Le Système SAS et les accès via OLE DB : une introduction

The Power to Know™

SOMMAIRE

Introduction.

OLE DB est une technologie Microsoft, en fait une API (application programming interface ou interface de programmation) permettant l'accès à des données sous des formes diverses et variées – relationnelles, mais aussi non-relationnelles et multi-dimensionnelles – par l'intermédiaire d'une seule interface de programmation COM.
OLE DB n'est pas un produit, au sens traditionnel du terme : ce qui entre en jeu, c'est comment les consommateurs OLE DB (consumers) communiquent avec les fournisseurs de données OLE DB (providers). Il se présente donc comme un modèle multi-couche, où les composants serveur contiennent les données, et les composants client établissent les connexions et accèdent aux données. Les fournisseurs et les consommateurs sont des objets COM, dialoguant en utilisant une série d'interfaces COM.

Nous allons voir dans ce document comment la technologie OLE DB peut être utilisée avec le Système SAS, d'abord du côté consommateur de données et ensuite du côté fournisseur de données (version du Système SAS utilisée : 8.2 (TS2M0)). Auparavant, nous ferons un rappel des autres modules existant dans l'offre SAS pour accéder à des bases externes, OLE DB étant le dernier module mis en avant dans ce domaine, avec la version 8 du Système SAS.

I- Les différents modes d'accès aux bases externes.

Dans notre offre, nous proposons plusieurs modules pour accéder à des données externes : Base SAS, SAS/ACCESS to ODBC, SAS/ACCESS to PC Files Format et la famille des SAS/ACCESS to "un SGBD" pour un accès natif aux données (Oracle, DB2, Sybase, etc).
Ce dernier ne sera pas abordé davantage dans le cadre de ce document, des spécificités existant dans le paramétrage, selon la base de données considérée.
Les trois premiers, de même que le module SAS/ACCESS to OLE DB, permettent par exemple de lire des fichiers Excel ou d'exporter des données SAS au format Excel.
Le code à écrire dans SAS étant différent selon le module utilisé, nous allons voir ci-dessous des exemples de syntaxe dans chacun des quatre cas : Base SAS et les SAS/Access to PC Files Formats, ODBC et OLEDB.

1- Avec Base SAS.

Le code se base sur les liens DDE, la plus ancienne méthode, historiquement, utilisée dans SAS. La première contrainte est que les deux applications entrant en communication doivent être en cours d'exécution. Le programme ci-dessous lit un fichier Excel - qui doit déjà être ouvert - contenant des données sur dix lignes et trois colonnes, dans la feuille nommée "Feuil1".

```
*-- Lecture d'un fichier Excel depuis SAS, via une étape DATA --;
filename xlsdde DDE "excel|feuil1!l1c1:l10c3";

data ma_table;
  infile xlsdde;
  input variable1$ variable2 variable3;
run;
```

Au niveau de l'écriture à proprement parler du code, plusieurs autres contraintes existent :

- il faut spécifier la plage de données sur laquelle la lecture sera faite (dans l'instruction FILENAME)
- et écrire l'étape data adéquate, où la totalité des noms des variables et leur type doivent être spécifiés explicitement, ce qui peut être fastidieux dans le cas de fichiers volumineux (en terme de nombre de variables).

Une Note Technique est consacrée aux liens DDE : http://ftp.sas.com/techsup/download/technote/ts325.pdf

2- Avec SAS/ACCESS to PC Files Format.

Ce module permet l'accès à une liste définie de sources de données. En version 8, nous proposons l'accès aux fichiers au format :

- Microsoft Excel (2000, 97, 7, 5, 4)
- Microsoft Access (2000, 97)
- DBase
- Lotus (1, 3, 4)
- Texte

Avant la version 8, les procédures à utiliser étaient ACCESS (pour importer ce type de données dans SAS) et DBLOAD (pour exporter des tables SAS sous ces formats). Avec la version 8, deux nouvelles procédures sont apparues : IMPORT et EXPORT. Leur syntaxe est plus simple, avec moins d'options à préciser par défaut.
Nous proposons également une interface à ces procédures (en passant par les menus File > Import Data, et Export Data), laquelle génère automatiquement le code adéquat.

```
 *-- Lecture d'un fichier Excel avec la procédure IMPORT en utilisant la syntaxe
minimale -- ;
 proc import datafile='c:\excel\pays.xls' out=matable;
 run;

 *-- Par défaut, c'est la première feuille qui est lue. L'instruction range permet
d'en choisir une autre -- ;
 proc import datafile='c:\excel\noms.xls' out=matable;
    range="Diplôme$";
 run;

 *-- La syntaxe est différente selon le type de fichier lu -- ;
 *-- Pour lire une table MS ACCESS, le nom de la base doit être indiqué dans
l'instruction database, et le nom de la table dans l'option table. -- ;
 *-- L'ajout de l'option dbms est nécessaire pour indiquer le format de la base de
données à lire -- ;
 proc import table='noms' out=work.noms dbms=access;
    database="C:\tests\access\mabase.mdb";
 run;
```

3- Avec SAS/ACCESS to ODBC.

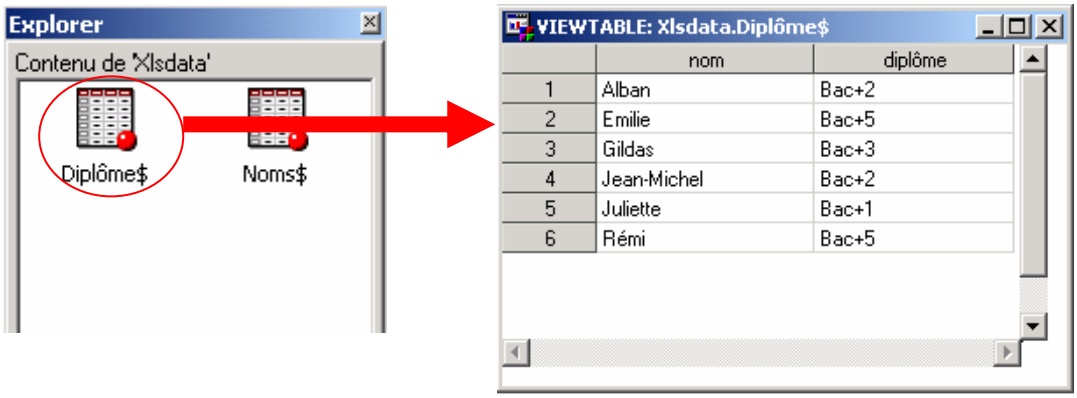
Le module permet d'accéder à une source de données, dès lors qu'elle dispose d'un pilote ODBC, sur l'environnement où SAS est installé.
L'étape préliminaire nécessaire est la configuration du pilote ODBC de la source de données. C'est à ce niveau que le classeur Excel, sur lequel sera fait la connexion, est spécifié.
Puis, dans SAS, cette connexion se fait soit via une procédure SQL, soit en utilisant une instruction libname (depuis la version 8) et le moteur ODBC.

```
*-- Information : le nom de la source de données définie dans l'administrateur de
sources de données ODBC est « Excel pour SAS » -- ;
*-- Cette procédure SQL permet par exemple de créer une table SAS temporaire
contenant les données de la feuille Nom$ -- ;
proc sql;
  connect to odbc (dsn="Excel pour SAS");
  create table test2 as
   select * from connection to odbc
    (select * from "Noms$");
  disconnect from odbc;
quit;
```

```
*-- Ou l'assignation d'un bibliothèque pointant sur le classeur Excel permet ensuite de visualiser ses feuilles dans l'explorateur SAS, les lire ou en écrire de nouvelles -- ;
libname xlsdata odbc dsn="Excel pour SAS";
```

Voyons maintenant les opérations possibles suite à la définition de la bibliothèque utilisant une source de données ODBC Excel.

- Visualisation des feuilles du classeur Excel via l'explorateur SAS.



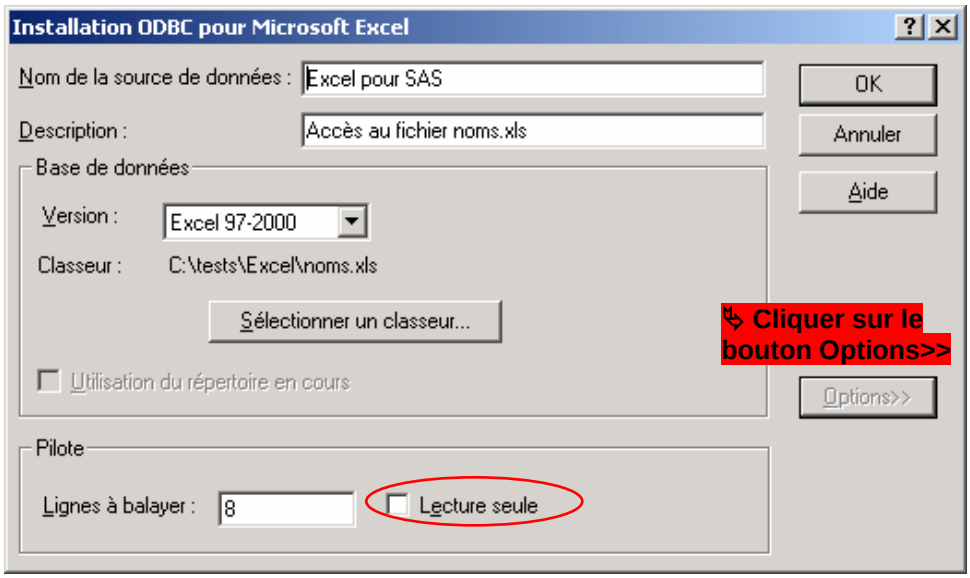
- Lecture des feuilles Excel dans SAS.

Une syntaxe particulière doit être utilisée, en raison du caractère $ apparaissant en suffixe des feuilles Excel :

```
data test1;
  set xlsdata.'Diplôme$'n;
run;
```

- Ecriture de nouvelles feuilles Excel depuis SAS.

L'insertion de nouvelles feuilles dans le classeur Excel se fait simplement avec une étape data.
Attention, il faut au préalable vérifier que l'option "lecture seule" est décochée, au niveau de la configuration de driver ODBC Excel ("Excel pour SAS").

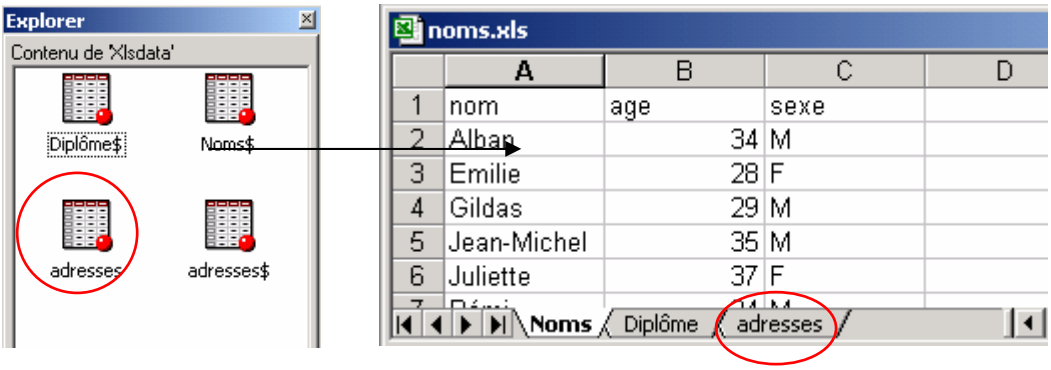


Ensuite, la syntaxe est “classique” :

```
data xlsdata.adresses;
  set matable;
run;
```

Et la vérification immédiate, soit par l’explorateur SAS, soit en ouvrant le classeur Excel (il faut au préalable désassigner la bibliothèque dans SAS - libname xlsdata clear;).

Vérification dans SAS *Vérification dans Excel*



L’opération inverse, à savoir la lecture d’une table SAS depuis une application externe, se fait en utilisant le pilote ODBC de SAS (fourni sur nos CDs d’installation). Dans cette configuration, le module SAS/Access to ODBC n’est pas nécessaire (Base SAS suffit).

Un document complet, en Français, sur les accès ODBC avec SAS est disponible sur notre site Internet : *http://www.sas.com/offices/europe/france/services/techsup/notes/document_tech7.pdf*

4- Avec SAS/ACCESS to OLE DB.

De la même manière que pour l’accès via ODBC, SAS/ACCESS to OLE DB permet d’accéder à toute source de données mettant à disposition un fournisseur OLE DB. OLE DB étant une technologie Microsoft, le module n’est disponible que sur plate-formes Windows.

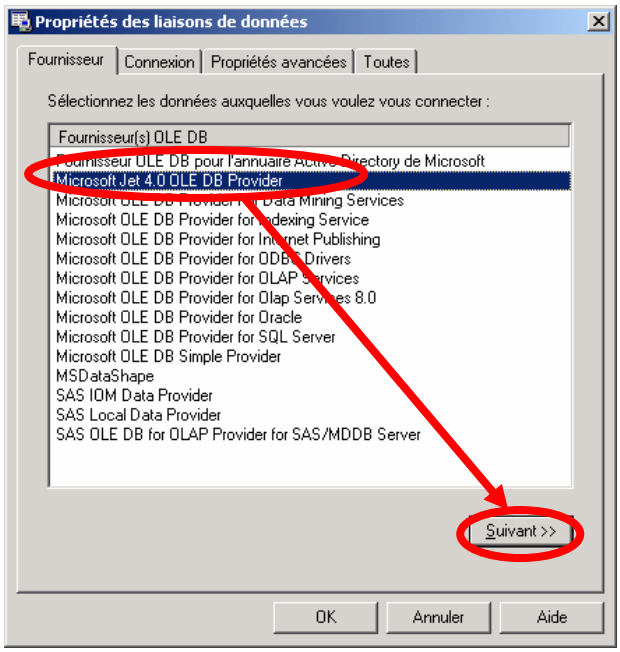
Dans SAS, il existe deux méthodes pour se connecter à une base via OLE DB, en utilisant soit les services OLE DB (la méthode utilisée par défaut dans l’interface SAS/ACCESS), soit le fournisseur OLE DB. Elles se différencient sur quelques points. Au niveau de la syntaxe, la première présente l’avantage d’être plus simple et éventuellement assistée. Et il est également préférable de l’utiliser pour bénéficier d’une optimisation des performances.

a) Connexion via les services OLE DB.

Très souvent, c'est la méthode la plus rapide et facile, dans la mesure où si le nom du fournisseur nous est inconnu, une interface est disponible pour trouver ce nom ainsi que les propriétés adéquates.
Pratiquement, il suffit de soumettre une instruction LIBNAME minimale (uniquement avec un libref et le moteur OLE DB spécifiés) et d'entrer les paramètres nécessaires à la connexion dans l'interface. Voyons comment cela se passe sur un exemple.

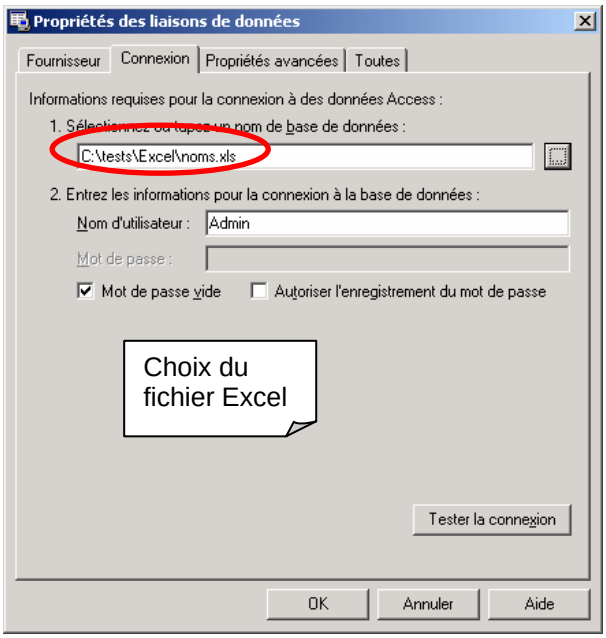
```
libname oledbxls oledb;
```

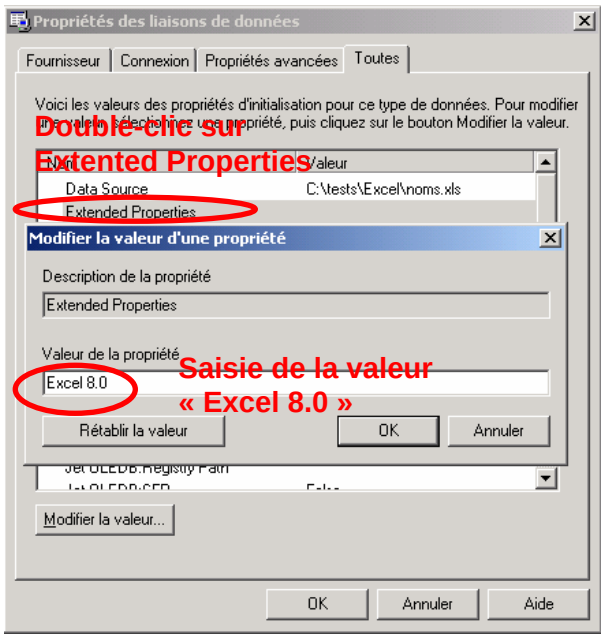
Cette instruction présume que l'on souhaite être prompté (l'option prompt=yes est ajoutée implicitement) et ouvre la fenêtre ci-dessous.



Choix du fournisseur, « Microsoft Jet 4.0 OLE DB Provider », pour lire un classeur Excel.

En fonction du fournisseur choisi, l'écran suivant (onglet Connexion), et les suivants, sont automatiquement adaptés en conséquence.





Un message dans la log SAS indique que la bibliothèque OLEDBXLS a été assignée. Son contenu est visualisable dans la fenêtre Explorer de SAS.

Il existe une astuce pour alors récupérer tous les paramètres passés lors de l'utilisation de l'interface, pour compléter l'instruction libname, et ainsi automatiser la déclaration de la bibliothèque dans son programme. C'est très utile dans le cadre de programmes qui seront soumis en batch par exemple.
Cette astuce consiste à utiliser le contenu de la macro-variable sysdbmsg. La log affiche alors ce type d'information :

```
84   %put &sysdbmsg;
OLEDB: Provider=Microsoft.Jet.OLEDB.4.0;Password="";Data
Source=c:\tests\excel\noms.xls;Extended Properties=Excel 8.0;Persist Security Info=True
```

laquelle est réutilisable directement dans une instruction libname, au niveau de l'option INIT_STRING= (valorisée à tout ce qui est affiché après OLEDB: dans la log).

```
libname new oledb
init_string = "Provider=Microsoft.Jet.OLEDB.4.0;Password="";Data Source=
c:\tests\excel\noms.xls;Extended Properties=Excel 8.0;Persist Security Info=True";
```

Toutes les options ne sont pas indispensables, l'instruction libname peut être simplifiée comme suit :

```
libname new oledb
init_string="Provider = Microsoft.Jet.OLEDB.4.0;
Data Source = c:\tests\excel\noms.xls;Extended Properties = Excel 8.0";
```

b) Connexion directe au fournisseur OLE DB.

Dans ce cas, des options supplémentaires doivent être spécifiées au niveau de l'instruction libname, à savoir :

- PROVIDER=, pour indiquer le nom du fournisseur OLE DB,
- OLEDB_SERVICES=NO (par défaut, elle est positionnée à Yes), il faut la positionner à No pour une connexion directe au fournisseur OLE DB,
- PROPERTIES=, pour spécifier la source de données.

Il faut aussi savoir que l'option INIT_STRING n'est pas autorisée.

```
*-- Définition d'une bibliothèque pointant sur une base Oracle -- ;
libname oledbora oledb provider=MSDAORA
                        properties=("user id"=scott "password"=tiger
                                    "data source"=oracle_path)
                        oledb_services=no;

*-- Définition d'une bibliothèque pointant sur un fichier Excel -- ;
libname oledbexc oledb provider="Microsoft.Jet.OLEDB.4.0"
                        properties=("data source" = 'C:\tests\excel\nom.xls')
                        provider_string="Excel 8.0"
                        oledb_services=no;
```

Les deux exemples ci-dessus montrent que la syntaxe varie sensiblement selon le fournisseur utilisé. Ainsi, lors d'une connexion à Oracle, une authentification est nécessaire (user = et password =) et, au niveau du paramètre « data source », le nom de l'instance Oracle doit être renseigné. L'option schema est également souvent utile, quand l'utilisateur se connectant n'est pas propriétaire des tables qu'il souhaite visualiser (par défaut, seules sont visibles dans la bibliothèque les tables créées en utilisant son propre user Oracle). Cette option s'ajoute au même niveau que provider = ou properties =.
Ou, lors de l'utilisation du fournisseur Microsoft Jet 4.0, il est nécessaire d'indiquer la chaîne « Excel 8.0 » pour que le fournisseur reconnaisse le type du fichier à traiter.

Le problème majeur de cette méthode est la nécessité de connaître à l'avance la valeur des paramètres comme le nom du fournisseur OLE DB. Or OLE DB ne dispose pas d'une interface du type de celle d'ODBC (avec son administrateur de source de données).
Il existe néanmoins une astuce pour lister, dans SAS, les fournisseurs disponibles sur sa machine, mais elle nécessite de connaître au moins l'un de ces fournisseurs (cf le code ci-dessous).

```
*-- Pour afficher (dans la fenêtre Output) la liste des fournisseurs OLE DB disponibles
sur la machine où la procédure SQL est exécutée -- ;
proc sql;
 connect to oledb
 (init_string="Provider=Microsoft.Jet.OLEDB.4.0;
  Data Source=c:\tests\excel\noms.xls;Extended Properties=Excel 8.0");
  select * from connection to oledb
  (OLEDB::PROVIDER_INFO());
run;
quit;
```

5- Tableau récapitulatif.

Le type de données à accéder, la nécessité de lire et/ou écrire dans tel ou tel type de format, la disponibilité d'un pilote ODBC ou OLE DB détermineront le (ou les) module(s) SAS utilisable(s).
Le tableau ci-dessous récapitule ces informations pour les quatre modules cités précédemment.

Module	Accès en lecture sur les données externes	Accès en écriture sur les données externes	Moteur V8 pour les instructions libname	Sources de données accessibles
Base (DDE)	Oui	Oui	Non	Applications Windows
SAS/Access to PC Files Format	Oui	Non	Non	Excel, Lotus, MS Access, Dbase
SAS/ACCESS to ODBC	Oui	Oui	Oui	Toute application disposant d'un pilote ODBC sur la machine
SAS/ACCESS to OLEDB	Oui	Oui	Oui	Toute application disposant d'un pilote OLE DB sur la machine
SAS/ACCESS to "un SGBD"	Oui	Oui	Oui	Oracle, DB2, Sybase, SQL Server, R/3, Informix, etc [1]

[1] *La liste complète est disponible sur notre site Internet : http://www.sas.com/products/access/index.html*

II- Exemples d'utilisation de SAS en tant que consommateur de données OLE DB.

Après avoir étudié les deux méthodes et la syntaxe pour accéder à des bases externes depuis SAS - ici consommateur de données - en utilisant le module SAS/Access to OLEDB, il est intéressant d'explorer les capacités de ce type d'accès. Ce sera l'objet de cette deuxième partie, par le biais de trois exemples d'utilisation.

1- Assignation de bibliothèques et opérations basiques sur des données externes.

Dans ce premier exemple, nous allons assigner deux bibliothèques SAS sur des bases Oracle et SQL Server. Elles sont ainsi accessibles via l'explorateur SAS mais aussi manipulables via des étapes data ou dans des procédures SAS, en utilisant la même syntaxe que pour des tables SAS, à savoir libref.table.

```
*-- Assignation de la bibliothèque OLEDBORA sur Oracle -- ;
libname oledbora oledb
        provider="MSDAORA"
        properties=("User ID"=scott "Data Source"=ServOra Password=tiger)
        oledb_services=yes;

*-- Assignation de la bibliothèque OLEDBSQL sur SQL Server -- ;
libname oledbsql oledb
        init_string="Provider=SQLOLEDB.1;Initial Catalog=Northwind;
                     Data Source=ServSql;Integrated Security=SSPI;
                     Persist Security Info=True";

*-- Description des deux bases via des procédures Contents -- ;
proc contents data=oleora.salaires;
run;

proc contents data=oledbsql.employes;
run;
```

Résultat des procédures CONTENTS sur les bases Oracle et SQL Server.

```
                    The CONTENTS Procedure

Data Set Name: OLEORA.SALAIRES     Observations:          .
Member Type:   DATA                Variables:             4
Engine:        OLEDB               Indexes:               0
Created:       .                   Observation Length:    0
Last Modified: .                   Deleted Observations:  0
Protection:                        Compressed:            NO
Data Set Type:                     Sorted:                NO
Label:
```

Note : Un certain nombre d'informations ne sont pas disponibles, en raison du moteur OLEDB utilisé, qui n'est pas en mesure de les récupérer dans le SGBD.

```
              -----Alphabetic List of Variables and Attributes-----

#    Variable        Type    Len    Pos    Format         Informat       Label
---------------------------------------------------------------------------------
3    CODEEMPLOI      Char      6     16    $6.            $6.            CODEEMPLOI
1    DATEEMBAUCHE    Num       8      0    DATETIME20.    DATETIME20.    DATEEMBAUCHE
2    ID              Char      6      8    $6.            $6.            ID
4    SALAIRE         Num       8     24                                  SALAIRE
```

```
                    The CONTENTS Procedure

Data Set Name: OLEDBSQL.employes        Observations:          .
Member Type:   DATA                     Variables:             6
Engine:        OLEDB                    Indexes:               0
Created:       .                        Observation Length:    0
Last Modified: .                        Deleted Observations:  0
Protection:                             Compressed:            NO
Data Set Type:                          Sorted:                NO
Label:

       -----Alphabetic List of Variables and Attributes-----

#    Variable    Type    Len    Pos    Format    Informat    Label
_____________________________________________________________________
1    Division    Char     30      0    $30.      $30.        Division
6    Id          Char      6    168    $6.       $6.         Id
2    Nom         Char     32     32    $32.      $32.        Nom
4    Pays        Char     25    112    $25.      $25.        Pays
3    Prenom      Char     32     72    $32.      $32.        Prenom
5    Ville       Char     16    144    $16.      $16.        Ville
```

2- Création de vue et utilisation de SQL Pass-through.

Plutôt que d'utiliser des bibliothèques, il est possible d'envoyer des requêtes directement à un SGBD, en utilisant la procédure SQL et le SQL Pass-through. Deux sections peuvent être distinguées dans le code de la procédure SQL ci-dessous : la connexion au SGBD, avec l'instruction CONNECT TO OLEDB, et ensuite le traitement demandé. Dans notre exemple, ce sera la création d'une vue avec :

- agrégation sur une variable (group by sur code emploi),
- sélection de certaines modalités de cette variable (clause where sur code emploi),
- calcul de la moyenne sur une autre variable (fonction avg sur salaire),
- et comptage du nombre d'individus (count(*)).

Ces opérations sont réalisées par le SGBD sur lequel la connexion a été faite, Oracle ici. Les avantages sont de réduire le trafic réseau et le temps de traitement dans SAS.

```
proc sql;
  connect to OLEDB (provider="MSDAORA"
                    properties=("User ID"=scott "Data Source"=ServOra
                                Password=tiger)
                    oledb_services=yes);
  create view consolidation as
    select * from connection to OLEDB
                (select codeemploi as "Code",
                        avg(salaire) as "Salaire_Moyen",
                        count(*) as "Nb_Employes"
                 from salaires
                 where codeemploi in ('PILOT1','FLTAT2','FINCLK')
                 group by CodeEmploi);
quit;

*-- Affichage de la vue sous format RTF -- ;
ods rtf file='c:\temp\euro.rtf';
proc print;
 format salaire_moyen eurox8.;
run;
ods rtf close;
```

Affichage du contenu de la vue.

Obs	Code	Salaire_Moyen	Nb_Employes
1	FINCLK	€31.811	53
2	FLTAT2	€29.079	63
3	PILOT1	€63.148	27

3- Jointure entre bases de formats différents et stockage externe pilotés depuis SAS.

Pour aller encore plus loin, le Système SAS peut être utilisé pour opérer une jointure entre une base Oracle et une base SQL Server, dont le résultat sera stocké sous un autre format, Excel par exemple. Aucune table ou résultat intermédiaire n'est stocké dans SAS.
Voyons comment cela se passe sur un exemple, reprenant les tables décrites dans le point 1, à savoir *Salaires* stockée dans Oracle et *Employes* dans SQL Server.

```
*-- Définition d'une bibliothèque, sur un nouveau fichier Excel -- ;
libname oleexcel oledb
init_string="Provider = Microsoft.Jet.OLEDB.4.0;
Data Source = c:\tests\excel\fusion.xls;Extended Properties = Excel 8.0";

*-- Jointure entre les deux bases externes (SQL Server et Oracle) dans un
fichier Excel, via une étape data : création de la feuille fusion -- ;
*-- Les bibliothèques ont été assignées dans le point 1 -- ;
data oleexcel.fusion(drop=pays ville);
 merge oledbsql.employes oleora.salaires ;
 by id;
 if pays='FRANCE';
run;

*-- Même opération, avec une procédure SQL : feuille fusion 2 créée -- ;
proc sql;
  create table oleexcel.fusion2 as
    select dateembauche, ora.id, codeemploi, salaire, division, nom, prenom
    from oleora.salaires as ora, oledbsql.employes as sqlsv
    where ora.id = sqlsv.id and pays='FRANCE';
quit;
```

Le fichier Fusion.xls est créé. Les feuilles Fusion et Fusion 2 contiennent les même informations, à savoir le résultat de la jointure entre les tables Employes et Salaires, sur la variable de clé ID, pour les employés Français.

fusion.xls

	A	B	C	D	E	F	G	H
1	Division	Nom	Prenom	Id	DATEEMBAUCHE	CODEEMPLOI	SALAIRE	
2	AIRPORT OPERATIONS	LE ROY	PATRICIA	E00072	04/03/1984	BAGCLK	24000	
3	AIRPORT OPERATIONS	LACOSTE	CLAIRE	E00083	18/12/1991	CHKCLK	20000	
4	HUMAN RESOURCES & FACILITIES	FAURE	JEROME	E00182	22/05/1988	RESCLK	27000	
5	HUMAN RESOURCES & FACILITIES	MEUNIER	CHRISTOPHE	E00187	23/10/1986	FACCLK	23000	
6	AIRPORT OPERATIONS	GRISET	ELIANE	E00220	12/06/1990	CHKCLK	41000	
7	SALES & MARKETING	BENTZ	MARIE	E00229	15/03/1994	SALCLK	44000	
8	AIRPORT OPERATIONS	LIGONNIERE	DAVE	E00245	03/10/1986	BAGCLK	28000	
9	HUMAN RESOURCES & FACILITIES	MIDONET-FAUVEL	FRANCIS	E00264	02/04/1987	FACCLK	32000	
10	AIRPORT OPERATIONS	HARTMANN	NICOLAS	E00477	01/11/1987	CHKCLK	34000	
11	FLIGHT OPERATIONS	RIGHENZI	NICOLAS	E00551	20/01/1989	MECH01	35000	
12	HUMAN RESOURCES & FACILITIES	BECKER	RAPHAELLE	E00592	13/03/1989	RECEPT	20000	
13	CORPORATE OPERATIONS	AUGUY	SERGE	E00593	21/01/1988	OFFMGR	105000	
14	HUMAN RESOURCES & FACILITIES	NIOT	JEAN FRANCOIS	E00636	24/03/1982	FACMNT	21000	
15	AIRPORT OPERATIONS	PICHOT	OLIVIER	E00659	16/09/1981	GRCREW	34000	
16	FLIGHT OPERATIONS	RADULOVITCH	PASCALE	E00704	06/06/1993	MECH01	32000	
17	SALES & MARKETING	BELDA	SALIMA	E00736	16/03/1983	SALCLK	23000	
18	AIRPORT OPERATIONS	WISER	VERONIQUE	E00843	01/01/1992	GRCREW	20000	
19	AIRPORT OPERATIONS	PICARD	JEAN-LUC	E01036	26/03/1994	GRCREW	35000	
20	HUMAN RESOURCES & FACILITIES	MOREAU	GLORIA	E01078	02/08/1986	FACMNT	29000	
21	FLIGHT OPERATIONS	VAUSSY	CLAIRE	E01094	26/09/1984	MECH03	24000	
22	AIRPORT OPERATIONS	POSTIC	BLANDINE	E01328	01/11/1990	GRCREW	34000	
23	HUMAN RESOURCES & FACILITIES	DELORGE	LAURENT	E01353	09/07/1980	HRCLK	32000	
24	AIRPORT OPERATIONS	MAINWARING	CHRISTI	E01365	23/09/1983	BAGCLK	28000	
25	HUMAN RESOURCES & FACILITIES	MAURICE	HELENE	E01395	09/10/1984	FACCLK	40000	
26	AIRPORT OPERATIONS	PRINSLOO	PASCAL	E01405	11/12/1991	GRCREW	30000	
27	HUMAN RESOURCES & FACILITIES	DE CORLIEU	HUGUES	E01480	03/06/1992	HRCLK	26000	
28	HUMAN RESOURCES & FACILITIES	MBIRKO	SERGE	E01549	20/02/1992	FACCLK	41000	

fusion / fusion2

III- Présentation des fournisseurs OLE DB SAS.

Les deux premières parties de ce document exposaient comment accéder à des bases externes à SAS, depuis une session SAS.

La seconde va être consacrée à l'opération inverse : accéder à des données SAS, depuis une application externe. SAS sera donc ici fournisseur de données. Cette opération était, avant l'introduction des technologies OLE DB dans SAS, possible uniquement en utilisant les pilotes ODBC SAS ou ODBC Universel, à la condition que l'application cliente adhérait au standard ODBC.
Le point III du document http://www.sas.com/offices/europe/france/services/techsup/notes/document_tech7.pdf détaille la configuration à mettre en place.
Avec la version 8, trois fournisseurs OLE DB SAS sont disponibles, lesquels permettent d'accéder aux tables et vues SAS stockées sur toute plate-forme où le Système SAS est supporté (et donc peut être installé), à partir d'une application compatible OLE DB - ADO.Il s'agit du :

- *SAS Local Data Provider*, pour accéder à des données SAS en local, sur son PC
- *SAS/SHARE Data Provider*, pour accéder à des données SAS sous le contrôle d'un serveur SAS/SHARE.
- *SAS IOM Data Provider*, pour accéder à des données SAS localisées sur un serveur distant, où le module Integration Technologies est installé. La connexion se fait alors via le protocole Bridge et un object spawner.
- *SAS OLE DB for OLAP Provider for SAS/MDDB Server*, se distingue des trois précédents dans la mesure où il permet l'accès aux bases multi-dimensionnelles SAS. La connexion se fait via un serveur Open Olap, démarré sur le serveur SAS.

Une application développée en Visual Basic est disponible sur notre site Internet pour tester l'accès aux données SAS via les trois premiers fournisseurs listés ci-dessus. Vous la trouverez à cette adresse : *http://support.sas.com/rnd/eai/samples/hello/index.html*
L'interface est appelée via l'exécutable HelloProviders.exe (installé en décompressant l'archive zip : *http:/support.sas.com/rnd/eai/samples/hello/hellosample.zip*).
Veillez à bien valider que les composants nécessaires à son bon fonctionnement soient installés :

- La version 8.2 du Système SAS, avec un ou plusieurs des fournisseurs OLE DB SAS.
- "Microsoft Visual Basic 6.0".
- "Microsoft ActiveX Data Objects 2.5 Library" (incluse avec le système d'exploitation Windows).

Ensuite, nous verrons comment l'accès se configure depuis Excel.

1- SAS Local Data Provider.

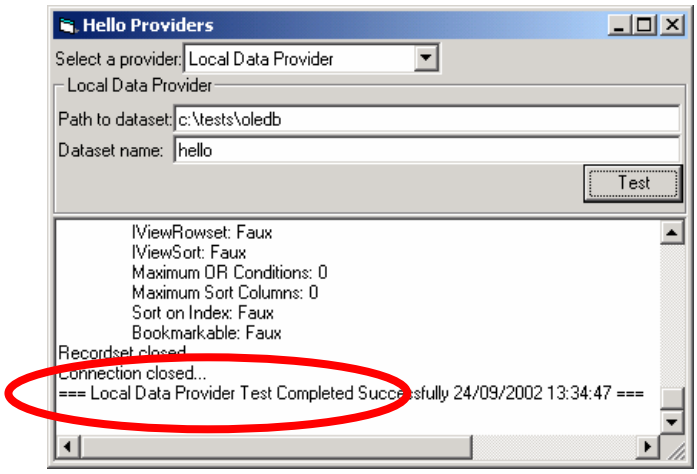
Ce fournisseur OLE DB est disponible avec le module Base SAS et fournit un accès sur des tables SAS.
Il est installé dans le répertoire "C:\Program Files\SAS Institute\Shared Files\SAS OLE DB Data Providers" (chemin proposé par défaut), comme le seront d'ailleurs les autres fournisseurs OLE DB SAS. Il est possible de vérifier sa présence en s'assurant que le fichier **sasafbas.dll** existe dans ce répertoire.

Aucun paramétrage n'est nécessaire par défaut dans SAS. La connexion est faite automatiquement au serveur SAS local. Toutes les tables visibles dans l'explorateur Windows seront accessibles via ce type de connexion, qu'elles aient été créées sur :

- Windows, avec la version 6, 7 ou 8 du Système SAS,
- ou Unix, OS/2, Open VMS Alpha, avec la version 7 ou 8 du Système SAS (utilisation de la fonctionnalité CEDA - intégrée à Base SAS depuis la version 8.2 - permettant de lire des tables SAS provenant d'un environnement différent).

a) Test de connexion depuis l'application Visual Basic HelloProviders.

L'application s'exécute en double-cliquant sur HelloProviders.exe. Elle permet de tester la connexion à une table SAS stockée sur la même machine.

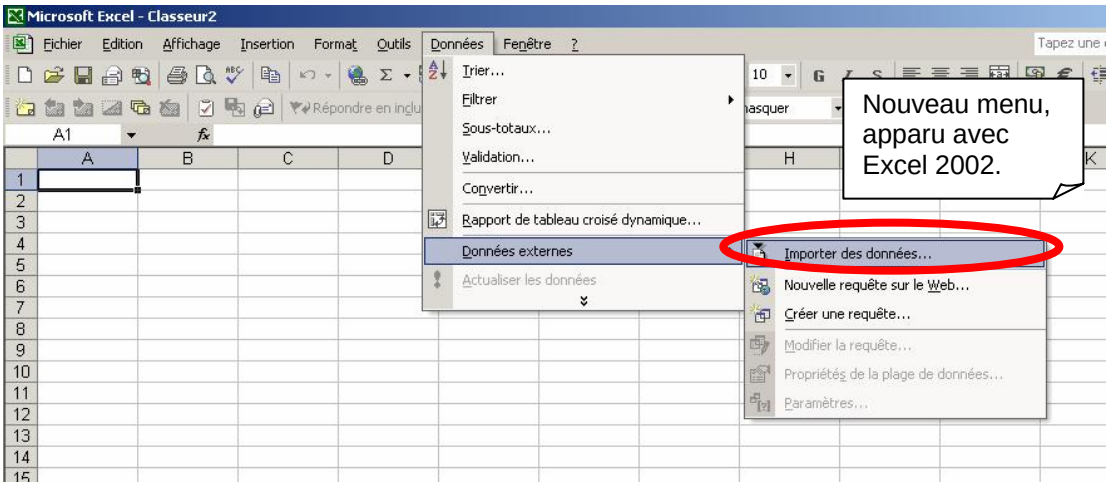


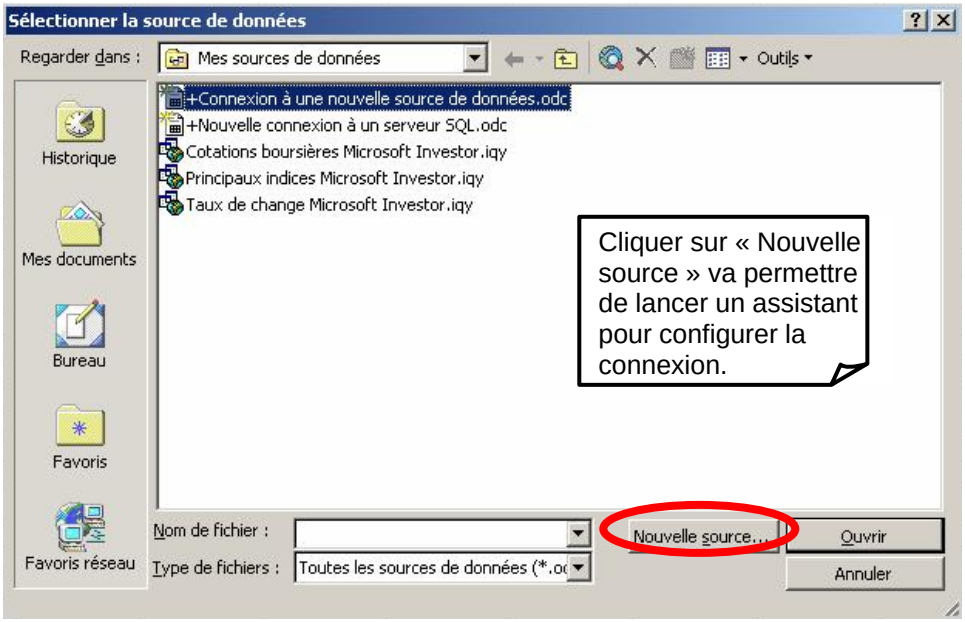
Les paramètres entrés :

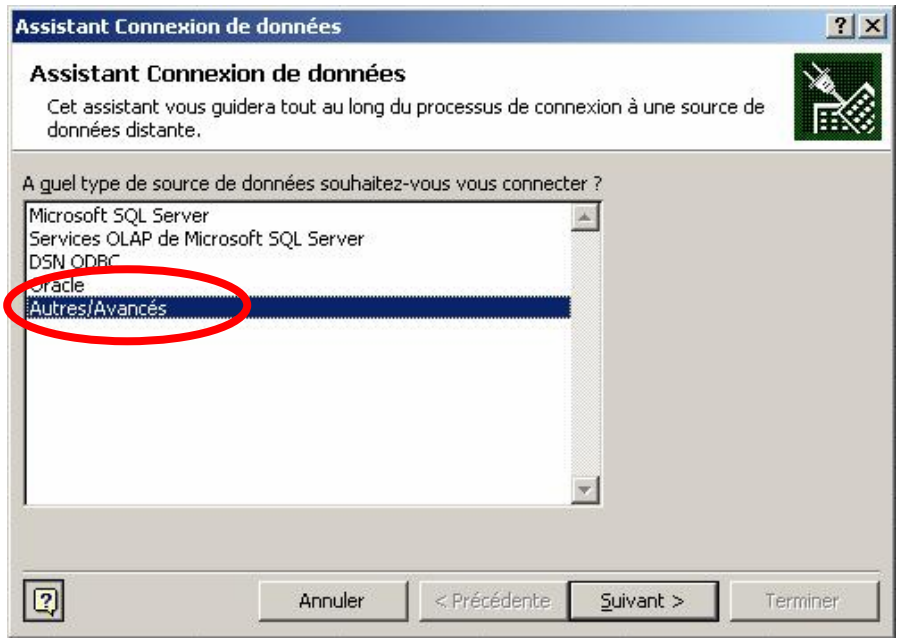
c:\tests\oledb : répertoire dans lequel la table SAS va être recherchée.
hello : nom de la table SAS.

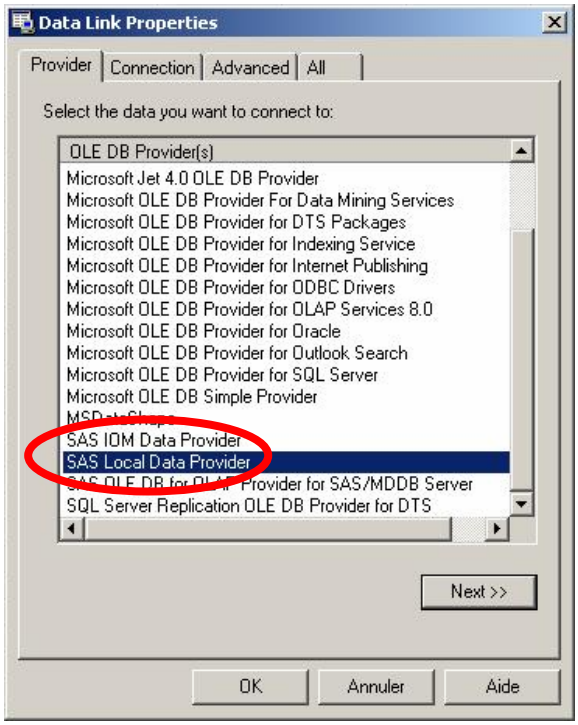
b) Test de connexion depuis Excel.

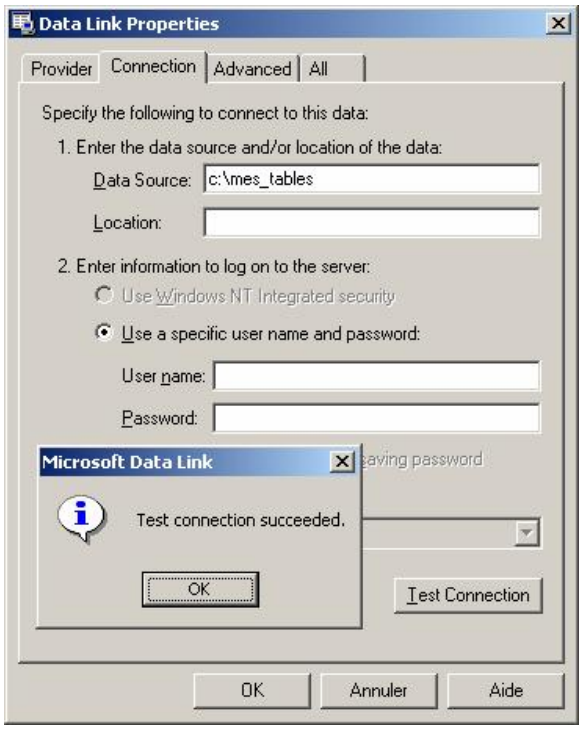
A partir de la version Excel 2002 (du pack Microsoft Office XP), il est possible d'accéder à des données externes via OLE DB. Voyons sur un exemple les étapes à suivre.





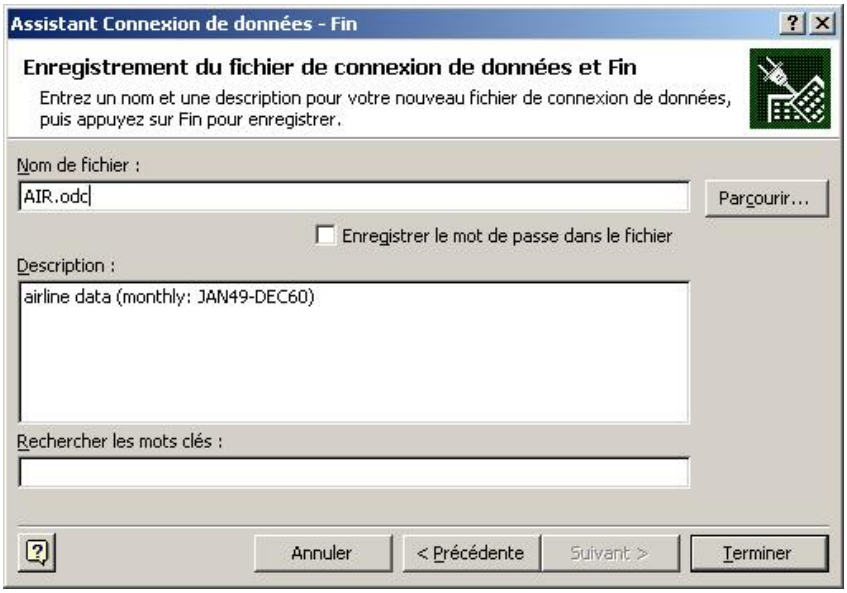




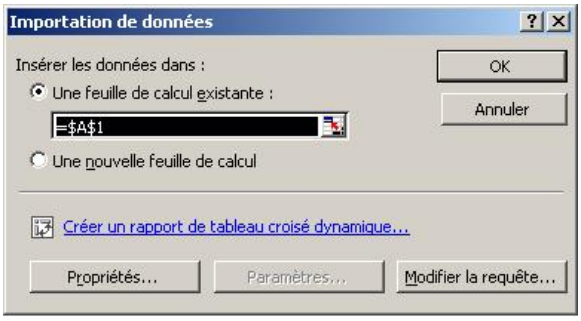


Après avoir spécifié le répertoire où se trouvent les tables SAS, cliquer sur le bouton « Test Connection » permet de s'assurer que la connexion est correctement définie.

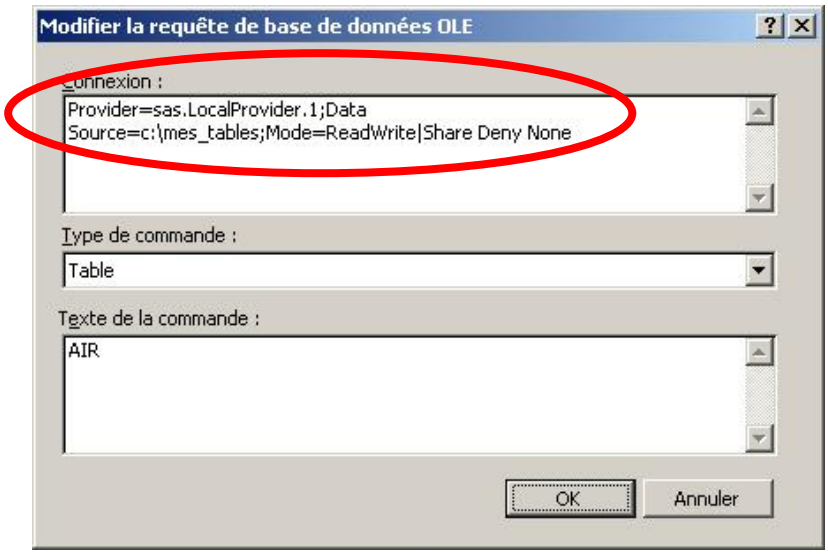
Cet écran permet de choisir la table à afficher dans Excel



Enregistrer dans un fichier (ici AIR.odc) tous les paramètres sélectionnés dans les écrans précédents permettra d'utiliser ultérieurement cette connexion.



L'importation est prête. En cliquant sur OK, les données seront insérées dans la feuille Excel active.
Le bouton « Modifier la requête… » est intéressant dans la mesure où il permet de visualiser – et éventuellement modifier – les propriétés de la connexion.



2- SAS/SHARE Data Provider.

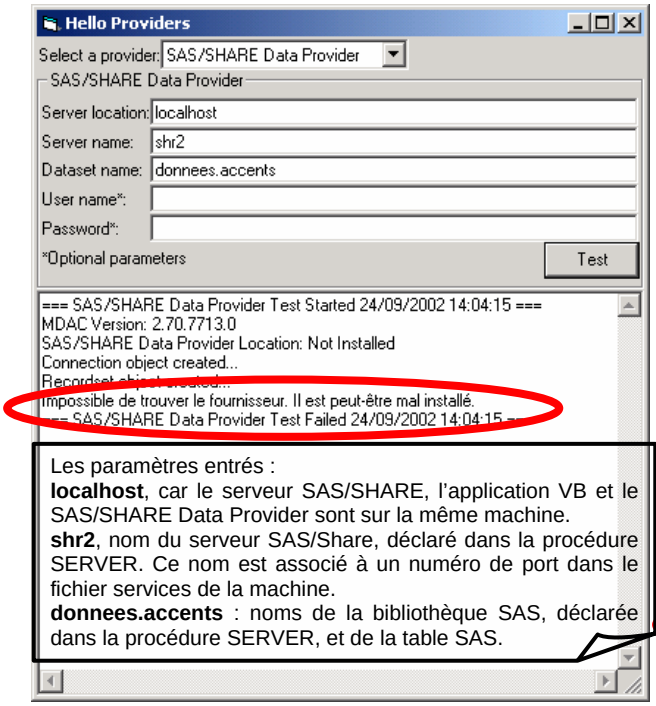
Ce fournisseur OLE DB est disponible avec le module SAS/SHARE. Il est possible de vérifier sa présence en s'assurant que le fichier **sasafshr.dll** existe dans le sous-répertoire "SAS OLE DB Data Providers". Deux fichiers d'aide sont également présents dans ce répertoire : sasafshr.chm et adocook.chm, pour vous aider dans l'utilisation de ce fournisseur.

Un serveur SAS/Share doit être démarré sur le serveur, avec une syntaxe telle que :

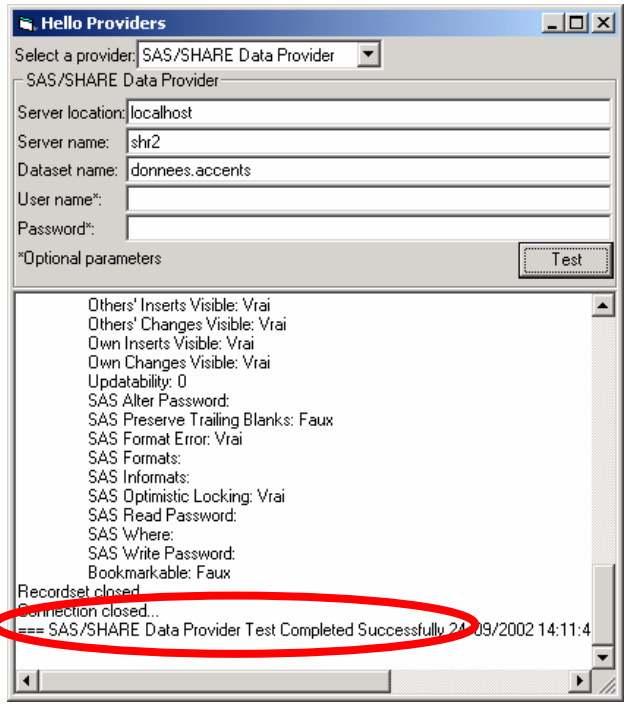
```
PROC SERVER ID=shr2 authenticate=optional;
 allocate library donnees "c:\tests\v8";
RUN;
```

Une bibliothèque appelée DONNEES est définie, laquelle sera ensuite utilisée dans les tests de connexion.

a) Test de connexion depuis l'application Visual Basic HelloProviders.



Les paramètres entrés :
localhost, car le serveur SAS/SHARE, l'application VB et le SAS/SHARE Data Provider sont sur la même machine.
shr2, nom du serveur SAS/Share, déclaré dans la procédure SERVER. Ce nom est associé à un numéro de port dans le fichier services de la machine.
donnees.accents : noms de la bibliothèque SAS, déclarée dans la procédure SERVER, et de la table SAS.

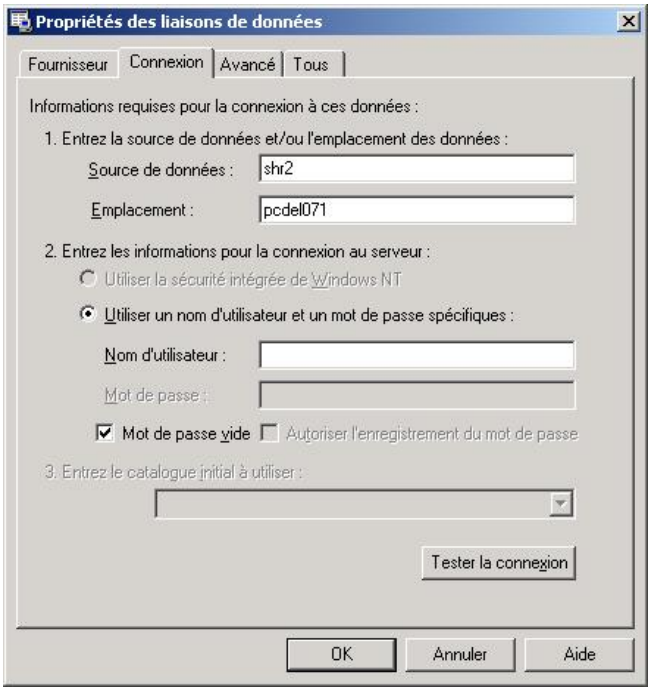


Le premier test de connexion a échoué. Le message laisse supposer un problème d'installation du fournisseur "*SAS/SHARE Data Provider".* L'exécutable permettant de l'installer est fourni sur le CDROM SAS Client-Side Components de la version 8.2 : D:\share\shroledb.exe (D étant ici le disque correspondant au lecteur de CDROM).
Après son installation, le deuxième test montre que la connexion fonctionne.

Quand le serveur SAS/SHARE et l'application devant accéder aux données ainsi mises à disposition sont démarrés sur des machines distinctes (ce qui sera le cas le plus souvent), il faut bien penser à déclarer le service dans le fichier services du client.
Exemple : `shr2  5010/tcp  # pour serveur SAS/share`
Et il faut donner, dans le champ "Server Location", le nom de la machine sur laquelle le serveur SAS/SHARE a été démarré, et non plus localhost.

b) Test de connexion depuis Excel.

Le principe reste le même que lors de l'utilisation du fournisseur SAS Local Data Provider dans la section précédente.
Les différences vont porter bien sûr sur les onglets de l'écran "Propriétés des liaisons de données " ou "Data link properties", dans l'onglet Fournisseur bien sûr et sur l'onglet "Connection". Dans "Source de données", il faut spécifier le nom du serveur SAS/SHARE (shr2 dans notre exemple) et dans Emplacement, le nom de la machine où le serveur SAS/SHARE a été démarré (localhost). La case Mot de passe vide doit être cochée.



Les propriétés de la connexion sont renseignées dans le fichier ODC comme suit :

Provider=sas.SHAREProvider.1;
Data Source=shr2;
Location=pcdel071;
Mode=ReadWrite|Share Deny None;
SAS Server Release=8

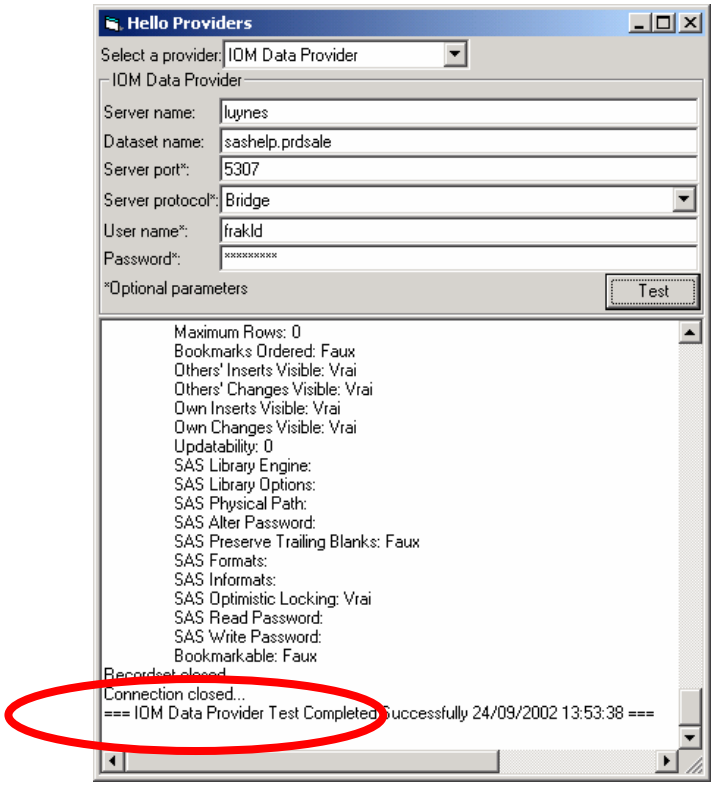
3- SAS IOM Data Provider.

Ce fournisseur est disponible avec le module Integration Technologies. Il est possible de vérifier sa présence en s'assurant que le fichier **sasaorio.dll** existe dans le sous-répertoire "SAS OLE DB Data Providers". Si ce n'est pas le cas, il peut être installé à partir de l'exécutable inttech.exe localisé sur le CDROM SAS Client-Side Components, dans le répertoire D:\inttech\install.

Côté configuration, un minimum est nécessaire sur le serveur. En effet, un processus "SAS Object Spawner" doit être démarré sur la machine où le module Integration Technologies est installé. Lors de demandes de connexion d'applications clientes, une session SAS est alors démarrée.

Des articles ont été consacrés à la mise en place des Object Spawners dans la lettre d'information du support clients français (Allo Support) sur les plate-formes Windows (n°4) OS/390 (n°5) et Unix (n°8). *Ils sont disponibles sur notre site Internet :*
http://www.sas.com/offices/europe/france/services/techsup/allosupport.html

a) Test de connexion depuis l'application Visual Basic HelloProviders.



Les paramètres entrés :

luynes : nom du serveur
sashelp.prdsale : nom de la bibliothèque et de la table à accéder
5307 : port sur lequel l'object spawner est déclaré
Bridge : protocole

b) Test de connexion depuis Excel.

Les informations nécessaires sont le numéro de port et le nom de la machine sur laquelle l'object spawner a été démarré, à indiquer respectivement dans les champs Source de données et Emplacement. Le nom de l'utilisateur et le mot de passe devront aussi être renseignées pour se connecter à la machine distante.

4- SAS OLE DB for OLAP Provider for SAS/MDDB Server.

Ce dernier fournisseur se distingue des autres dans la mesure où il permet l'accès aux structures multi-dimensionnelles SAS.

Les grandes étapes à suivre, pour préparer l'accès aux bases multi-dimensionnelles (MDDB) via ce fournisseur, sont les suivantes (opérations à réaliser sur le serveur) :

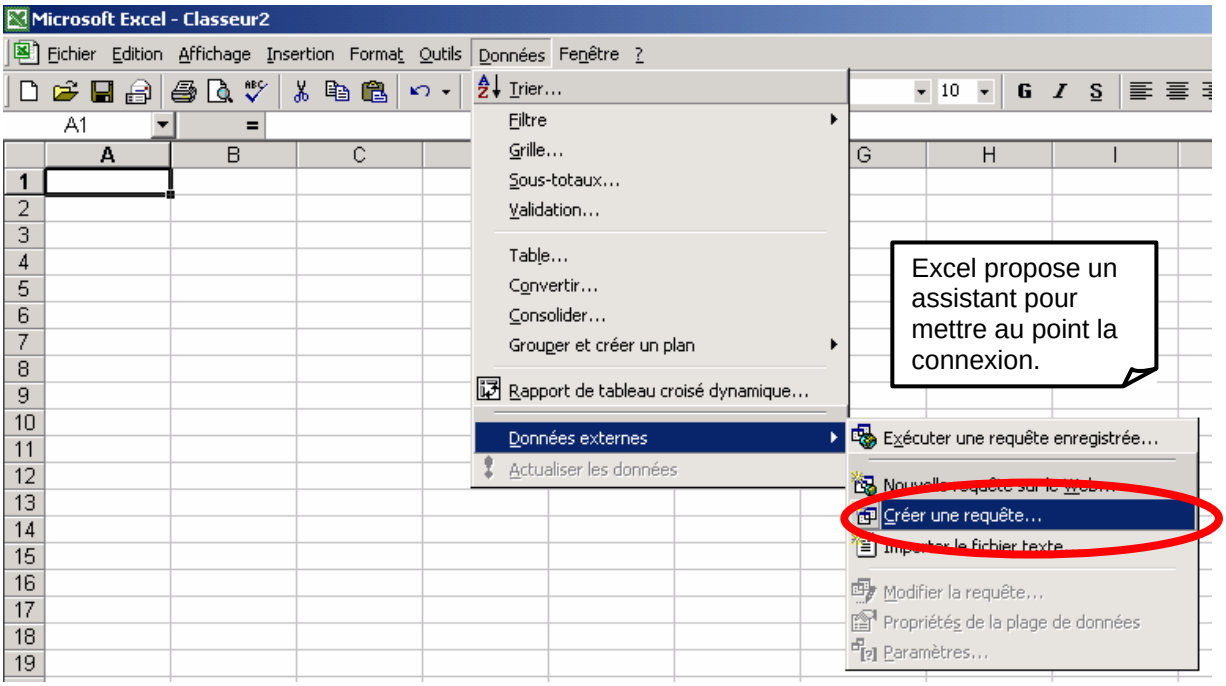
- enregistrer la MDDB dans un référentiel
- ajouter l'attribut OLAPMETA au(x) MDDB(s)
- démarrer un serveur Open Olap ; la commande est :
  AF C=SASHELP.OPNOLAP.LISTENER.SCL PORT=5154 (5154 est le port utilisé par défaut, quand l'option n'est pas ajoutée)

Des informations détaillées sont disponibles dans l'aide en ligne SAS : SAS System Help ➔ Help on SAS Software Products ➔ Open Olap Server.

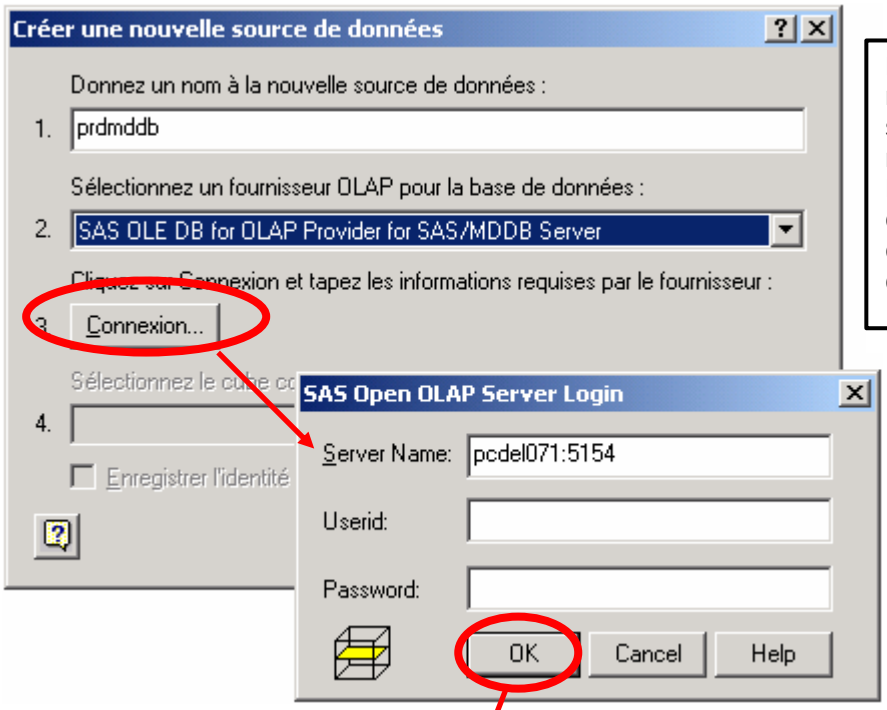
Du côté client, le fournisseur SAS OLE DB for OLAP Provider for SAS/MDDB Server doit être installé. Il est disponible sur le CDROM SAS Client-Side Components, dans la rubrique "SAS/MDDB Server Software". Le nom de l'exécutable est ooscl30.exe. En choisissant une installation personnalisée

(custom), il faut cocher Open OLAP Server (Client). Le répertoire d'installation proposé par défaut est C:\Program Files\SAS Institute\Shared Files\Open OLAP Solutions, la dll concernée porte le nom SASMprov.dll.
Un deuxième composant peut être installé. Il est intéressant de le mentionner car il permet d'administrer le serveur Open Olap. Ainsi la commande *ping* permet de vérifier que le serveur est bien démarré et qu'il répond, et *stop* de l'arrêter. Ce composant apparaît lors du processus d'installation sous la dénomination Open OLAP Server (Administrator) et est installé par défaut dans le répertoire C:\Program Files\SAS Institute\Open OLAP Server.

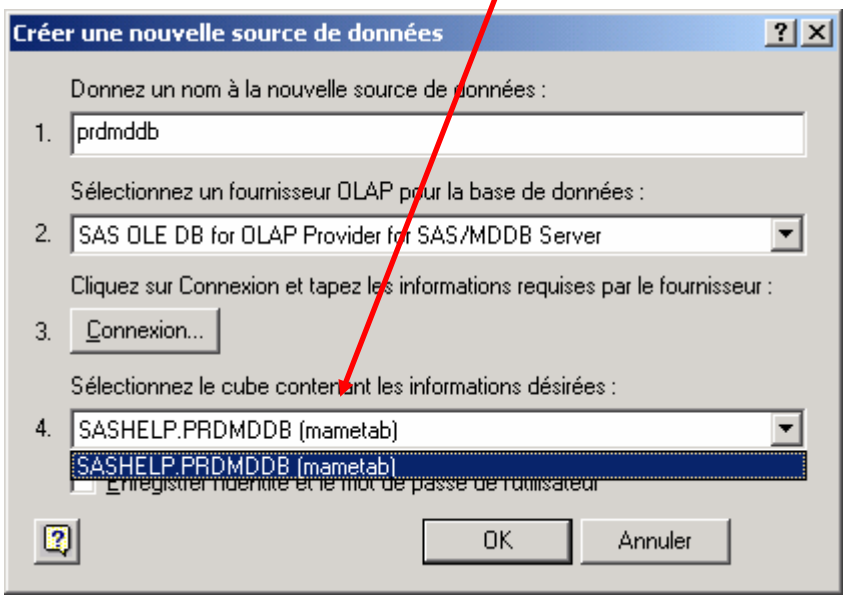
Voyons maintenant concrètement sur un exemple comment une MDDB SAS peut être accédée depuis Microsoft Excel.



Aller dans l'onglet Cubes OLAP pour créer une nouvelle source de données.



Plusieurs paramètres sont nécessaires pour définir la source de données : un nom, le fournisseur OLE DB, le nom de la machine où le serveur Open Olap a été démarré et enfin le cube.

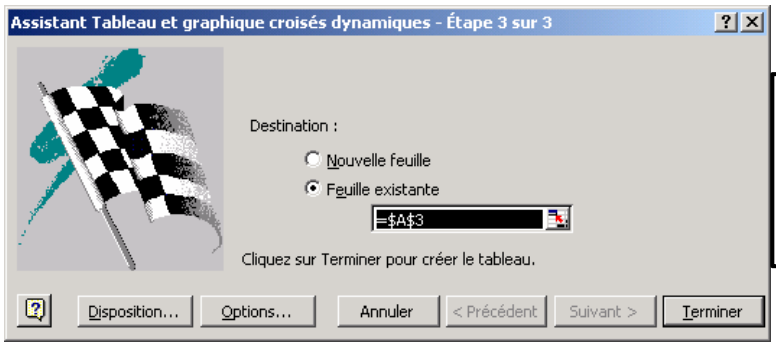


Le nom du cube est PRDMDDB, est stocké dans la bibliothèque SASHELP et est enregistré dans la métabase mametab.

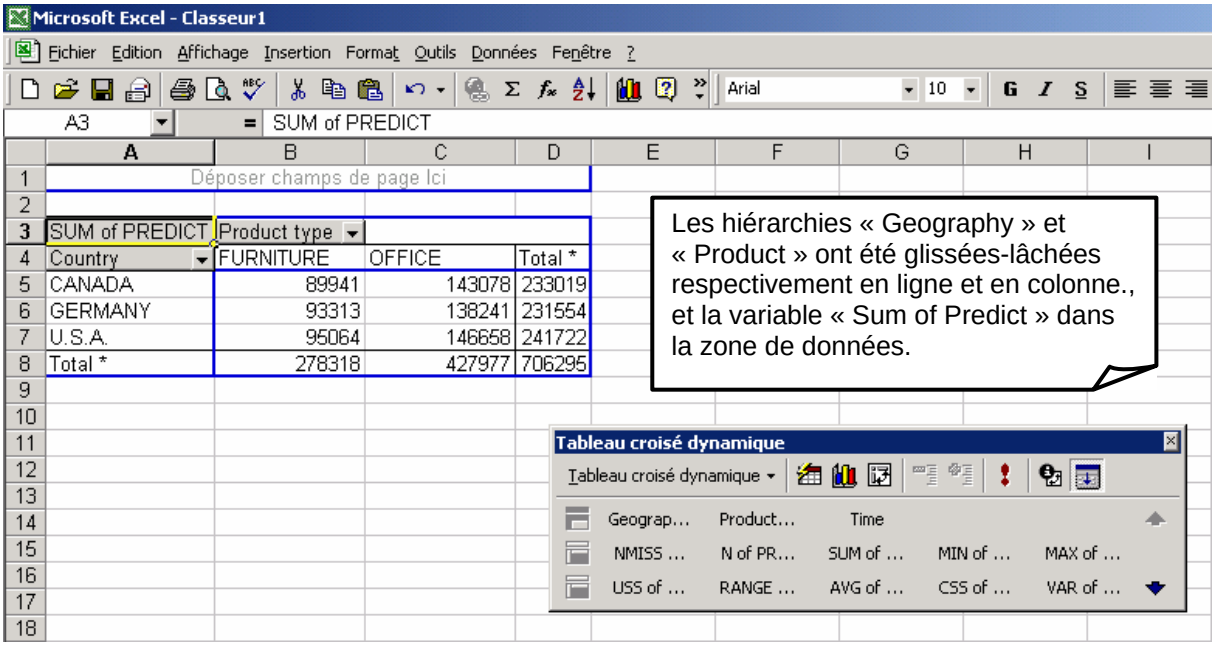
La source de données prdmddb est maintenant définie.

La définition de la source de données est visualisable dans le fichier prdmddb.oqy, stocké par défaut (sur Windows 2000) dans le répertoire :
C:\Documents and Settings\<userid>\Application Data\Microsoft\Requêtes

```
QueryType=OLEDB
Version=1
CommandType=Cube
Connection=Provider=SASMPROV.1;Persist Security Info=False;User ID="";Data
Source=pcdel071:5154;Location="";Mode=Read;Extended Properties=""
CommandText=SASHELP.PRDMDDB (mametab)
```



La dernière étape de l'assistant permet de choisir la feuille et la plage de données pour insérer les données.



Les hiérarchies « Geography » et « Product » ont été glissées-lâchées respectivement en ligne et en colonne., et la variable « Sum of Predict » dans la zone de données.

De très nombreux exemples et des "recettes" pour vos développements utilisant nos fournisseurs OLE DB sont disponibles sur notre site Internet, dans notre « ADO/OLE DB Cookbook » :
http://support.sas.com/rnd/eai/oledb/index.htm

SAS France
Domaine de Grégy - BP 5
77166 Grégy-sur-Yerres
Tél. : 01 60 62 11 11
Fax : 01 60 62 11 99

SAS Europe, Middle East & Africa
P.O. Box 10 53 40
Neuenheimer Landstr. 28-30
D-69043 Heidelberg, Germany
Tel: +49 6221 4160, Fax: +49 6221 474850

www.sas.com/france