produit est disponible dans les environnements de développements suivants :

Systeme d'exploitation	Windows XP Professional (Service Pack 1 or later) Windows 2000 Professional (Service Pack 1 or later) Windows NT 4.0 Workstation (Service Pack 6a or later)
SAS	SAS® 9.1.3 Service Pack 4 *or* SAS Release 8.2
Machine	Processeurs de classe Intel ou Intel-compatible Pentium II (minimum recommandé) 512 MB RAM au minimum (plus de mémoire est fortement recommandé pour améliorer les performances) 700 MB d'espace disque

Vous retrouverez les éléments évoqués dans cet article ainsi que beaucoup d'autres informations sur le site web de SAS (lien : <http://support.sas.com/rnd/appdev/index32.html>)

Pascal Nicolas
Consultant Support Clients
SAS France

LES NOUVEAUTES SPDS 4.4

Découvrez en quelques points les nouveautés du produit SAS® Scalable Performance Data Server® 4.4.



Caractéristiques :

Catégories : SPDS, Système

OS : Unix, Windows

Version : SPDS 4.4

Vérifié le 01/03/2007

Changement pour les plate-formes supportées

SPDS 4.4 est officiellement supporté avec la version Unix Solaris x64 (il ne s'agit pas, bien entendu, de la seule plate-forme supportée mais de l'ajout lié à la version 4.4 de SPDS). En revanche, les plate-formes Linux ia64 et UNIX HP TRU64 ne sont plus supportées avec SPDS 4.4.

Avec SPDS 4.4, la documentation en ligne est maintenant disponible sur [SAS OnlineDoc 9.1.3](#)

Vues matérialisées (« materialized views »)

Une vue matérialisée permet de sauvegarder le résultat d'une vue SQL dans une table temporaire. Lorsqu'une requête est exécutée, plutôt que d'exécuter la vue afin d'obtenir le résultat, la table temporaire est alors utilisée. Si n'importe laquelle des tables utilisées en entrée est modifiée, la vue matérialisée mettra automatiquement à jour la table temporaire des résultats.

La vue matérialisée est uniquement supportée en conjonction avec l'interface « SPDS Server SQL Pass-Through ». L'utilisation de vues matérialisées permettra des gains de performance notables lors de l'utilisation de requêtes SQL faisant appel à des vues SQL.

Pour de plus amples détails concernant les vues matérialisées, on se reportera au chapitre « materialized views » du [guide d'utilisation de SPDS 4.4](#)

SPDS Server Profiling

Un « SPDS Server Profiling » et un « Logger Process » sont disponibles pour suivre et enregistrer l'activité des processus liés au serveur SPDS. Une fois enregistré, le journal peut être sauvegardé en tant que table SAS pour permettre une analyse ultérieure.

Le « SPDS Server Manager », qui fait partie de SAS® Management Console, peut se connecter au « SPDS Server Profiling » afin de fournir, en temps réel, des informations relatives aux processus SPDS (notamment l'activité CPU et la consommation mémoire).

Ces fonctionnalités ne sont disponibles que sur un serveur SPDS 4.4 installé sur une plate-forme UNIX.

Pour de plus amples détails concernant le « SPDS Server Profiling », on se reportera au chapitre « SMC SPD Server Manager » du [guide d'Administration du serveur SPDS 4.4](#).

Authentification des mots de passe avec LDAP (*)

L'authentification LDAP permettra d'authentifier un utilisateur avec un serveur LDAP au lieu de l'utilisation de la base PSMGR. Cette méthode d'authentification permettra à un utilisateur d'utiliser le même compte/mot de passe que ceux utilisés au niveau système sur le serveur UNIX (ou Windows). Ce dernier point est conditionné par une concordance des règles de définition des mots de passe (nombre minimal de caractères, nombre de chiffres présents dans le mot de passe, etc.) entre les différents systèmes.

L'administrateur peut choisir le mode d'authentification en positionnant les paramètres appropriés au niveau du serveur : soit en utilisant LDAP, soit en utilisant la base PSMGR. Une fois la méthode choisie, toutes les authentifications se feront avec cette méthode. Même si la méthode d'authentification choisie est LDAP, il faudra tout de même avoir créé un compte dans la base PSMGR afin de maintenir certaines informations nécessaires au fonctionnement du serveur SPDS (telle que l'appartenance à un groupe ou le niveau d'accès autorisé).

(*) Cette fonctionnalité n'est disponible qu'avec SPDS installé sur un serveur Windows ou un serveur Solaris.

Pour de plus amples détails concernant l'authentification LDAP avec le serveur SPDS, on consultera le chapitre « SPD Server Password Manager utility » du [guide d'Administration du serveur SPDS 4.4](#).

« Dynamic Locking »

« Dynamic Locking » vous permettra d'établir des règles de « locking » plus souples sur un domaine qui autorise différents clients à partager des accès en lecture et en écriture sur des tables du domaine, et ce sans avoir d'erreur de « locking ». Cette nouvelle méthode (i.e « dynamic locking ») diffère de la méthode déjà connue « SPDS record level locking » : les clients se connectent à un « SPD User Proxy » distinct pour chaque bibliothèque (correspondant à une instruction « libname ») du domaine.

Cette méthode peut permettre des améliorations de performance, notamment dans les cas où des accès concurrentiels (en lecture et en écriture) à une table sont nécessaires. Il faudra toutefois estimer les améliorations de performance au cas par cas.

Pour de plus amples détails concernant le « SPD Server Dynamic Locking », on consultera le chapitre « Dynamic Locking » du [guide d'Administration du serveur SPDS 4.4](#).

Définition d'une fenêtre de ports utilisables ("Surfacing ports through an internet firewall")

Si un site dispose d'un pare-feu, il devient nécessaire de contrôler les ports utilisés par le serveur SPDS et le client pour communiquer, afin que ces ports ne soient pas bloqués par le pare-feu. Certains ports utilisés par le serveur SPDS sont définis au démarrage du serveur, et peuvent donc aisément être identifiés, et donc autorisés. En revanche, des ports sont alloués dynamiquement afin de permettre chaque connexion au serveur SPDS, ainsi que les processus « User Proxy » qui en découlent. Ces ports sont généralement alloués en fonction des ports disponibles. Afin de contrôler les valeurs assignées pour ces ports, les paramètres serveur MINPORTNO et MAXPORTNO sont maintenant disponibles.

Pour de plus amples détails concernant le contrôle des ports utilisés, on consultera les chapitres « SPD Server Host Parameter Files » du [guide d'Administration du serveur SPDS 4.4](#) et « How do I know which ports must be surfaced through an internet firewall » du [guide d'utilisation de SPDS 4.4](#)

Les documentations complètes SPDS 4.4 (guide d'utilisation et guide d'administration) sont disponibles sur [le site du support dans la partie documentation](#).

Antoine Tardien
Consultant Support Clients
SAS France

Directeurs de la publication :

Philippe Hoffmann
Géraldine Deschamps

Comité de rédaction :

Antoine Tardien
Pascal Lemétayer
Pascal Nicolas

Comité de relecture

Fabienne Bernard
Mouloud Dey
Philippe Hoffmann



SAS FRANCE - DOMAINE DE GRÉGY - GRÉGY SUR YERRES - 77257 BRIE COMTE ROBERT - FRANCE

TÉL.: +33 (0) 1 60 62 11 11 - FAX: +33 (0) 1 60 62 11 99 WWW.SAS.COM/FRANCE

Copyright © 2007, SAS Institute Inc. Tous droits réservés.