TP – Informatique Décisionnelle

OLAP : Mondrian et Pentaho

PARTIE 1 : Installer et lancer Pentaho sur Mac

1. Télécharger la version libre de Pentaho sur :

   http://sourceforge.net/projects/pentaho/files/Business%20Intelligence%20Server/

2. Extraire le fichier *biserver-ce-5.0.1-stable.zip* téléchargé.
3. En utilisant la ligne de commandes, se situer sur le répertoire crée lors de l'extraction du fichier ZIP et lancer le script :
   ./start-pentaho.sh
4. Vérifier que le script est bien exécuté (sans erreurs de Java).
5. Ouvrir un navigateur internet et ouvrir l'URL suivante :
   http://localhost:8080/Pentaho
6. Concernant les paramètres de l'utilisateur, lancez la session de Pentaho en tant qu'Administrateur (User name : Admin ; Password : password).
7. Pour arrêter Pentaho, lancer la commande : *./stop-pentaho.sh*

Installer et lancer Pentaho sur Windows :

1. Télécharger la version libre de Pentaho sur :

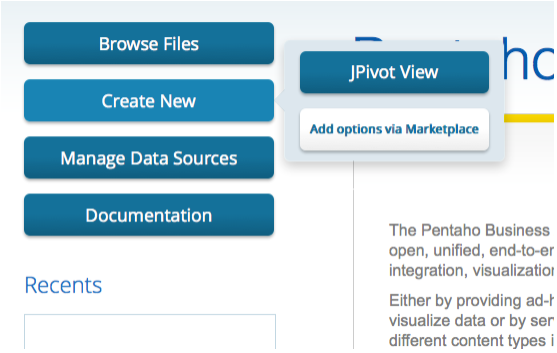
   http://sourceforge.net/projects/pentaho/files/Business%20Intelligence%20Server/

2. Extraire le fichier *biserver-ce-5.0.1-stable.zip* téléchargé.
3. En utilisant la ligne de commandes, se situer sur le répertoire crée lors de l'extraction du fichier ZIP et aller sur : *tomcat\bin* et lancer la commande *catalina.bat version.*
4. Si la commande donne comme résultat la version de Tomcat, alors tout est correct, sinon, vérifier dans les variables du système de Windows les valeurs suivantes (par exemple) :
   JAVA_HOME=C:\Program Files (x86)\Java\jdk1.6.0_22
   JRE_HOME=C:\Program Files (x86)\Java\jre1.6.0_22
5. Lancer une nouvelle ligne de commandes en tant que administrateur. Sur le même répertoire *bin* de tomcat lancer la commande :
   service.bat install
6. Pour lancer le serveur Tomcat, sur le même répertoire, lancer la commande *tomcat6w.exe*. Une fenêtre devrait apparaître. Aller sur la section Java et dans Java Options, ajouter les lignes suivantes :
   -Dsun.rmi.dgc.client.gcInterval=3600000
   -Dsun.rmi.dgc.server.gcInterval=3600000
   -XX:+UseConcMarkSweepGC
   -XX:+CMSPermGenSweepingEnabled
   -XX:+CMSClassUnloadingEnabled
   -XX:MaxPermSize=256m
   -Xss1M
   Ces lignes permettent, entre autres, gérer la mémoire utilisé pour la machine virtuel Java ainsi que de gérer les différentes versions du JRE.

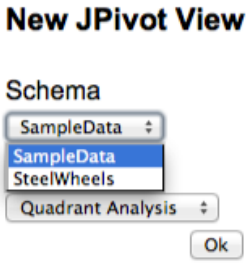
7. Sur la même fenêtre, aller sur la section General et appuyez sur le bouton START pour démarrer Tomcat.
8. Pour démarrer Pentaho, lancer le script :
   start-pentaho.bat
   Vérifier que le script est bien exécuté (sans erreurs de Java).
9. Ouvrir un navigateur internet et ouvrir l'URL suivante :
   http://localhost:8080/pentaho
10. Concernant les paramètres de l'utilisateur, lancez la session de Pentaho en tant qu'Administrateur (User name : Admin ; Password : password).
11. Pour arrêter Pentaho, lancer la commande : *stop-pentaho.sh*

PARTIE 2 : Interface de Pentaho

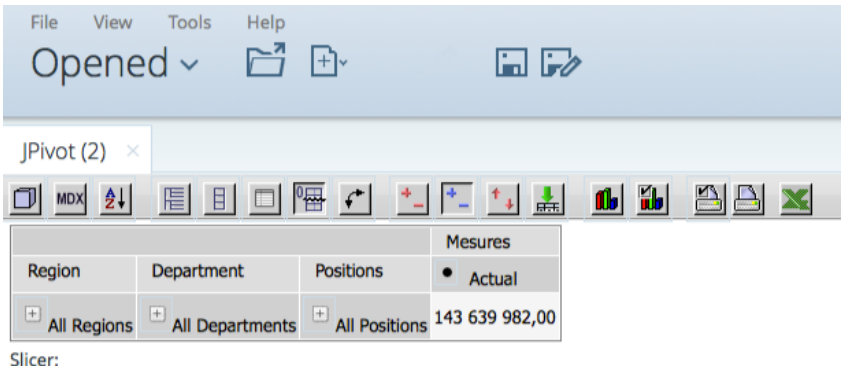
Après être authentifié, nous allons créer un JPivot (librairie conçu en Java pour la navigation et exploitation des informations via des opérations OLAP).



Deux jeux de données de essai sont proposés pour Pentaho. Dans ce TP, nous allons utiliser le premier.



Une fois sélectionnée le Schéma et le Cube, nous accédons à l'interface de manipulation de données et construction de requetés OLAP.



PARTIE 3 : Exemple fourni par Pentaho (Sample data)

Des données : la base de données d'exemple (représenté par un script SQL) se trouve dans le répertoire :

*biserver-ce**data**hsqldb*

Le fichier est *sampledata.script*

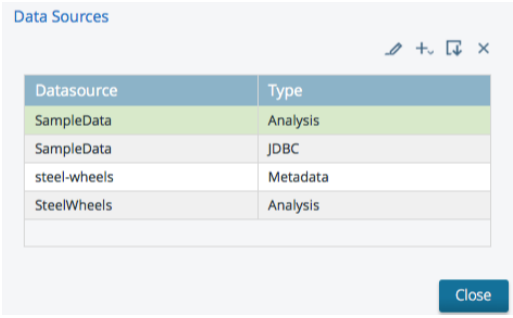
Le cube (fait) : le cube qui a été construit à partir de la base de données est représenté par une structure XML. Elle comporte les dimensions, hiérarchies ainsi que les indicateurs (« Measures »). Nous allons utiliser le cube Quadrant Analyse qui a la structure suivante :

```
<?xml version="1.0"?>
<Schema name="SampleData">
<!-- Shared dimensions -->
 <Dimension name="Region">
  <Hierarchy hasAll="true" allMemberName="All Regions">
   <Table name="QUADRANT_ACTUALS"/>
   <Level name="Region" column="REGION" uniqueMembers="true"/>
  </Hierarchy>
 </Dimension>
 <Dimension name="Department">
  <Hierarchy hasAll="true" allMemberName="All Departments">
   <Table name="QUADRANT_ACTUALS"/>
   <Level name="Department" column="DEPARTMENT" uniqueMembers="true"/>
  </Hierarchy>
 </Dimension>
 <Dimension name="Positions">
  <Hierarchy hasAll="true" allMemberName="All Positions">
   <Table name="QUADRANT_ACTUALS"/>
   <Level name="Positions" column="POSITIONTITLE" uniqueMembers="true"/>
  </Hierarchy>
 </Dimension>
 <Cube name="Quadrant Analysis">
  <Table name="QUADRANT_ACTUALS"/>
  <DimensionUsage name="Region" source="Region"/>
  <DimensionUsage name="Department" source="Department" />
  <DimensionUsage name="Positions" source="Positions" />
  <Measure name="Actual" column="ACTUAL" aggregator="sum" formatString="#,###.00"/>
  <Measure name="Budget" column="BUDGET" aggregator="sum" formatString="#,###.00"/>
  <Measure name="Variance" column="VARIANCE" aggregator="sum" formatString="#,###.00"/>
<!--   <CalculatedMember name="Variance Percent" dimension="Measures"
formula="([Measures].[Variance]/[Measures].[Budget])*100" /> -->
 </Cube>
</Schema>
```

QUESTIONS :

1. Quels tableaux de la base de données sont utilisés ?
2. Quelles sont les dimensions et quelles colonnes elles représentent ?
3. Quelles sont les indicateurs proposés dans le schéma précèdent ?

NOTE : Le schéma précèdent est disponible sur la section « Manage Data Sources » du « Home ».

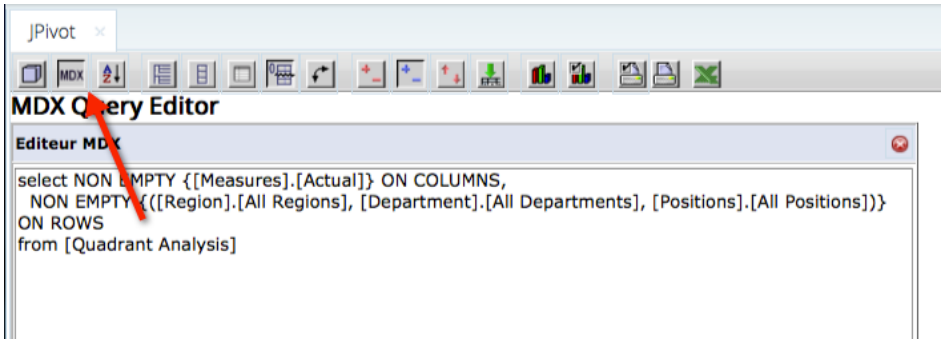


PARTIE 4 : Requêtes MDX

Comment vous avez vu dans le cours, Pentaho utilise le langage MDX (MultiDimensional eXpresions) dont la syntaxe est relativement différente à celle du SQL standard.

```
SELECT      axis specification ON COLUMNS,
            axis specification ON ROWS
FROM        cube_name
WHERE       slicer_specification
```

Pour écrire et lancer nos requêtes MDX nous allons appeler le Plug-in « Editeur MDX » dans l'interface JPivot de Pentaho :



Par exemple, pour extraire les mesures (indicateurs) du cube « Quadrant Analyse », lancer la requête ci-dessous :

```
SELECT {[Measures].MEMBERS} ON COLUMNS
FROM [Quadrant Analysis]
```

Elle est équivalent à :

```
SELECT {AddCalculatedMembers([Measures].Members)} ON COLUMNS
FROM [Quadrant Analysis]
```

La fonction AddCalculatedMembers permet d'agréger les valeurs de tous les indicateurs du cube et est implicite.

PARTIE 5 : Exercices.

1. Montrer les totales des indicateurs par département.
2. Montrer les totales des indicateurs par région et pour le département de Finances.
3. Montrer le budget pour les régions Centre et Sud et pour les départements Ventes et RH.
4. Modifier la requête précédente de façon à montrer deux résultats : a) toutes les régions montrées indépendamment ; et b) toutes les régions confondues.
5. Montrer, pour chaque région et pour chaque département, le budget utilisé pour l'analyste.
6. La information obtenue dans la requête précédente contient plusieurs valeurs nulles. Que pouvez-vous en déduire ? Comment filtrer ces valeurs nulles ?
7. Ordonner les résultats précédents par budget (ordre croissant).
8. Toujours sur la requête précédente, filtrer les analystes qui ont un budget supérieur à 270 000
9. Montrer la moitié de la mesure Variance (1/2*Variance) par département et par région.

Solutions :

Exercice 1 :

```
SELECT {[Measures].MEMBERS} ON COLUMNS,
{[Department].MEMBERS} ON ROWS
FROM [Quadrant Analysis]
```

Exercice 2 :

```
SELECT {[Measures].MEMBERS} ON COLUMNS,
{[Region].MEMBERS} ON ROWS
FROM [Quadrant Analysis]
WHERE ([Department].[Finance])
```

Exercice 3 :

```
SELECT {[Region].[Central], [Region].[Southern]} ON COLUMNS,
{[Department].[Sales], [Department].[Human Resource]} ON ROWS
FROM [Quadrant Analysis]
WHERE [Measures].[Budget]
```

Exercice 4 a :

```
SELECT {[Region].CHILDREN} ON COLUMNS,
{[Department].[Sales], [Department].[Human Resource]} ON ROWS
FROM [Quadrant Analysis]
WHERE [Measures].[Budget]
```

Exercice 4 b :

```
SELECT {[Region].[All Regions]} ON COLUMNS,
{[Department].[Sales], [Department].[Human Resource]} ON ROWS
FROM [Quadrant Analysis]
WHERE [Measures].[Budget]
```

Exercice 5 :

```
SELECT {[Department].CHILDREN} ON COLUMNS,
{[Region].CHILDREN} ON ROWS
FROM [Quadrant Analysis]
WHERE ([Measures].[Budget], [Positions].[Analyst Relations])
```

Exercice 6 :

Il existe des Analystes uniquement dans le département de Marketing et Communications

```
SELECT NON EMPTY({[Department].CHILDREN}) ON COLUMNS,
{[Region].CHILDREN} ON ROWS
FROM [Quadrant Analysis]
WHERE ([Measures].[Budget], [Positions].[Analyst Relations])
```

Exercice 7 :

```
SELECT NON EMPTY({[Department].CHILDREN}) ON COLUMNS,
ORDER ({[Region].CHILDREN}, [Measures].[Budget], ASC) ON ROWS
FROM [Quadrant Analysis]
WHERE ([Measures].[Budget], [Positions].[Analyst Relations])
```

Exercice 8 :

```
SELECT NON EMPTY({[Department].CHILDREN}) ON COLUMNS,
FILTER({[Region].CHILDREN}, ([Measures].[Budget],
[Positions].[Analyst Relations])
> 270000.00) ON ROWS
FROM [Quadrant Analysis]
WHERE ([Measures].[Budget], [Positions].[Analyst Relations])
```

Exercice 9 :

```
WITH MEMBER Measures.HalfVariance AS '([Measures].[Variance])/2'
SELECT {[Department].CHILDREN} ON COLUMNS,
{[Region].CHILDREN} ON ROWS
FROM [Quadrant Analysis]
WHERE ([Measures].[HalfVariance])
```