

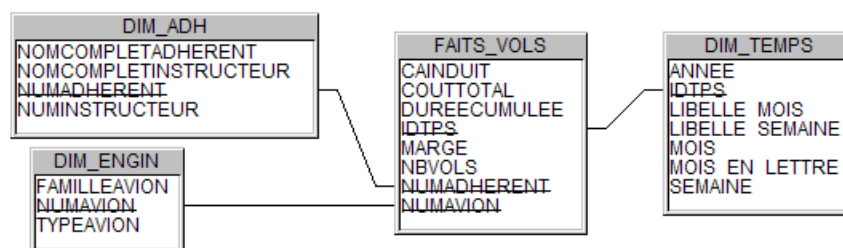
Schéma en étoile, requêtes analytiques avec Oracle OLAP, MDX et XML/A avec Mondrian

1. Objectifs

Il s'agit ici d'implanter un schéma en étoile, celui de la base de données AEROCLUB, dans Oracle. De plus, avec le même SGBD, on abordera les extensions analytiques de SQL.

Les requêtes MDX et XML/A seront abordées avec des outils Open Source : Mondrian, JPivot et Apache derby. Mondrian est un serveur OLAP écrit en java. JPivot est une bibliothèque de balises Java Server Pages qui permet de faire des requêtes OLAP en MDX ou en XML/A et qui peut fonctionner en tant que java portlet. Apache derby est un SGBD relationnel en java compatible SQL 92 qui permet un fonctionnement autonome dans une application Web J2EE. Des éléments de configuration de Mondrian sont fournis pour vous permettre de comprendre voire d'utiliser cet outil Open Source.

2. Création des tables de dimensions et de faits.



La relation universelle du domaine d'étude est :

U(idtps	NUMBER(11,0)	Identifiant de la semaine de l'étude,
	semaine	NUMBER(6,0)	Numéro de la semaine dans une année,
	libelle_semaine	VARCHAR(32)	Concaténation du numéro de la semaine avec l'année,
	mois	NUMBER(6,0)	Numéro du mois dans une année,
	libelle_mois	VARCHAR(32)	Concaténation du numéro de mois avec l'année,
	mois_en_lettre	VARCHAR(32)	Le nom du mois en français,
	annee	NUMBER(6,0)	Année des vols,
	numadherent	NUMBER(6,0)	Numéro dans la base de production de l'adhérent,
	numinstructeur	NUMBER(6,0)	Numéro dans la base de production de l'instructeur,
	nomcompletadherent	VARCHAR(32)	Nom complet de l'adhérent qui n'est pas instructeur,
	nomcompletinstructeur	VARCHAR(32)	Nom complet de l'instructeur suiveur des adhérents,
	numavion	VARCHAR(32)	Identification d'un avion de l'aéro-club,
	typeavion	VARCHAR(32)	Type de l'avion de l'aéro-club,
	familleavion	VARCHAR(32)	Famille de l'avion de l'aéro-club,
	nbvols	NUMBER(6,0)	Recensement des vols,
	dureecumulee	NUMBER(6,0)	Cumul des heures de vols,
	cainduit	NUMBER(38,2)	Chiffre d'affaires induit par les vols,
	couttotal	NUMBER(38,2)	Coût total induit par le vol,
	marge	NUMBER(38,2)	Marge sur les vols concernés
)			

A noter que tous les attributs sont obligatoires (NOT NULL).

Le schéma en étoile va utiliser les optimisations fournies par Oracle.

Les ordres CREATE TABLE et CREATE INDEX auront les options NOLOGGING. Des plans d'exécutions vont être étudiés sur ce schéma. Autant avoir des noms d'index intelligibles. Les clés primaires ne seront pas déclarées dans l'ordre CREATE TABLE mais dans un ordre ALTER TABLE avec l'option USING INDEX avec le nom d'un index créé précédemment. Ainsi peut-t-on choisir le nom des index des clés primaires. Les clés étrangères auront les options RELY DISABLE NOVALIDATE. La table de faits sera partitionnée par rangée selon les années 2000, 2001, 2002.

Nom de la partition	Valeur limite de idtps
Fvoltps2000	60
Fvoltps2001	117
Fvoltps2002	176
Fvolxxxx	400

3. Création des dimensions Oracle et des tables agrégées sous la forme de vues matérialisées.

Afin de faciliter Oracle dans la réécriture de requêtes, proposez l'ordre de création des dimensions `CREATE DIMENSION`. Les dimensions créées devront respecter le tableau suivant.

Dimension / Table	Attribut	Membre	Enfant de (membre de la dimension)	Dépendance (attributs de tables)
Dim_engin	Numavion	Numavion	Typeavion	
Dim_engin	Typeavion	Typeavion	Familleavion	
Dim_engin	Familleavion	Familleavion		
Dim_temps	Libelle_semaine	Semaine	Mois	idtps, semaine
Dim_temps	Libelle_mois	Mois	Annee	
Dim_temps	Annee	Annee		
Dim_adh	Nomcompletadherent	Adherent	Instructeursuiveur	Numadherent
Dim_adh	nomcompletinstructeur	Instructeursuiveur		Numinstructeur

N'oubliez pas de valider la dimension avec la consultation de la vue `mview$_exceptions` après l'ordre `EXECUTE DBMS_OLAP.VALIDATE_DIMENSION('dimension',USER,FALSE,FALSE)` ;

Les tables agrégées à implémenter sous forme de vues matérialisées à rafraîchissement manuel sont : `AGG_ANNEE_VOLS`, `AGG_SUIVEUR_VOLS`, `AGG_TOUT_VOLS`, `AGG_TYPE_VOLS`.

Après la création des tables, utilisez le script `INSERTIONS.sql` pour remplir les tables de données. Vous pouvez plutôt utiliser les tables externes. Étudiez le script `ms_olap_avion.sql` qui décrit comment on a alimenté le schéma en étoile dans une base de données MS Access. Il y a aussi des indications pour un processus ETL sous Talend aussi. Quel est le problème des semaines par rapport aux mois ?

Procédez au rafraîchissement des tables agrégées avec l'ordre `EXECUTE DBMS_MVIEW.REFRESH('agg_xxx_vols')` ;

Demandez l'établissement des statistiques sur votre schéma à l'aide de `EXECUTE DBMS_UTILITY.ANALYZE_SCHEMA('votreLogin','ESTIMATE')` ;

Testez les vues matérialisées à l'aide de requêtes portant sur la table de faits et des informations données par l'AUTO TRACE pour obtenir les plans d'exécution et les statistiques sur la restitution des résultats des requêtes.

```
ALTER SESSION SET QUERY_REWRITE_ENABLED = TRUE;
ALTER SESSION SET QUERY_REWRITE_INTEGRITY = TRUSTED;
SET AUTOTRACE ON
SET TIMING ON
```

La colonne `fact_count` indique le nombre de tuples de la table de faits qui ont été synthétisés par un tuple de la table agrégée. Cette information est utilisée par le serveur OLAP Mondrian, lorsqu'il est configuré pour cela.

Les indicateurs portant sur l'ensemble du recensement des vols.

```
AGG_TOUT_VOLS(id,nbvols,dureecumulee,cainduit,couttotal,marge,fact_count)
```

Les indicateurs annuels des vols.

```
AGG_ANNEE_VOLS(annee,nbvols,dureecumulee,cainduit,couttotal,marge,fact_count)
```

Déclinaison des indicateurs par type d'avion.

```
AGG_TYPE_VOLS(typeavion,nbvols,dureecumulee,cainduit,couttotal,marge,fact_count)
```

Déclinaison des indicateurs par instructeur suiveur.

```
AGG_SUIVEUR_VOLS(numinstructeur,nbvols,dureecumulee,cainduit,couttotal,marge,fact_count)
```

4. Stratégies de jointures STAR et STAR TRANSFORMATION.

La stratégie de jointure STAR s'appuie sur un index composé UNIQUE BTREE sur toute la clé primaire de la table de faits. De plus à cause de la règle de l'attribut leader, elle nécessite la création d'autres index composés sur la table de faits. Tous ces index seront locaux aux partitions. Sur les tables de dimensions, des index peuvent être proposés pour faciliter des recherches. Ils peuvent inclure la fonction UPPER. Cette optimisation est efficace si le nombre de tables de dimension est limité, la table de fait est dense (il y a un tuple pour toutes les combinaisons des dimensions) et que la cardinalité des clés étrangères est élevée.

La stratégie de jointure STAR TRANSFORMATION s'appuie sur des index BITMAP simple sur les parties de clé primaire de la table de faits. Les index UNIQUE sont des BTREE et sont sur les clés primaires et les clés candidates. Des index BITMAP sur les attributs informations des tables de dimensions peuvent accélérer les recherches. Cette optimisation est efficace si le nombre de tables de dimension est grand, la table de fait est remplie de manière non dense et que la cardinalité des clés étrangères est faible (les clés étrangères ont peu de valeurs distinctes).

Etablissez les deux politiques d'indexation à l'aide de scripts CIDXSTAR.sql et CIDXSTARTRANSFORMATION.sql. puis testez à l'aide d'une requête (voir STARQUERY.sql) avec éventuellement des conseils chaque stratégies. Vous créerez aussi les scripts de destruction des indexes DIDXSTAR.sql et DIDXSTARTRANSFORMATION.sql.

5. Requêtes analytiques de SQL 1999.

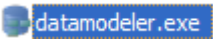
- 1- Par type d'avion et par instructeur suiveur, donnez le chiffre d'affaires généré en mars 2000.
- 2- Par type d'avion et par instructeur suiveur, donnez le chiffre d'affaires généré en mars 2000, les sous totaux par types d'avion et le chiffre d'affaire total.
- 3- Par type d'avion et par instructeur suiveur, donnez le chiffre d'affaires généré en mars 2000, les sous totaux par types d'avion, par instructeurs suiveurs et le chiffre d'affaire total.
- 4- Par type d'avion et par instructeur suiveur, ne donnez qu'en mars 2000, les sous totaux par types d'avion, par instructeurs suiveurs du chiffre d'affaires généré.
- 5- Par type d'avion et par instructeur suiveur, donnez qu'en mars 2000, le rang global et le rang par type d'avion du chiffre d'affaires généré.
- 6- Par type d'avion et par instructeur suiveur, donnez le chiffre d'affaires généré en mars 2000, les sous totaux par types d'avion et le chiffre d'affaire total ainsi que le rang global et le rang par type d'avion du chiffre d'affaires généré.

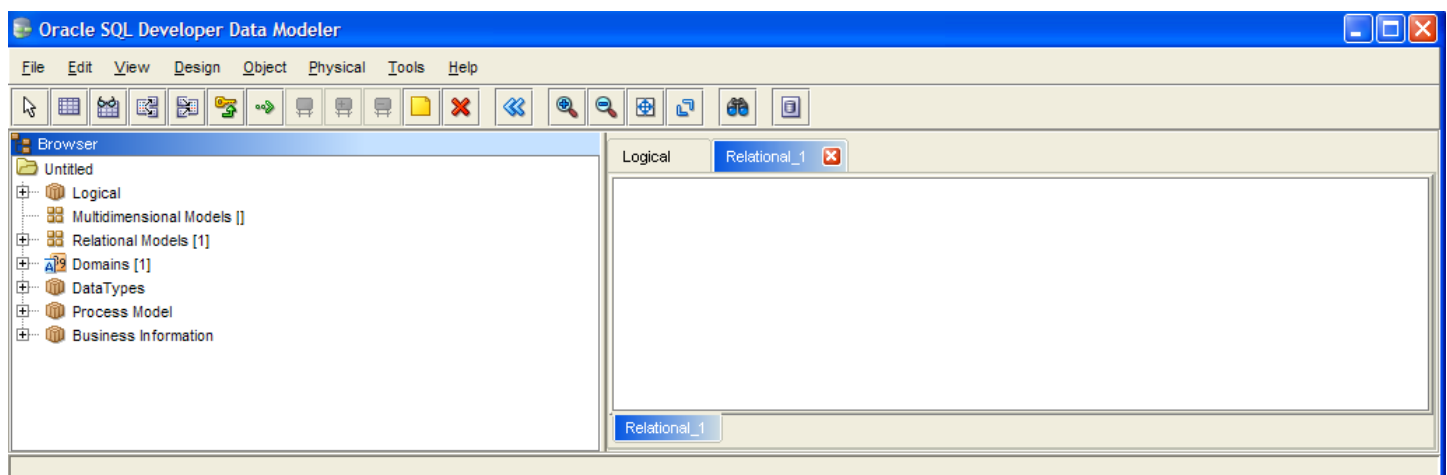
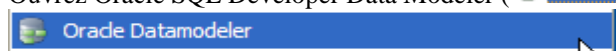
Utilisez notamment les extensions analytiques GROUP BY CUBE, GROUP BY ROLLUP, GROUP BY GROUPING SETS, RANK() OVER (ORDER BY), RANK() OVER (PARTITION BY ORDER BY) de SQL 1999 sur la requête suivante :

```
SELECT typeavion, nomcompletinstructeur, SUM(CaInduit) AS "Chiffre d'affaires"
FROM DIM_ENGIN, FAITS_VOLS, DIM_ADH, DIM_TEMPS
WHERE DIM_ENGIN.numavion=FAITS_VOLS.numavion
      AND DIM_TEMPS.idtps=FAITS_VOLS.idtps
      AND DIM_ADH.numadherent=FAITS_VOLS.numadherent
      AND annee=2000 AND mois_en_lettre='Mars'
GROUP BY typeavion, nomcompletinstructeur
ORDER BY typeavion, nomcompletinstructeur;
```

Essayer d'obtenir le même résultat avec le SQL 1992 (clause UNION ALL). Pour comparer, utilisez l'AUTO TRACE.

6. Modélisation ROLAP et HOLAP avec Oracle SQL Developer Data Modeler.

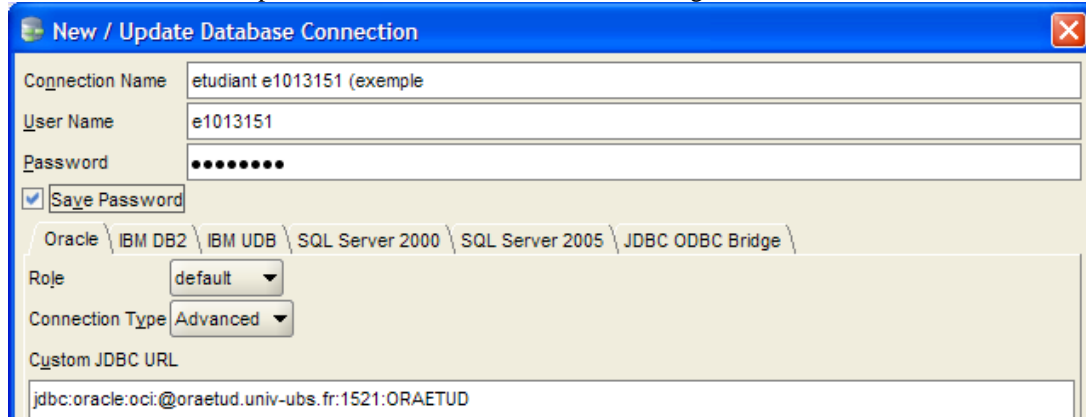
Ouvrez Oracle SQL Developer Data Modeler (). Le menu Démarrer permet aussi de l'ouvrir :



6.1 – Modélisation ROLAP via Reverse Engineering

La commande File / Import / Data dictionary permet de se connecter à un SGBD pour faire du reverse engineering. Par défaut seul le pilote JDBC Oracle est déclaré.

La connexion à un compte avec authentification radius est configurée comme ceci :



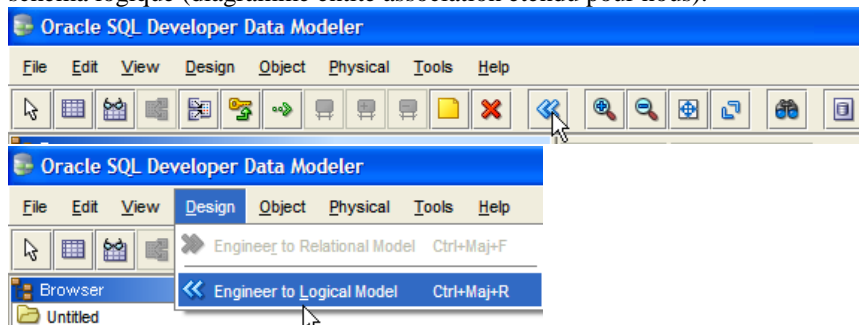
Notamment, l'URL JDBC doit être : `jdbc:oracle:oci:@oraetud.univ-ubs.fr:1521:ORAETUD`

Vous pouvez importer le schéma physique à l'aide de l'assistant qui vous permet de choisir notamment les tables, séquences, les dimensions :

<input checked="" type="checkbox"/>	DUBOISM	AGG_ANNEE_VOLS
<input checked="" type="checkbox"/>	DUBOISM	AGG_SUVEUR_VOLS
<input checked="" type="checkbox"/>	DUBOISM	AGG_TOUT_VOLS
<input checked="" type="checkbox"/>	DUBOISM	AGG_TYPE_VOLS
<input checked="" type="checkbox"/>	DUBOISM	DIM_ADH
<input checked="" type="checkbox"/>	DUBOISM	DIM_ENGIN
<input checked="" type="checkbox"/>	DUBOISM	DIM_TEMPS
<input checked="" type="checkbox"/>	DUBOISM	FAITS_VOLS
<input checked="" type="checkbox"/>	DUBOISM	DIM_TEMPS_IDTPS_SEQ
<input checked="" type="checkbox"/>	DUBOISM	DIM_ADH
<input checked="" type="checkbox"/>	DUBOISM	DIM_ENGIN
<input checked="" type="checkbox"/>	DUBOISM	DIM_TEMPS

Affectez un type à chaque table (fait, dimension, résumé) afin d'obtenir le schéma relationnel (schéma logique pour nous) représenté à la page suivante.

Disposant du schéma physique et relationnel (schéma logique pour nous), vous pouvez obtenir par reverse engineering le schéma logique (diagramme entité association étendu pour nous).



6.2 – Modélisation HOLAP (Espace Analytiques de travail d'Oracle)

Faire les deux tutoriels [Tutoriel Oracle Education - Building OLAP 12c cubes](#) et [Tutoriel Oracle Education - Querying OLAP 12c cubes](#). De plus, étudiez l'exemple de modélisation HOLAP Oracle avec Oracle SQL Developer Data Modeler nommé Global11_demo disponible sur le bureau à distance. Étudiez les scripts `CDimension_calendar_day_2012.sql`, `CDimension_calendar_week_2012.sql`, `CDimension_calendar_week_2012_DIM_TIM_CALENDAR_WEEK.sql` et `OLAP_CALENDAR_WEEK.sql`. Que peut-on dire des dimensions de type TIME dans les solutions ORACLE_AW ? Prégénérer une telle dimension TIME via un assistant de Oracle SQL Developer récent.

Créez un nouveau schéma multidimensionnel. Puis déduisez le schéma multidimensionnel à partir de votre modélisation ROLAP (et des dimensions récupérées !) à l'aide de la commande Engineer From Oracle Model.

Créez une vue relationnelle d'accès vers le cube HOLAP Oracle nommée `SQLAccessToFAITS_VOLS` comprenant les indicateurs additifs et les identifiant des dimensions en utilisant l'assistant SQL Access to Oracle AW pour générer le code comprenant l'appel à la fonction `OLAP_TABLE`.

Exportez votre modèle multidimensionnel vers un cube HOLAP Oracle avec la commande File / Export / To Oracle AW.

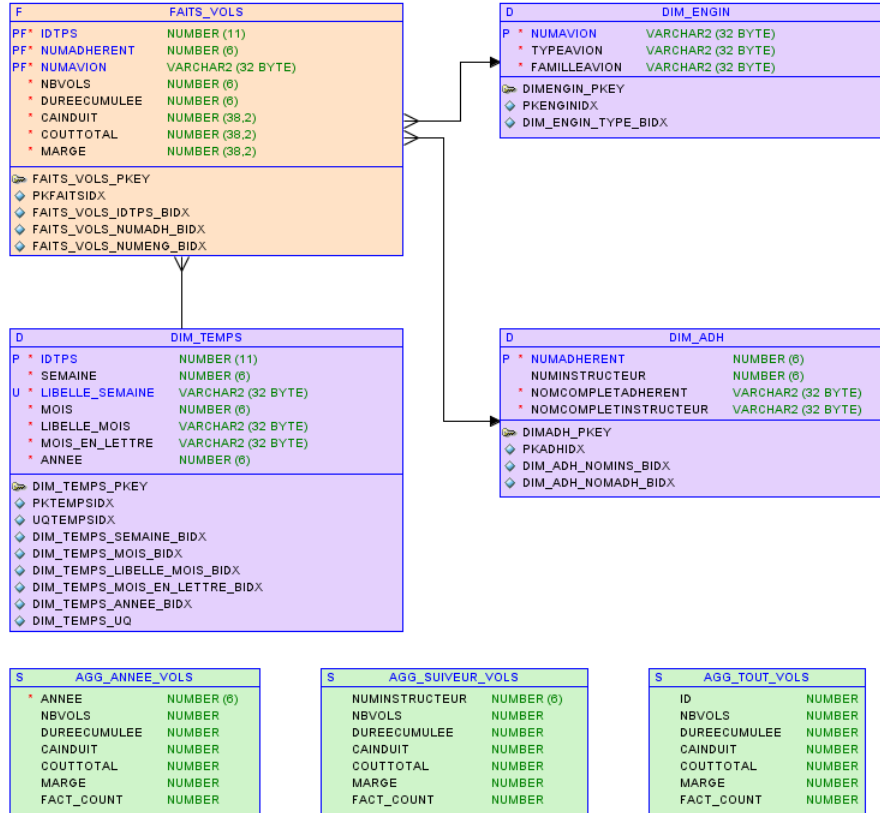


Schéma logique (pour nous) ou relationnel ROLAP avec Oracle SQL Developer Data Modeler.

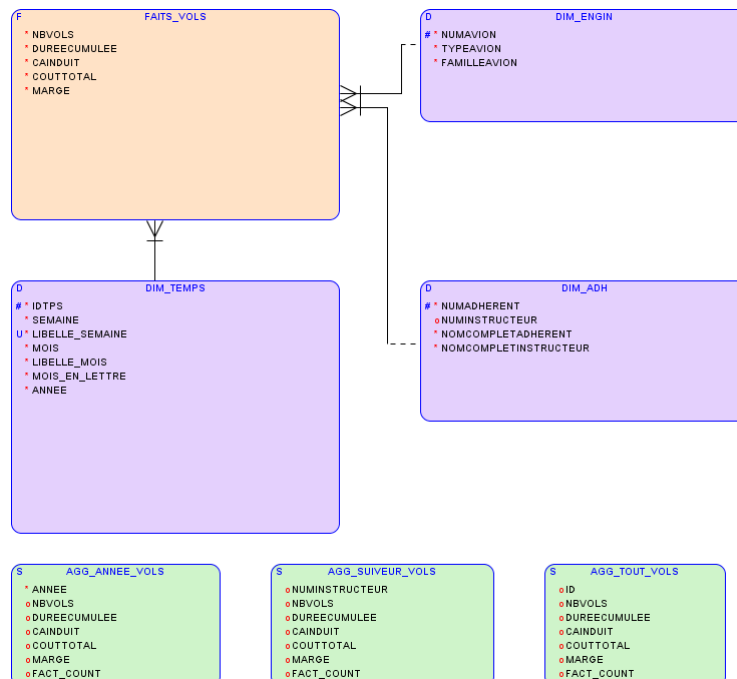


Schéma conceptuel (pour nous) ou logical ROLAP avec Oracle SQL Data Modeler.

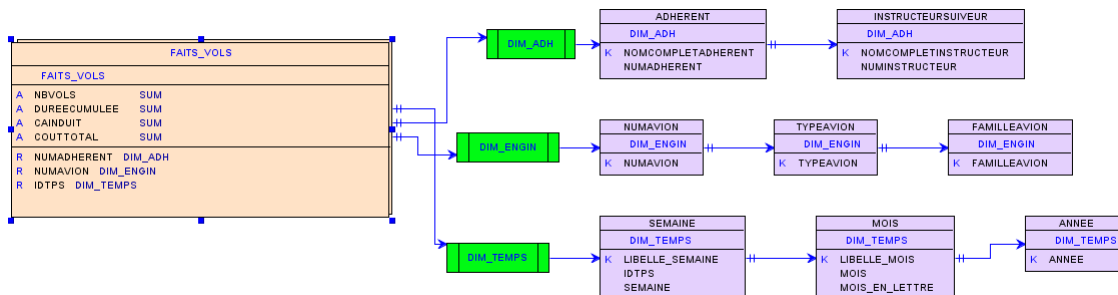
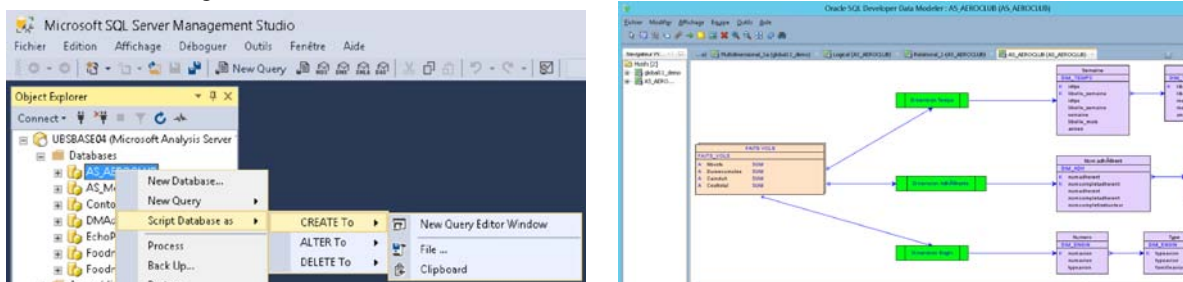


Schéma multidimensionnel HOLAP Oracle avec Oracle SQL Developer Data Modeler.

Alimentez votre cube HOLAP Oracle avec l'outil Oracle Analytic Workspace Manager ou avec Oracle SQL Developer et créez les vues relationnelles sur cet espace analytique de travail.

6.3 – Modélisation HOLAP SQL Server Analysis Services de Microsoft

Sur le serveur statdeci9.univ-ubs.fr via une connexion RDS (bureau à distance) vous pouvez déduire un schéma conceptuel ROLAP à partir du serveur statdeci8.univ-ubs.fr port 1433 base de données AEROCLUB_OLAP avec pour utilisateur AEROCLUB_OLAP/P@ssw0rd. Il vous faudra reconstruire tout le schéma multidimensionnel pour pouvoir l'exporter en script XML/A pour la création d'un cube Analysis Services. Pour cela, il faut déclarer dans Oracle SQL Developer Data Modeler le pilote JDBC jtds. Déclarer la connexion authentification Windows vers SQL Server. Puis importer le script AS_AEROCLUB_OLAP.xml qui se trouve dans z:\prof\mdubois\ia3\OLAP. Il a été généré via le logiciel SQL Server Management Studio.



7. Requêtes MDX.

Sur le bureau à distance statdeci9.univ-ubs.fr, le site est : <http://ubsbase04:9090/mondrian/>. Le site sur le réseau de l'UBS http://www-i.univ-ubs.fr/etud/projets/p_07_dubois_m_csdia3/ pour ce qui concerne JPivot, ne semble pas marcher.

Le MDX (Multi Dimensional Expression) est le langage de requêtage proposé par Microsoft pour accéder aux bases multidimensionnelles. Proposez les requêtes équivalentes aux requêtes SQL 1999 en MDX. Les fonctions CrossJoin, Union, Hierarchize et Children pourront être utilisées. Il vous faudra vérifier les résultats avec le SQL 1992 de Apache Derby. Utilisez plusieurs outils d'interrogation afin de connaître les possibilités de Mondrian. Pour vous aider, voici une présentation de MDX. Exécutez et analysez toutes les requêtes MDX, tout en lisant les explications.

```
SELECT FROM Vols
```

Requête MDX 0 : la plus simple requête MDX, qui ne marche pas avec la JSP « Basic interface for MDX queries ».

```
SELECT
  {[Measures].[Nombre de vols]}
ON COLUMNS,
  {[([Temps].[Tout],
    [Avions].[Tous les avions],
    [Adherents].[Tous les adherents])]}
ON ROWS
FROM Vols
```

Requête MDX 1 : pour le niveau le plus agrégé, donner le nombre de vols.(=Requête 0)

Results:

[Mesures].[Nombre de vols]
3,160

```
SELECT
  {[Mesures].[Nombre de vols],
  [Mesures].[Duree cumulee],
  [Mesures].[Chiffre d'affaires induit],
  [Mesures].[Cout total induit],
  [Mesures].[Marge induite]}
ON COLUMNS,
  {[Temps].[Tout].[2001].[Mars].[11],
  [Avions].[Tous les avions].[PIPER].[Piper Arrow].[F GPFA],
  [Adherents].[Tous les adherents].[Michel DUBOIS].[Eric Martin]}
ON ROWS
FROM Vols
```

Requête MDX 2 : pour une cellule du cube, donnez tous les indicateurs.

7.1 – L'environnement de Travail du serveur OLAP Mondrian

Le serveur Mondrian peut recevoir des requêtes MDX :

- à partir de l'utilitaire en ligne de commande cmdrunner ;
- à partir de l'utilitaire graphique Mondrian Workbench ;
- à partir de la Java Server Pages http://www-i.univ-ubs.fr/etud/projets/p_07_dubois_mcsdia3/MDadhoc.jsp; (contient toutes les requêtes)
- à partir de la Java Server Pages http://www-i.univ-ubs.fr/etud/projets/p_07_dubois_mcsdia3/testpage.jsp?query=MDmondrian (portlet JPivot : interface web plus léchée)
- à partir de la Java Server Pages http://www-i.univ-ubs.fr/etud/projets/p_07_dubois_mcsdia3/MDxmlaTest.jsp (via le protocole XML/a).

Le SGBD Apache Derby peut recevoir des requêtes SQL :

- à partir du serveur OLAP Mondrian (données nécessaires au résultat des requêtes MDX) interrogé par les applications web ;
- à partir de la Java Server Pages http://www-i.univ-ubs.fr/etud/projets/p_07_dubois_mcsdia3/MDQuery.jsp (contient des requêtes).

```
java -jar workbench.jar
```

Ligne de commande 1 Appel à l'utilitaire graphique Mondrian Workbench

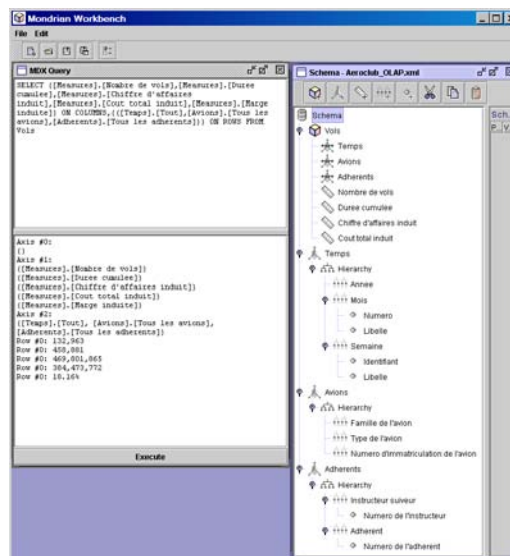


Figure 1 L'utilitaire graphique Mondrian Workbench


```
java -jar cmdrunner.jar -p aeroclub.properties "SELECT {[Measures].[Nombre de vols],[Measures].[Duree cumulee],[Measures].[Chiffre d'affaires induit],[Measures].[Cout total induit],[Measures].[Marge induite]} ON COLUMNS, {[([Temps].[Tout],[Avions].[Tous les avions],[Adherents].[Tous les adherents])}] ON ROWS FROM Vols"
```

Ligne de commande 2 L'utilitaire CMDRUNNER de Mondrian : la requête MDX peut aussi être entrée sans les guillemets après le prompt ">" et doit être terminée par un ";"

Sous linux, dans le répertoire /ubs/forum/vannes/prof/lcsd01/MD/linux, les commandes workbench.sh et cmdrunner.sh permettent de lancer les deux outils qui par défaut se connectent au compte oracle mdubois.

Sous windows, dans le répertoire G:\vannes\prof\lcsd01\MD\win32, les commandes workbench.bat et cmdrunner.bat permettent de lancer les deux outils qui par défaut se connectent au compte oracle mdubois.

A noter que l'utilitaire cmdrunner.xxx accepte les requêtes MDX terminées par un point virgule. Faire -h pour l'aide. Il est configuré pour proposer un code SQL de génération d'une table d'agrégats souhaitable pour la requête MDX.

La base de données hébergée par le schéma Oracle mdubois est plus volumineuse que celle accessible par les applications web. Par contre, ces dernières ont en plus l'exemple FoodMart.

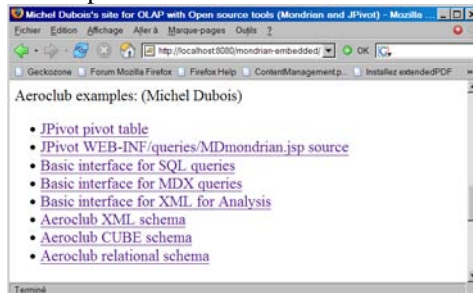


Figure 2 Index du site web du TP

Test Query uses Mondrian OLAP

Barre JPivot

Dimensions		Mesures		Mesures	
Promotion Media	Product	Unit Sales	Store Cost	Store Sales	
All Media	All Products	266 773	225 627,23	565 238,13	

Slicer: [Year=1997] Filtre Cellule Table

[back to index](#)

Figure 3 Présentation succincte de la portlet JPivot

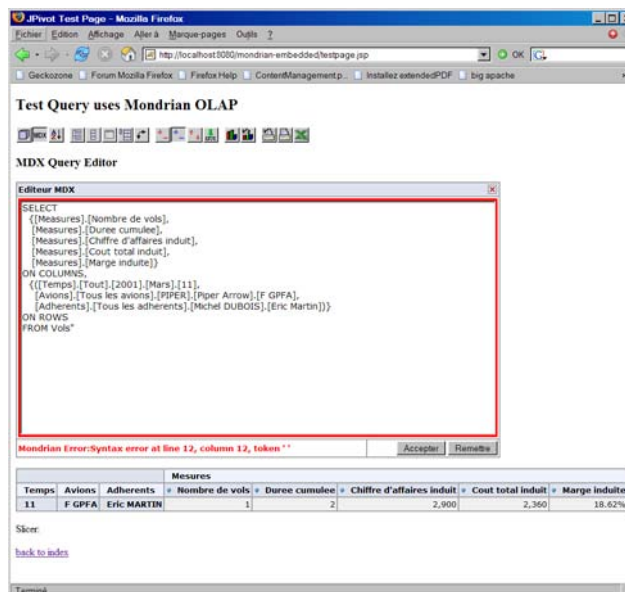


Figure 4 Edition MDX par la portlet/JSP JPivot

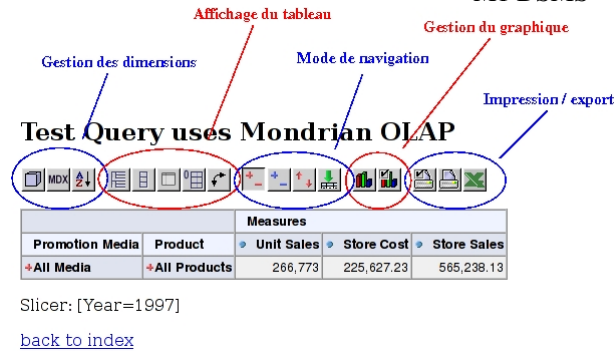


Figure 5 Les autres possibilités de la portlet/JSP JPivot

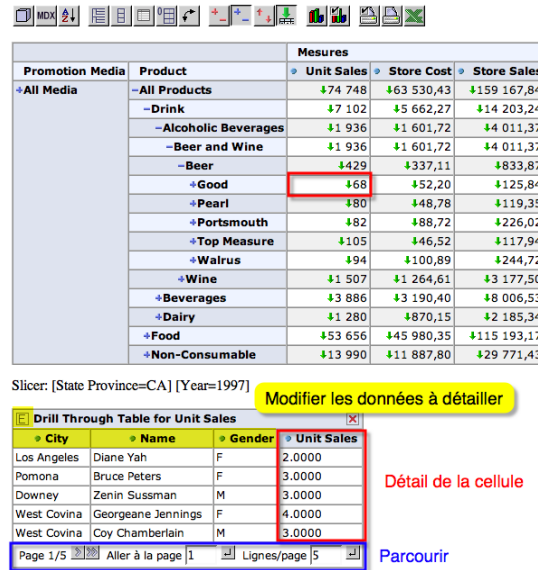


Figure 6 Visualisation des données originales (Drill Through) avec la portlet/JSP JPivot

Le premier bouton permet d'ajouter ou d'enlever des champs (colonnes ou lignes) ou des filtres. Il y a trois sections : une pour les dimensions affichées sur les colonnes, une pour celles affichées sur les lignes, et une pour les autres, qui peuvent éventuellement servir comme filtre. Pour déplacer les dimensions d'une section à une autre, il faut utiliser les icônes à gauche du nom de la dimension (). En cliquant sur une dimension, vous pouvez sélectionner les valeurs qui seront affichées ou pas. Les dimensions qui ne sont pas sélectionnées en tant que colonnes ou lignes sont utilisées comme filtre. Finalement, vous pouvez changer l'ordre des dimensions en utilisant les flèches (). Il est à noter qu'après chaque modification du cube, il faut la valider à l'aide du bouton "OK" en bas du formulaire pour que les modifications prennent effet.

Il est possible à tout moment de visualiser ou de modifier une requête MDX via le bouton de la barre de JPivot.

Le bouton permet de trier les données selon sa convenance.

Les boutons sont dédiés à des changements de présentation, plus particulièrement, la fusion des lignes ou des colonnes.

Le bouton permet de supprimer les cellules dont la valeur est nulle. Le bouton quant à lui permet d'interchanger les colonnes et les lignes.

Les boutons sont dédiés à la navigation. L'appui sur le "+" permet de faire un "drill-down" (zoomer sur les données pour découvrir de nouvelles sous catégories), l'appui sur le "-" permet de faire un "roll-up" (agréger les données sur une catégorie commune). Il est possible lors du "drill-down" d'être en mode remplacement afin de supprimer les lignes de plus haut niveau. Le bouton permet de faire du "drill-through". Ce qui permet d'afficher des flèches vertes sur les données et ainsi d'avoir une répartition détaillée de la donnée suivant les différentes dimensions.

Le bouton permet de représenter à l'aide d'un graphique les données que vous venez d'agréger. Pour choisir le type de graphique et éventuellement changer ses propriétés, il faut cliquer sur le bouton .

Les trois derniers boutons permettent de définir les options d'impression PDF et d'exporter en PDF ou en XLS (MS Excel) les résultats.

Pour une aide additionnelle sur JPivot, vous pouvez consulter : <http://blog.developpez.com/talend/p2947/technologie/open-source/navigation-multidimensionnelle-avec-jpi/>

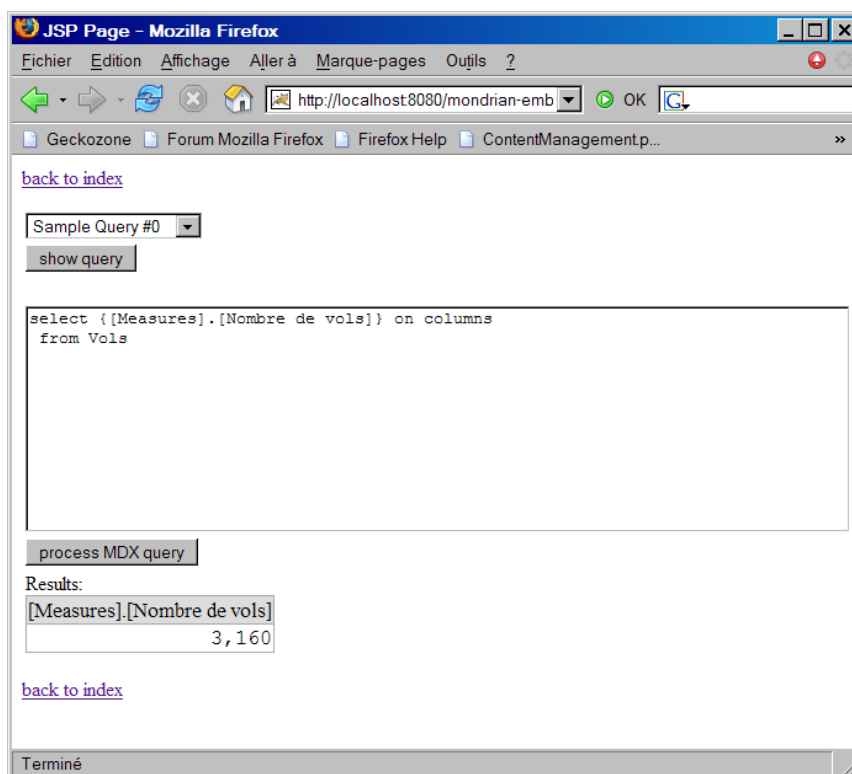


Figure 7 La Java Server Page « Basic interface for MDX queries » pour requêter en MDX

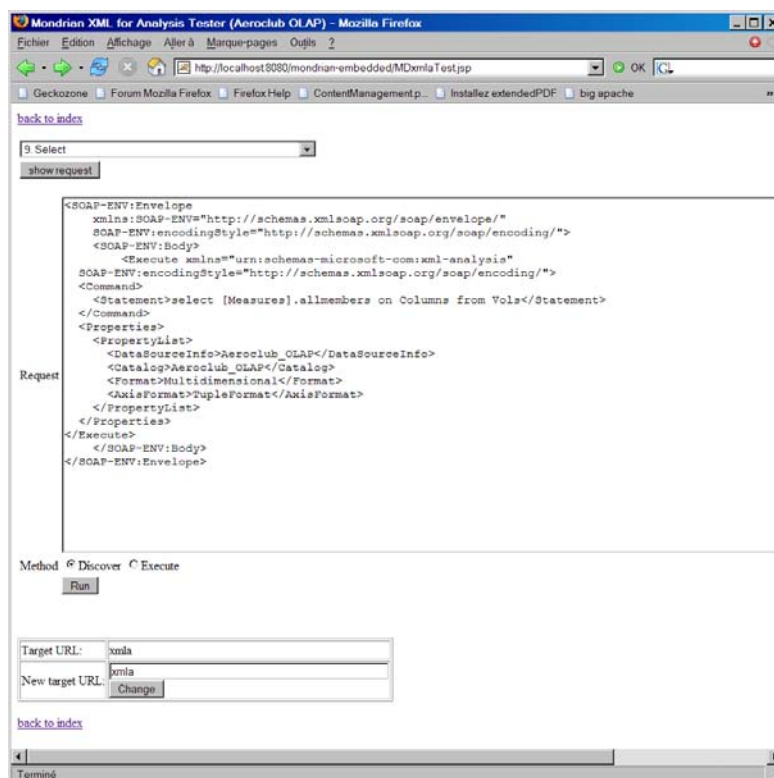
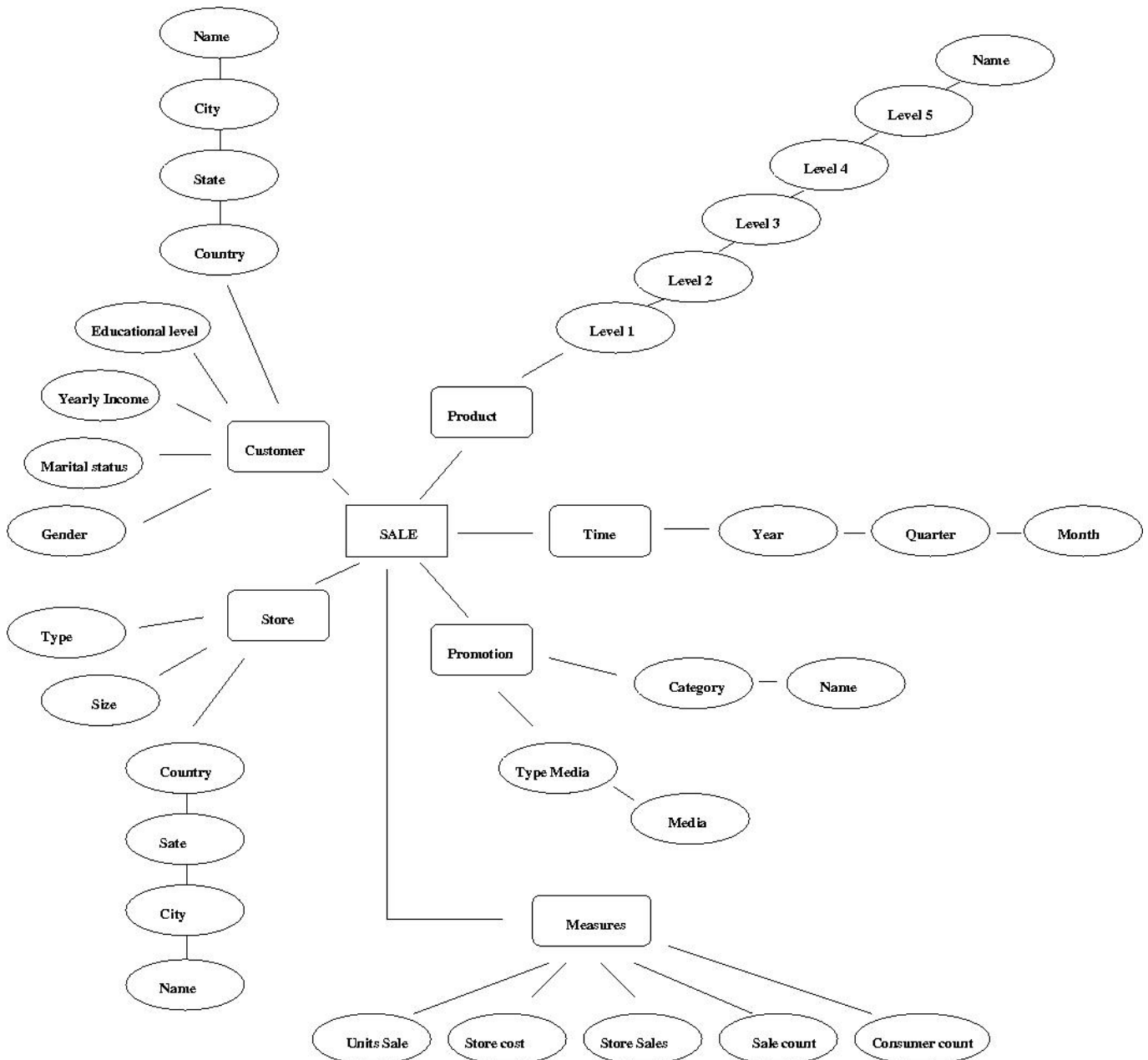


Figure 8 Requête MDX avec le protocole XML/A dans la JSP « Basic interface for XML/A queries »

L'implémentation MDX de Mondrian est assez complète pour un produit Open Source. Mais elle se limite aux requêtes. La définition des cubes ne se fait pas à l'aide des instructions MDX comme sous SSAS mais à l'aide d'un fichier XML de définition d'un schéma. Mondrian Workbench permet d'éditer un tel schéma. A noter que Mondrian est livré avec une base de données très réaliste (FoodMart) qui était livrée avec SSAS 2000 et qui vous permettra de vous entraîner avec un exemple taille réelle. Cependant pour un premier apprentissage, le petit exemple AeroClub a été jugé préférable. De plus, il est préférable de travailler le TP dans un environnement Web dont le point d'entrée est : http://www-i.univ-ubs.fr/etud/projets/p_07_dubois/csdia3/

Sur le schéma Foodmart, reproduisez les graphiques qui permettent de répondre aux questions suivantes:



Identifiez sur le schéma les dimensions, les différents types d'attributs, et les mesures.

1. Quel est le produit le plus vendu en 1997 ?
difficulté: choix des dimensions et de la mesure. [voir la solution](#)
2. Quel est la répartition des ventes entre les différents type de boisson en 1997 ?
difficulté: navigation + choix du graphique. [voir la solution](#)
3. Y-a-t-il une différence hommes/femmes dans répartition de la consommation entre les différents types de produits ?
difficulté: tableau croisé. [voir la solution](#)

4. Y-a-t-il des variations de consommation saisonnières ? mensuelles ?
difficulté: tableau croisé + choix graphique. [voir la solution](#) [voir la solution](#)
5. Quelle est la distribution des revenus des consommateurs en Californie (CA) ?
difficulté: tri dimension + choix mesure + filtrage. [voir la solution](#)

7.2 – Quelques définitions : dimensions, hiérarchies, niveaux, propriétés et membres

Nous allons ici définir un certain nombre de termes et concepts.

Le MDX (Multi Dimensional eXpression) fait partie des spécifications OLE DB for OLAP. D'autres éditeurs comme SAS Institute (SAS OLAP Server) le supportent.

Ce langage est fait pour naviguer dans les bases multidimensionnelles et requêter tous les objets (dimensions, hiérarchies, niveaux, membres et cellules) pour obtenir entre autre (basiquement) une représentation sous forme de tableaux croisés. Il en découle une approche très hiérarchisée.

Tout d'abord un « CUBE » est composé de « DIMENSIONS ». Une dimension peut contenir de 1 à N HIERARCHIES/ « HIERARCHY ». Par exemple, la dimension temps peut contenir 2 hiérarchies. La première pourrait être le « temps par semestre » (Année / Semestre / Mois), la deuxième pourrait être le « temps par trimestre » (Année / trimestre / Mois).

Une Hiérarchie (H) est composée de NIVEAUX/LEVELS (L). Un Niveau est finalement un des attributs de votre base de données. L'essentiel est que la séquence de niveaux réponde à une logique de navigation. Exemple : La hiérarchie « temps par trimestre » peut être composée des niveaux Année, Trimestre et Mois (sens agrégé->détail).

Un niveau est lui-même composé de MEMBRES/MEMBERS. Les membres sont les valeurs détectée par le moteur OLAP et stocké dans les méta données. Exemple : les membres du niveau « Mois » sont « Janvier », « Février », « Mars », « Avril » ...

De plus, un niveau peut comporter des PROPRIETES/PROPERTIES. Elles correspondent aux attributs déterminés par le niveau et qui complètent sa description. Ces informations descriptives ne participent pas à la navigation. Exemple : le sexe d'une personne, son adresse.

En résumé, l'arborescence suivante résume un peu l'idée du modèle :

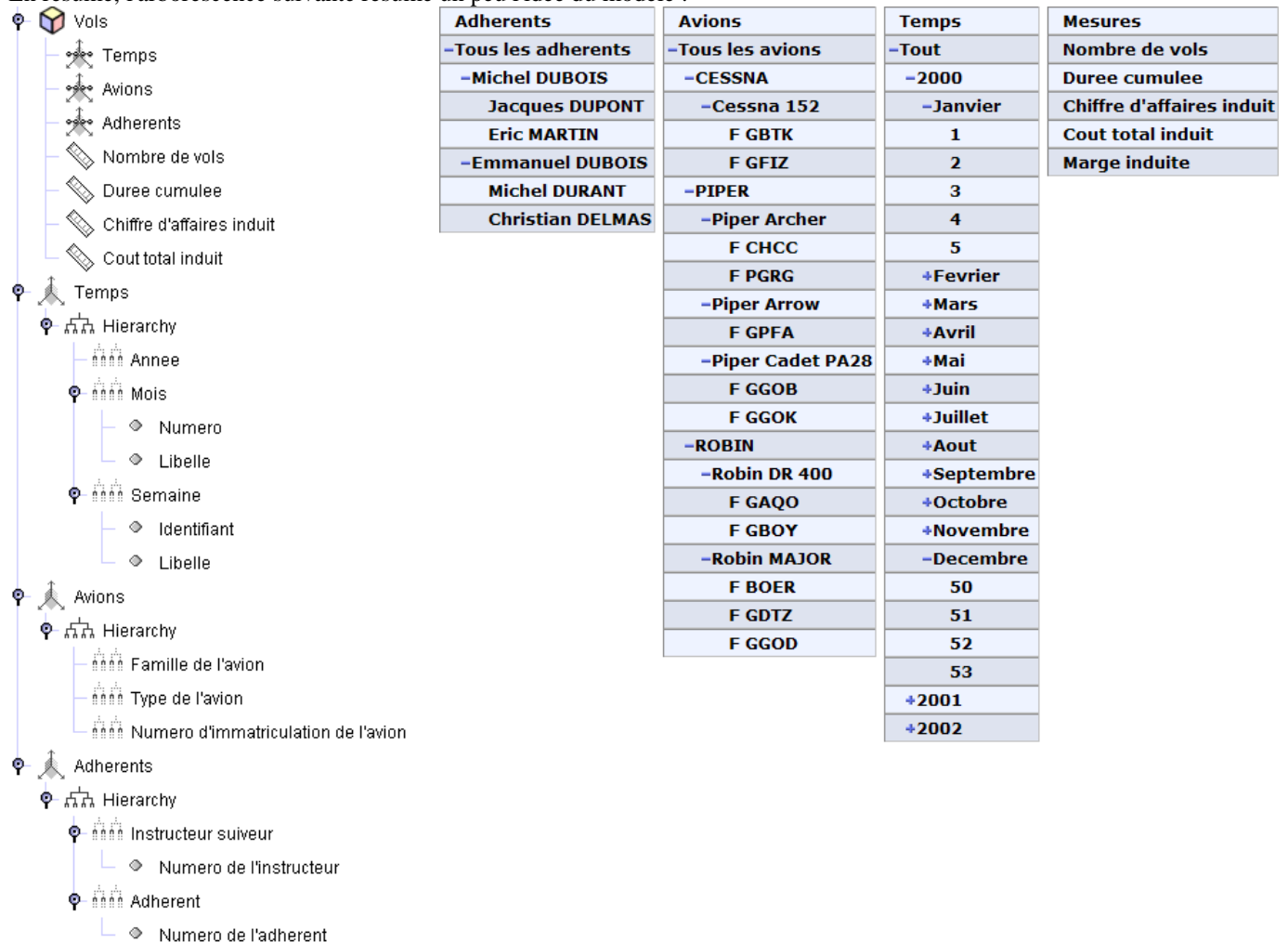


Figure 9 La structure hiérarchique d'un cube (à gauche) et les membres (à droite)

Dans le schéma exemple, la propriété "Numéro" complète la description des membres du niveau [Mois] de la hiérarchie / dimension [Temps]. Remarquons aussi la présence de 5 autres propriétés sur les hiérarchies / dimensions [Adherents] et [Temps]. Les mesures forment naturellement une dimension nommée [mesures] dans chaque cube. Le reste de l'arborescence se comprend aisément.

Le nom d'un objet OLAP (cube, dimension, hiérarchie, niveau et membre) doit être entre crochets s'il contient un espace, s'il commence par un chiffre ou s'il est identique à un mot réservé du langage. Si ce n'est pas le cas, on peut toujours mettre les crochets. C'est même conseillé pour faire la différence entre objet OLAP et mot clés de MDX.

Lorsqu'il y a ambiguïté, il faut qualifier l'objet. Par exemple, il y a plusieurs mois de mars : 2000, 2001 et 2002. Il faut utiliser [2002].[Mars]. Sinon, c'est la première occurrence qui sera utilisé ([Mars] = [2000].[Mars]) sous MS SQL Server Analysis Service (SSAS) ou il y a une erreur avec Mondrian.

Une expression MDX peut renvoyer une valeur numérique, une chaîne de caractères ou une valeur manquante (mot clé NULL). Les opérateurs numériques (+, -, *, /, ^) et alphanumériques (+) peuvent être utilisés. Il n'y a pas de conversion de type donc on ne peut combiner les opérateurs numériques et alphanumériques.

7.3 – Membres

Un Membre est donc une réalisation d'un niveau. Un membre est un élément d'une dimension représentant une ou plusieurs occurrences de données. Dans une dimension, un membre correspond à un ou plusieurs enregistrements de la base de données sous-jacente dont la valeur de cette colonne tombe sous cette catégorie. Un membre est le niveau de référence le plus bas lors de la description des données des cellules d'un cube. Les ENFANTS/CHILDREN d'un membre sont tous les membres du niveau immédiatement après.

On représente ici tous les membres du niveau « Famille de l'avion » : ✈CESSNA, ✈PIPER, ✈ROBIN (✈AUTRE est présent sur le compte Oracle ops@dubois). Nous pouvons illustrer l'utilisation de Membres dans des requêtes simples :

```
SELECT
{[AVIONS].[TOUS LES AVIONS].[PIPER]}
  ON COLUMNS
FROM [VOLS]
/* Resultat
Axis #0:
{}
Axis #1:
{[Avions].[Tous les avions].[PIPER]}
Row #0: 30,537
*/
```

Requête MDX 3

```
SELECT
{[AVIONS].[TOUS LES AVIONS].[PIPER].[PIPER ARCHER].[F CHCC]}
  ON COLUMNS
FROM [VOLS]
/* Resultat
Axis #0:
{}
Axis #1:
{[Avions].[Tous les avions].[PIPER].[Piper Archer].[F CHCC]}
Row #0: 6,459
*/
```

Requête MDX 4

Les résultats sont donnés par rapport à la Mesure par défaut.

7.4 - Tuples

Un TUPLE est une coordonnée dans le cube. Il s'exprime par une suite de 1 ou N Membres entre parenthèse :

```
([AVIONS].[TOUS LES AVIONS].[PIPER].[PIPER ARCHER].[F CHCC],
[TEMPS].[TOUT].[2001].[MARS].[11],
```

Finalement, la représentation d'une cellule dans un cube suit une représentation mathématique (géométrique) en (x,y,z) s'il n'y a que trois axes. Un tuple identifie une cellule d'un cube. Il est composé d'un membre (référéncé de manière explicite ou implicite) provenant de chaque hiérarchie contenue dans le cube. Ainsi, les composantes d'un tuple doivent être issues de dimensions différentes. Si aucun membre d'une hiérarchie spécifique n'est référéncé de manière explicite dans le tuple, le membre par défaut de cette hiérarchie est implicitement inclus dans le tuple. Vous pouvez noter que je ne cite pas les autres dimensions dans mon tuple, ce qui veut dire que je prends par défaut comme coordonnée de référence le membre par défaut (la plupart du temps, le membre [all]).

Quelques exemples :

```
SELECT
{
([AVIONS].[TOUS LES AVIONS].[PIPER].[PIPER ARCHER].[F CHCC],
[TEMPS].[TOUT].[2001].[MARS].[11],
[MEASURES].[NOMBRE DE VOLS])
}
ON COLUMNS
FROM [VOLS]
/* Resultat
Axis #0:
{}
Axis #1:
{[Avions].[Tous les avions].[PIPER].[Piper Archer].[F CHCC],
[Temps].[Tout].[2001].[Mars].[11], [Measures].[Nombre de vols]}
Row #0: 64
*/
```

Requête MDX 5 : pour un avion et une semaine, donnez le nombre de vols, tous les adhérents confondus.

```
SELECT
{([AVIONS].[TOUS LES AVIONS].[PIPER].[PIPER ARCHER].[F CHCC])}
ON COLUMNS
FROM [VOLS]
/* Resultat
Axis #0:
{}
Axis #1:
{[Avions].[Tous les avions].[PIPER].[Piper Archer].[F CHCC]}
Row #0: 6,459
*/
```

Requête MDX 6 : pour un avion, donnez le nombre de vols (sur toute la période d'étude, tous les adhérents confondus)

La dernière requête reprend une des requêtes sur les membres justes en ajoutant des parenthèses. Dans la mesure où dans les premiers exemples nous avions des requêtes sur des membres seuls, elles étaient implicites. Dans la pratique je vous recommande de vous attacher à les mettre pour une relecture facile de vos requêtes.

7.5 – Set (jeu de membres)

Un jeu est un ensemble de tuples en première approximation. Là encore j'aime bien le rapprochement avec la notion d'ensemble mathématique, car vous pouvez le voir comme un ensemble de n-uple discret, ou bien comme une plage. Plus exactement, un jeu est une collection ordonnée (ce qui n'est pas le cas d'un ensemble) de tuples dont les composantes doivent partager les mêmes dimensions qui apparaissent dans le même ordre dans la définition de chaque tuple. La fonction `Item` permet d'obtenir le tuple en fonction de la position ordinale spécifiée en argument (0 permet d'obtenir le premier élément). Le jeu commence par une accolade { dans le quel on liste les tuples en les séparant par une « , » (énumération) ou par « : » (plage), et se termine par une accolade appariée }. Certaines fonctions retournent un jeu. Dans ce cas là, les accolades ne sont pas nécessaires.

```
SELECT
{
([Temps].[Tout].[2001].[Mars].[11])
}
```

```
{
([Temps].[Tout].[2001].[Mars].[12])
}
ON COLUMNS
FROM [vols]
```

Requête MDX 7 : Afficher l'indicateur par défaut pour chaque élément d'un ensemble de 2 semaines, adhérents et avions confondus

Results:

[Temps].[Tout].[2001].[Mars].[11]	[Temps].[Tout].[2001].[Mars].[12]
41	32

7.6 – Syntaxe de base

Dans son approche le MDX est proche du SQL sur son aspect Select et Where même si la similarité n'ira pas plus loin. Ceci dit, c'est un bon moyen pour l'appréhender. Je vous propose le prototype suivant :

```
SELECT [<axis_specification>
      [, <axis_specification>...]]
FROM [<cube_specification>]
[WHERE [<licer_specification>]]
```

Avant de découvrir comment consulter un cube, définissons quelques opérations de navigation dans un cube :

- le dépliage (drilldown) qui consiste à passer à un niveau inférieur dans une dimension ;
- le pliage (drillup ou rollup) qui consiste à passer à un niveau supérieur dans une dimension ;
- le tranchage ou découpage (slice) qui consiste à ne garder qu'une tranche du cube (une tranche étant une coupe du cube selon un membre) ;
- le cubage (dice) qui s'intéresse à plusieurs tranches à la fois ;
- et la rotation qui consiste à choisir les dimensions que l'on veut en colonnes et en lignes.

Avec ces cinq opérations, on peut naviguer dans les données (i.e. pratiquer le data surfing). Il faut y ajouter une sixième opération : le drillthrough (traduit maladroitement par extraction dans Analysis Services, le terme de désagrégation étant préférable) qui permet de retrouver les données qui ont permis de calculer un agrégat.

Le résultat d'une requête MDX est un tableau multidimensionnel dont les cellules ne contiennent qu'une valeur (contrairement aux cubes) et dans lequel on peut naviguer avec les opérations décrites précédemment.

Parmi les dimensions du cube initial, certaines servent d'axes (un axe contient les membres issus d'un dépliage et/ou d'un cubage) pour le résultat (clause SELECT) et d'autres de tranchage (clause WHERE) mais pas les deux à la fois.

Pour MDX, les mesures constituent une dimension, et peuvent donc faire l'objet d'un axe ou d'un tranchage.

La spécification d'un axe doit être SET suffixé du mot clef ON suivi du nom d'axe, par exemple :

```
{
([Avions].[Tous les avions].[PIPER].[Piper Arrow] , [Measures].[Nombre de vols]) ,
([Avions].[Tous les avions].[PIPER].[Piper Arrow].[F GPFA] , [Measures].[Cout total induit])
}
ON COLUMNS
```


La notion d'axe peut faire référence à un numéro d'ordre si vous avez plus de 2 axes de restitution, ou tout simplement aux noms d'axes explicites « columns » tout d'abord et « rows » :

```
SELECT
{
  ( [Avions].[Tous les avions].[PIPER].[Piper Arrow] , [Measures].[Nombre de vols]) ,
  ( [Avions].[Tous les avions].[PIPER].[Piper Arrow].[F GPFA] , [Measures].[Cout
total induit])
}
ON Axis(0),
{
  ([Temps].[Tout].[2001]) ,
  ([Temps].[Tout].[2002])
} ON axis(1)
FROM [Vols]
```

Requête MDX 8

```
SELECT
{
  ( [Avions].[Tous les avions].[PIPER].[Piper Arrow] , [Measures].[Nombre de vols]) ,
  ( [Avions].[Tous les avions].[PIPER].[Piper Arrow].[F GPFA] , [Measures].[Cout
total induit])
}
ON COLUMNS,
{
  ([Temps].[Tout].[2001]) ,
  ([Temps].[Tout].[2002])
} ON ROWS
FROM [Vols]
```

Requête MDX 9

Results:

	[Avions].[Tous les avions].[PIPER].[Piper Arrow]	[Avions].[Tous les avions].[PIPER].[Piper Arrow].[F GPFA]
	[Measures].[Nombre de vols]	[Measures].[Cout total induit]
[Temps].[Tout].[2001]	168	712,720
[Temps].[Tout].[2002]	178	693,840

Figure 10 Résultat des requêtes 8 et 9

Hyperion, Microsoft SSAS (Sql Server Analysis Services) supportent aussi PAGES, SECTIONS, CHAPTERS, à la différence de Mondrian qui n'accepte que les deux premières dimensions (COLUMNS et ROWS). MS SSAS en supporte 128. Hyperion Essbase Analytic Services en supportent 64. Au-delà des 5 premiers axes, il faut utiliser : AXIS (5) , ..., AXIS (127) .

7.7 – La spécification d'un Slicer

Dans la syntaxe nous disposons d'un Where dans lequel vous indiquez un TUPLE qui peut comporter plusieurs dimensions sur lequel on « slice » (découpe). Mondrian accepte aussi des accolades si on souhaite préciser un jeu. Cependant ce dernier doit être réduit par Mondrian en un tuple (*i.e.* un singleton) à l'aide de la fonction Aggregate qui utilise alors la fonction d'agrégation appropriée à chaque mesure. Si ce n'est pas possible, il y a une erreur.

```
SELECT
{
  ( [Avions].[Tous les avions].[PIPER].[Piper Arrow] , [Measures].[Nombre de vols])
,
  ( [Avions].[Tous les avions].[PIPER].[Piper Arrow].[F GPFA] , [Measures].[Cout
total induit])
}
ON COLUMNS,
{

```

```
( [Temps].[Tout].[2001]) ,
( [Temps].[Tout].[2002])
} ON ROWS
FROM [Vols]
WHERE
([Adherents].[Tous les adherents].[Michel DUBOIS].[Eric Martin])
```

Requête MDX 10

Results:

[Adherents].[Tous les adherents].[Michel DUBOIS].[Eric MARTIN]	[Avions].[Tous les avions].[PIPER].[Piper Arrow]	[Avions].[Tous les avions].[PIPER].[Piper Arrow].[F GPFA]
[Measures].[Nombre de vols]	[Measures].[Cout total induit]	
[Temps].[Tout].[2001]	42	177,000
[Temps].[Tout].[2002]	43	171,100

Figure 11 Résultat de la requête 10

Lorsque l'on assigne une dimension ON COLUMNS, ON ROWS et dans un SLICER, on l'utilise. Une dimension ne peut être utilisée pour l'affichage dans une requête MDX qu'une seule fois. Cependant, une dimension peut encore être utilisée de manière interne pour des raisons de calculs ou de filtrage.

Le grand danger relatif aux découpes de cube, est que certaines applications MDX n'affichent pas les tranches alors que c'est une information indispensable pour comprendre les valeurs des mesures.

L'exécution d'une requête MDX entraîne le calcul sur chaque axe des jeux de manière indépendante puis le retour de toutes les combinaisons possibles à l'intersection des axes.

Il nous reste donc à détailler les fonctionnalités immédiatement accessibles par le MDX et voir en quoi elles permettent de résoudre les problématiques métier.

7.8 – Children / Members / CrossJoin

Le MDX manipule des jeux en les positionner sur les axes. Bien entendu, on ne saisit pas tous les membres un par un par un, mais on va disposer de fonction pour retrouver les enfants, les parents, filtrer.

Nous avons vu la notion de Membres comme identifiant sur un axe. Un jeu peut toute fois reprendre une série de membre.

Accès par Membres / Fonction Members :

```
SELECT {[Measures].[Nombre de vols]} ON COLUMNS,
[Temps].[Semaine].Members
ON ROWS
FROM [Vols]
```

Requête MDX 11

Accès par Membres / Fonction Children :

```
SELECT
{([Measures].[Nombre de vols]),
([Measures].[Duree cumulee])} ON COLUMNS,
{([Temps].[Tout].[2001].[Mars].Children)} ON ROWS
FROM Vols
```

Requête MDX 12

Results:

	[Measures].[Nombre de vols]	[Measures].[Duree cumulee]
[Temps].[Tout].[2001].[Mars].[10]	31	108
[Temps].[Tout].[2001].[Mars].[11]	41	146
[Temps].[Tout].[2001].[Mars].[12]	32	105
[Temps].[Tout].[2001].[Mars].[13]	19	63

La fonction M.Children est comme la fonction M.Parent permet de se déplacer dans la hiérarchie respectivement en descente et en montée sauf qu'elle renvoie un jeu alors que M.Parent ne renvoie qu'un membre. De plus elle peut être confondue avec H.Members ou L.Members qui renvoient aussi un jeu de membres. Il existe aussi les instructions suivantes :

- FirstChild : retourne le premier (dans l'ordre de tri de la dimension/du niveau) des enfants d'un membre : [2001].FirstChild => [2001].[Janvier]
- LastChild : retourne le dernier enfant d'un élément
- Siblings : retourne tous les membres qui ont le même élément père que celui sur lequel est portée l'instruction.

```
SELECT Descendants([Temps].[Tout].[2001].[Mars]) ON COLUMNS,  
[Avions].[Tous les avions].[PIPER].[Piper Arrow].Children ON ROWS  
FROM Vols
```

Requête MDX 13

Results:

	[Temps].[Tout].[2001].[Mars]	[Temps].[Tout].[2001].[Mars].[10]	[Temps].[Tout].[2001].[Mars].[11]	[Temps].[Tout].[2001].[Mars].[12]	[Temps].[Tout].[2001].[Mars].[13]
[Avions].[Tous les avions].[PIPER].[Piper Arrow].[F GPFA]	11	1	4	6	(null)

Descendants(M,1) est équivalent à M.children. Au lieu de la distance, on peut utiliser le niveau. Descendants(M,2) donne les petits enfants. Descendants(M) donne tous les descendants. Ancestor(M) renvoie un membre plus élevé dans la hiérarchie.

```
SELECT [Temps].members ON COLUMNS,  
filter([Avions].members,  
not isempty([Measures].[Nombre de vols])) ON ROWS  
FROM Vols  
WHERE [Adherents].[Tous les adherents].[Michel DUBOIS].[Eric Martin]
```

Requête MDX 14

```
SELECT {[Nombre de vols], [Duree cumulee]} ON COLUMNS,  
ORDER({[2000].CHILDREN, [2001].CHILDREN},  
([Nombre de vols], [Duree cumulee]), ASC) ON ROWS  
FROM Vols
```

Requête MDX 15

```
SELECT [Avions].[numero d'immatriculation de l'avion].members ON COLUMNS,  
filter([Temps].[semaine].members,  
[Temps].CurrentMember.Parent.Name="Mars" AND  
[Temps].CurrentMember.Parent.Parent.Name="2001") ON ROWS  
FROM Vols  
WHERE [Adherents].[Tous les adherents].[Michel DUBOIS].[Eric Martin]
```

Requête MDX 16

```
select head( [Temps].levels(2).members,1) on ROWS  
from Vols
```

Requête MDX 17

```
SELECT TOPCOUNT(AVIONS.[NUMERO D'IMMATRICULATION DE L'AVION].MEMBERS, 3,  
(measures.[Duree cumulee], [2000])) ON COLUMNS  
FROM Vols
```

Requête MDX 18

Les fonctions Head(J, cardinal), Tail(J,cardinal), Union(J,J), Except(J,J), Intersect(J,J), Hierarchize(J),Distinct(J) renvoient des jeux de membres.

La fonction `Hierarchize(J)` permet d'organiser le jeu en ordre hiérarchique sans éliminer les doublons. Dans une hiérarchie, l'ordre naturel est utilisé par défaut pour les membres. Les enfants suivent immédiatement les parents. Pour avoir l'inverse, il faut préciser `POST` en second argument.

La fonction `Order(J, expression ORDERBY, ASC|DESC|BASC|BDESC)` permet de trier un ensemble. Mais le problème avec `ASC` et `DESC` est que la hiérarchie est respectée (dans notre exemple, les mois de 2000 seront triés séparément des mois de 2001). Pour ne pas tenir compte de la hiérarchie, il suffit d'utiliser `BASC` et `BDESC`.

La fonction `TopCount(J, cardinal, expression)` permet de faire des classements. Elle est la combinaison d'un `Head` et d'un `Order`. On peut ne garder que les membres extrêmes : filtrer les 3 avions les plus utilisés (durée cumulée en 2000). On a évidemment la fonction `BottomCount`. Il existe aussi `TopSum` et `TopPercent`, etc.

La fonction `Filter(J, Prédicat)` évalue le prédicat en fonction de l'élément du Set et du slicer. L'évaluation à 0 du prédicat pour un membre entraîne l'exclusion de ce membre dans l'ensemble résultat.

L'expression conditionnelle MDX ou prédicat `[Temps].CurrentMember.Name="Mars"` renvoie 0 ou 1. Outre les opérateurs de comparaison (`=`, `>`, `>=`, `<`, `<=`), des expressions conditionnelles MDX peuvent être combinées avec les opérateurs `AND`, `OR`, `XOR` et `NOT`.

La fonction `IsEmpty(expression)` renvoie 1 si l'expression ne renvoie pas de valeur. Elle est utile pour éviter une division par 0. Si l'expression est un tuple, elle comporte 2 niveaux de parenthèses.

Sans utiliser la fonction `Filter`, on peut éliminer simplement les membres qui ne contiennent pas de données : `SELECT NON EMPTY {...} ON COLUMNS FROM Vols`.

Les fonctions `Sum`, `Min`, `Max`, `Avg`, `Filter`, `TopCount`, `BottomCount`, `TopSum`, `BottomSum`, `TopPercent`, `BottomPercent`, `Order`, `Generate` prennent en entrée une expression MDX dont l'évaluation est itérative sur les tuples du jeu en entrée. Pour chaque tuple présent dans ce jeu, il y a sauvegarde du contexte courant, évaluation de l'expression dans le contexte du tuple courant, utilisation du résultat dans une tâche propre à la fonction (pour `Sum`, il s'agit d'un cumul, pour `Order` un tri, etc.) puis restauration du contexte précédemment sauvegardé. La fonction `Generate` permet de construire le jeu résultat par union des jeux générés en fonction du second argument.

Croisement de Set / CrossJoin :

Pour avoir des représentations de type tableaux croisés, nous allons croiser des jeux ensembles (produits cartésiens):

```
SELECT
{
  CROSSJOIN
  (
    { ([Avions].[Tous les avions].[PIPER].[Piper Arrow]) , ([Avions].[Tous les avions].[PIPER].[Piper Archer]) },
    { ([Measures].[Nombre de vols]),([Measures].[Duree cumulee]) }
  )
} ON COLUMNS,
{
  CROSSJOIN (
    [Temps].[Tout].[2001].[Decembre].Children,
    [Temps].[Tout].[2002].[Decembre].Children)
} ON ROWS
FROM [Vols]
```

Requête MDX 19

Results	[Avions].[Tous les avions].[PIPER].[Piper Arrow]				[Avions].[Tous les avions].[PIPER].[Piper Archer]			
	[Mesures].[Nombre de vols]		[Mesures].[Duree cumulee]		[Mesures].[Nombre de vols]		[Mesures].[Duree cumulee]	
[Temps].[Tout].[2001].[Decembre].[49]	[Temps].[Tout].[2002].[Decembre].[49]	4	14	4	17			
[Temps].[Tout].[2001].[Decembre].[49]	[Temps].[Tout].[2002].[Decembre].[50]	1	2	2	5			
[Temps].[Tout].[2001].[Decembre].[49]	[Temps].[Tout].[2002].[Decembre].[51]	3	9	3	15			
[Temps].[Tout].[2001].[Decembre].[49]	[Temps].[Tout].[2002].[Decembre].[52]	2	8	4	15			
[Temps].[Tout].[2001].[Decembre].[50]	[Temps].[Tout].[2002].[Decembre].[49]	4	14	4	17			
[Temps].[Tout].[2001].[Decembre].[50]	[Temps].[Tout].[2002].[Decembre].[50]	1	2	2	5			
[Temps].[Tout].[2001].[Decembre].[50]	[Temps].[Tout].[2002].[Decembre].[51]	3	9	3	15			
[Temps].[Tout].[2001].[Decembre].[50]	[Temps].[Tout].[2002].[Decembre].[52]	2	8	4	15			
[Temps].[Tout].[2001].[Decembre].[51]	[Temps].[Tout].[2002].[Decembre].[49]	4	14	4	17			
[Temps].[Tout].[2001].[Decembre].[51]	[Temps].[Tout].[2002].[Decembre].[50]	1	2	2	5			
[Temps].[Tout].[2001].[Decembre].[51]	[Temps].[Tout].[2002].[Decembre].[51]	3	9	3	15			
[Temps].[Tout].[2001].[Decembre].[51]	[Temps].[Tout].[2002].[Decembre].[52]	2	8	4	15			
[Temps].[Tout].[2001].[Decembre].[52]	[Temps].[Tout].[2002].[Decembre].[49]	4	14	4	17			
[Temps].[Tout].[2001].[Decembre].[52]	[Temps].[Tout].[2002].[Decembre].[50]	1	2	2	5			
[Temps].[Tout].[2001].[Decembre].[52]	[Temps].[Tout].[2002].[Decembre].[51]	3	9	3	15			
[Temps].[Tout].[2001].[Decembre].[52]	[Temps].[Tout].[2002].[Decembre].[52]	2	8	4	15			

On peut aussi utiliser la fonction `NonEmptyCrossJoin` pour combiner les SET qui proviennent de plusieurs dimensions tout en éliminant les tuples qui produisent un résultat vide.

```
SELECT NONEMPTYCROSSJOIN(CROSSJOIN({[MICHEL DUBOIS], [EMMANUEL DUBOIS]} ,
{[2000]:[2002]}), {ROBIN:CESSNA}) ON COLUMNS,
  ADDCALCULATEDMEMBERS(MEASURES.MEMBERS) ON ROWS
FROM VOLS
```

Requête MDX 20

	Adherents									Emmanuel DUBOIS								
	*Michel DUBOIS																	
	Temps									Temps								
	*2000			*2001			*2002			*2000			*2001			*2002		
	Avions			Avions			Avions			Avions			Avions			Avions		
Mesures	*CESSNA	*PIPER	*ROBIN	*CESSNA	*PIPER	*ROBIN	*CESSNA	*PIPER	*ROBIN	*CESSNA	*PIPER	*ROBIN	*CESSNA	*PIPER	*ROBIN	*CESSNA	*PIPER	*ROBIN
Nombre de vols	173	287	248	202	283	334	189	260	324	127	361	204	122	386	260	94	381	264
Duree cumulee	599	997	847	724	1,011	1,151	637	886	1,113	444	1,238	724	418	1,390	865	328	1,293	906
Chiffre d'affaires induit	506,155	1,172,450	865,450	611,780	1,187,450	1,161,350	538,265	1,062,100	1,120,850	306,230	1,481,200	694,000	297,910	1,655,600	822,150	228,160	1,533,050	858,300
Cout total induit	416,904	957,900	709,300	503,904	970,180	952,100	443,352	867,400	918,940	249,924	1,209,720	566,960	243,528	1,352,280	672,300	186,288	1,252,300	701,400
Marge induite	17.63%	18.30%	18.04%	17.63%	18.30%	18.02%	17.63%	18.33%	18.01%	18.39%	18.33%	18.31%	18.25%	18.32%	18.23%	18.35%	18.31%	18.28%

L'opérateur deux points (:) permet d'utiliser l'ordre naturel des membres pour créer un jeu. Avant de l'utiliser, il vaut mieux s'assurer de l'ordre dans lequel sont stockés les membres. L'utilisation de la fonction AddCalculatedMembers me garantit que tous les indicateurs de la dimension [Mesures] définis dans le schéma Mondrian seront présents, que ce soit des mesures stockées ou calculées à la volée (voir les explications du point suivant).

```
SELECT Crossjoin({[Mesures].[Nombre de vols], [Mesures].[Duree cumulee]},
{[Temps].[Tout].[2000].[Janvier], [Temps].[Tout].[2000].[Decembre]}) ON COLUMNS,
  NON EMPTY Generate([Adherents].[Tous les adherents].[Michel DUBOIS].Children,
Crossjoin({[Adherents].[Instructeur suiveur].CurrentMember},
TopCount(Descendants([Avions].CurrentMember, 3.0), 5.0, ([Mesures].[Duree cumulee],
[Temps].[Tout].[2000].[Decembre])))) ON ROWS
from [Vols]
```

Requête MDX 21

	Adherents	Avions	Mesures			
			Nombre de vols		Duree cumulee	
			Temps		Temps	
			*+Janvier	*+Decembre	*+Janvier	*+Decembre
Jacques DUPONT	F GBTK		6	7	27	25
		F BOER			6	24
		F GGOB	1	2	3	10
		F GAQO	3	2	13	10
		F CHCC	5	3	20	9
Eric MARTIN	F BOER		9	6	28	24
		F GFIZ	1	7	3	22
		F GPFA	4	6	12	22
		F GBTK	6	4	25	15
		F CHCC		2		9

Pour obtenir les 5 meilleurs avions en durée cumulée au mois de décembre 2000 pour chaque adhérent suivi par Michel Dubois, il faut utiliser la fonction Generate qui va itérer sur les enfants de [Michel Dubois] et pour chacun d'entre eux, il va combiner (CrossJoin) cet enfant avec les cinq meilleurs avions.

7.9 - Membres Calculés / With

Un membre calculé est créé par une expression MDX au contraire d'un membre stocké qui voit ses valeurs importées lors de la création du cube ou calculée lors de l'agrégation. Il diffère aussi d'un membre dérivé, qui est une expression SQL. Un membre calculé voit ses valeurs définies à la volée et n'entrent pas dans le processus d'agrégation. Il dépend généralement d'autres membres. S'ils n'accroissent pas la taille du cube, les membres calculés peuvent ralentir le traitement d'une requête. Ils peuvent se répartir dans toutes dimensions. Les membres calculés ne sont pas retournés par défaut par la fonction Members, il faut utiliser la fonction AddCalculatedMembers ou la fonction AllMembers.

Un membre dérivé (notion spécifique à MS SSAS) appartient forcément à la dimension [Mesures]. Son expression SQL est évaluée avant le processus d'agrégation. Cette évaluation peut être polluée par la présence de valeurs manquantes. De plus il est conseillé d'utiliser la fonction Min ou Max comme fonction d'agrégation pour éviter des résultats erronés.

Nous avons créé dans le cube un membre calculé [Mesures].[Marge induite] qui est donc accessible directement dans les requêtes MDX :

```
<!-- Cube -->
<Cube name="Vols">
  <DimensionUsage name="Temps" source="Temps" foreignKey="IDTPS"/>
  <DimensionUsage name="Avions" source="Avions" foreignKey="NUMAVION"/>
  <DimensionUsage name="Adherents" source="Adherents" foreignKey="NUMADHERENT"/>
  <Measure name="Nombre de vols" column="NBVOLS" datatype="Integer" aggregator="sum" formatString="#.###"/>
  <Measure name="Duree cumulee" column="DUREECUMULEE" datatype="Integer" aggregator="sum" formatString="#.###"/>
  <Measure name="Chiffre d'affaires induit" column="CAINDUIT" datatype="Numeric" aggregator="sum" formatString="###,###,###.##"/>
  <Measure name="Cout total induit" column="COUTTOTAL" datatype="Numeric" aggregator="sum" formatString="###,###,###.##"/>
  <CalculatedMember name="Marge induite" dimension="Measures" datatype="Numeric" formula="([Measures].[Chiffre d'affaires induit]-[Measures].[Cout total induit]) / [Measures].[Chiffre d'affaires induit]">
    <CalculatedMemberProperty name="FORMAT_STRING" value="#.00%"/>
  </CalculatedMember>
</Cube>
```

Figure 12 : Extrait du fichier XML créant le cube [Vols] dans Mondrian

Je vous propose ici de rajouter un nouveau membre calculé [CA par heure de vol], mais cette fois ci en passant par le MDX à l'aide du mot clef « With » :

```
WITH
  MEMBER [Measures].[CA par heure de vol]
  AS
  (
    [Measures].[Chiffre d'affaires induit] / [Measures].[Duree cumulee]
  )
SELECT
  {[Measures].[Nombre de vols],
  [Measures].[Duree cumulee],
  [Measures].[Chiffre d'affaires induit],
  [Measures].[CA par heure de vol]}
ON COLUMNS,
  {[([Temps].[Tout],
  [Avions].[Tous les avions],
  [Adherents].[Tous les adherents])}]
ON ROWS
FROM Vols
```

Requête MDX 22

A noter que l'on peut aussi créer avec l'instruction WITH SET, un jeu nommé de Membres. Pour être accessible de manière globale, il faut le définir dans le fichier XML de schéma Mondrian. On peut créer localement à une requête MDX des ensembles nommés. Dans la définition d'un jeu nommé, on peut utiliser les fonctions ensemblistes Union, Except et Intersect.

```
WITH SET [Le meilleur adherent] AS
  'TOPCOUNT(Adherents.[Adherent].MEMBERS, 1, (measures.[Duree cumulee]))'
  SET [Le pire adherent] AS
  'BOTTOMCOUNT(Adherents.[Adherent].MEMBERS, 1, (measures.[Duree cumulee]))'
  SET [Les adherents moyens] AS
  'EXCEPT(EXCEPT(Adherents.[Adherent].MEMBERS,[Le Meilleur Adherent]),[Le Pire Adherent])'
SELECT {measures.[Duree cumulee]} ON COLUMNS,
{[Le meilleur adherent],[Le pire adherent],[Les adherents moyens]} ON ROWS
FROM Vols
WHERE [Temps].[Annee].[2000]
```

Requête MDX 23

7.10 – Manipuler le temps

Calculer un « Year to Date » consiste à cumuler une mesure sur du début de l'année jusqu'au mois en cours. L'idée est de renvoyer un set qui liste pour chaque mois, la liste des mois précédent et accumuler la mesure. Sur ce point on utilisera la fonction YTD() qui attend comme argument un membre quelconque de la hiérarchie et se charge de retrouver la liste des membres (donc un jeu). J'utilise ici en plus une fonction SetToStr qui nous permet de visualiser le retour sous forme de

chaîne et la fonction Properties qui me permet de récupérer la propriété numérique « Numéro » du niveau Mois (entre 1 et 12) définie dans le fichier XML de Mondrian afin de limiter l’affichage au 3 premier mois.

```
WITH MEMBER [Measures].[YTD / Q] AS 'SetToStr(Ytd([Temps].CurrentMember))'
SELECT {[Measures].[Duree cumulee], [Measures].[YTD / Q]} ON COLUMNS,
  Filter(
    Descendants([Temps].[Tout].[2001], [Temps].[Mois]),
    [Temps].CurrentMember.Properties("Numero") < 4
  ) ON ROWS
FROM [Vols]
```

Requête MDX 24

	Mesures	
Temps	Duree cumulee	YTD / Q
+Janvier	534	{[Temps].[Tout].[2001].[Janvier]}
+Fevrier	435	{[Temps].[Tout].[2001].[Janvier], [Temps].[Tout].[2001].[Fevrier]}
+Mars	422	{[Temps].[Tout].[2001].[Janvier], [Temps].[Tout].[2001].[Fevrier], [Temps].[Tout].[2001].[Mars]}

On constate que pour chaque ligne du tableau nous allons rechercher pour le membre en cours de la hiérarchie temps citée, la liste de tous les membres précédents depuis le début de l'année lui-même inclus.

Cela veut dire aussi que si nous allions chercher les descendants cette fois au niveau de la semaine par exemple, la formule de la mesure ne changerait pas et s'adapterait au nouveau contexte (nous le présenterons un peu plus loin) typiquement pour la recherche de point de rentabilité effectif dans l'année.

Pour en revenir à notre calcul, il nous reste à adjoindre la mesure pour obtenir les cumuls avec la fonction Sum qui travaille sur un jeu :

```
With Member [Measures].[YTD / Q]
as
  Sum(
    YTD( [Temps].CurrentMember ),
    [Measures].[Nombre de vols]
  )
Select
  { ([Measures].[Nombre de vols]) , [Measures].[YTD / Q] } on columns,
  Descendants([Temps].[Tout].[2001], [Temps].[Mois]) on rows
from [Vols]
```

Requête MDX 25

Results:

	[Measures].[Nombre de vols]	[Measures].[YTD / Q]
[Temps].[Tout].[2001].[Janvier]	155	155
[Temps].[Tout].[2001].[Fevrier]	125	280
[Temps].[Tout].[2001].[Mars]	123	403
[Temps].[Tout].[2001].[Avril]	126	529
[Temps].[Tout].[2001].[Mai]	150	679
[Temps].[Tout].[2001].[Juin]	117	796
[Temps].[Tout].[2001].[Juillet]	125	921
[Temps].[Tout].[2001].[Aout]	136	1,057
[Temps].[Tout].[2001].[Septembre]	129	1,186
[Temps].[Tout].[2001].[Octobre]	144	1,330
[Temps].[Tout].[2001].[Novembre]	126	1,456
[Temps].[Tout].[2001].[Decembre]	131	1,587

Toujours dans la série des questions récurrentes, on recherche le poids de chaque mois dans le résultat de l'année : un tel problème se nomme « Pourcentage par rapport au parent ».

```
With Member [Measures].[% Real]
as
(
    (
        ([Measures].[Nombre de vols],[Temps].currentMember )
        /
        ( ANCESTOR( [Temps].currentMember, [Temps].[Annee] ),
          [Measures].[Nombre de vols]
        )
    )
    , format_string="#.00%"
)

Select
    { ([Measures].[Nombre de vols]) , [Measures].[% Real] } on columns,
    Descendants([Temps].[Annee].[2001],[Temps].[Mois]) on rows
from [Vols]
```

Requête MDX 26

Results:

	[Measures].[Nombre de vols]	[Measures].[% Real]
[Temps].[Tout].[2001].[Janvier]	155	9.77%
[Temps].[Tout].[2001].[Fevrier]	125	7.88%
[Temps].[Tout].[2001].[Mars]	123	7.75%
[Temps].[Tout].[2001].[Avril]	126	7.94%
[Temps].[Tout].[2001].[Mai]	150	9.45%
[Temps].[Tout].[2001].[Juin]	117	7.37%
[Temps].[Tout].[2001].[Juillet]	125	7.88%
[Temps].[Tout].[2001].[Aout]	136	8.57%
[Temps].[Tout].[2001].[Septembre]	129	8.13%
[Temps].[Tout].[2001].[Octobre]	144	9.07%
[Temps].[Tout].[2001].[Novembre]	126	7.94%
[Temps].[Tout].[2001].[Decembre]	131	8.25%

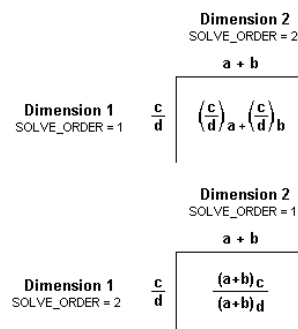
On aurait pu se satisfaire de rentrer en dur l'année 2001 dans la formule, mais en utilisant la fonction Ancestor, on retrouvera dynamiquement la valeur de l'ancêtre du membre en cours au niveau de l'année. On peut regarder du côté de la fonction Parent qui aurait pu nous permettre de construire le même résultat mais avec un niveau de paramétrage moindre. D'autres options sont disponibles dans la clause MEMBER (cf. l'aide en ligne). Comme par exemple, une description du format numérique à employer : FORMAT_STRING = '#.## euros'

7.11 – Ordre de résolution des membres calculés

La priorité de la résolution de la formule d'un membre calculé est définie par solve_order. Un membre avec priorité de 0 sera calculé avant un membre à priorité 1. Préciser la priorité permet d'éviter des écarts entre le résultat attendu et celui obtenu. De tels écarts peuvent apparaître lorsque plusieurs membres calculés sont dépendants ou qu'ils appartiennent à des dimensions différentes qui s'intersectent au niveau de cellules. C'est généralement le cas lorsque l'on combine pourcentage et différence.

Le diagramme ci-contre montre comment le changement de l'ordre de résolution peut changer de manière significative le résultat dans une cellule qui est le croisement des deux membres calculés.

De plus, voici un exemple :



```
WITH
    MEMBER [Measures].[CA par heure de vol]
```

```

AS
(
[Measures].[Chiffre d'affaires induit] / [Measures].[Duree cumulee]
), format_string = '##.##'
Member [Temps].[Evolution]
as
(
([Temps].[Annee].[2002] - [Temps].[Annee].&[2001])
/
[Temps].[Annee].&[2001]
) , format_string="#.00%"

SELECT
CrossJoin(
{([temps].[Annee].&[2001]),([temps].[Annee].[2002]),([Temps].[Evolution])},
{([Measures].[Chiffre d'affaires induit]), ([Measures].[Duree
cumulee]),([Measures].[CA par heure de vol])}
) ON COLUMNS,
[Adherents].[Adherent].Members
ON ROWS
FROM Vols

```

Requête MDX 27

	Temps								
	+2001			+2002			Evolution		
	Mesures			Mesures			Mesures		
Adherents	Chiffre d'affaires induit	Duree cumulee	CA par heure de vol	Chiffre d'affaires induit	Duree cumulee	CA par heure de vol	Chiffre d'affaires induit	Duree cumulee	CA par heure de vol
Michel DURANT	1,396,560	1,337	1044.55	1,394,550	1,323	1054.08	-0.14%	-1.05%	.14
Christian DELMAS	1,379,100	1,336	1032.26	1,224,960	1,204	1017.41	-11.18%	-9.88%	1.13
Jacques DUPONT	1,532,120	1,484	1032.43	1,350,540	1,295	1042.89	-11.85%	-12.74%	.93
Eric MARTIN	1,428,460	1,402	1018.87	1,370,675	1,341	1022.13	-4.05%	-4.35%	.93

Une autre méthode de référence aux membres consiste à utiliser la clé de membre. Une dimension utilise la clé du membre pour identifier de manière spécifique un membre particulier. Dans la syntaxe MDX, le caractère et commercial (&) différencie une clé de membre d'un nom de membre. Par exemple, les références suivantes utilisent la clé de membre du membre correspondant à la semaine 52 : [Temps].[Decembre].&[52]. L'utilisation d'une clé de membre garantit une identification correcte des membres dans les dimensions variables et les dimensions qui n'imposent pas des noms de membre uniques.

Si on regarde dans le détail, on se rend compte que l'on calcul non pas le pourcentage d'évolution des [CA par heure de vol], mais le ratio des évolutions entre le « pourcentage d'évolution du chiffre d'affaires induit » (-0,14%) et « le pourcentage d'évolution de la durée cumulée » (-1,05%), ce qui ne veut absolument rien dire !

Pour ne pas être contraint par ce type d'erreurs, il est possible spécifier un ordre de résolution des membres calculés avec solve_order :

```

WITH
MEMBER [Measures].[CA par heure de vol]
AS
(
[Measures].[Chiffre d'affaires induit] / [Measures].[Duree cumulee]
), format_string = '##.##', solve_order=0
Member [Temps].[Evolution]
as
(
([Temps].[Annee].[2002] - [Temps].[Annee].&[2001])
/
[Temps].[Annee].&[2001]
) , solve_order=1, format_string="#.00%"

SELECT
CrossJoin(
{([temps].[Annee].&[2001]),([temps].[Annee].[2002]),([Temps].[Evolution])},

```

```
{([Measures].[Chiffre d'affaires induit]), ([Measures].[Duree cumulee]), ([Measures].[CA par heure de vol])}
) ON COLUMNS,
    [Adherents].[Adherent].Members
ON ROWS
FROM Vols
```

Requête MDX 28

	Temps								
	+2001			+2002			Evolution		
	Mesures			Mesures			Mesures		
Adherents	Chiffre d'affaires induit	Duree cumulee	CA par heure de vol	Chiffre d'affaires induit	Duree cumulee	CA par heure de vol	Chiffre d'affaires induit	Duree cumulee	CA par heure de vol
Michel DURANT	1,396,560	1,337	1044.55	1,394,550	1,323	1054.08	- .14%	- 1.05%	.91%
Christian DELMAS	1,379,100	1,336	1032.26	1,224,960	1,204	1017.41	-11.18%	-9.88%	-1.44%
Jacques DUPONT	1,532,120	1,484	1032.43	1,350,540	1,295	1042.89	-11.85%	-12.74%	1.01%
Eric MARTIN	1,428,460	1,402	1018.87	1,370,675	1,341	1022.13	-4.05%	-4.35%	.32%

On retrouve bien : $(1054,08-1044,55) / 1044,55=0,00912=0,91\%$

A noter pour finir que l'on peut affecter un membre calculé sur une dimension autre que [Measures].

```
WITH member [Temps].[Tout].[2000].[1er trimestre] as
'([Temps].[Tout].[2000].[Janvier] + [Temps].[Tout].[2000].[Fevrier]) +
[Temps].[Tout].[2000].[Mars])'
member [Temps].[Tout].[2000].[2ieme trimestre] as '([Temps].[Tout].[2000].[Avril]
+ [Temps].[Tout].[2000].[Mai]) + [Temps].[Tout].[2000].[Juin])'
member [Temps].[Tout].[2000].[3ieme trimestre] as
'([Temps].[Tout].[2000].[Juillet] + [Temps].[Tout].[2000].[Aout]) +
[Temps].[Tout].[2000].[Septembre])'
member [Temps].[Tout].[2000].[4ieme trimestre] as
'([Temps].[Tout].[2000].[Octobre] + [Temps].[Tout].[2000].[Novembre]) +
[Temps].[Tout].[2000].[Mars])'
SELECT Except(AddCalculatedMembers([Temps].[Tout].[2000].Children),
[Temps].[Tout].[2000].Children) ON COLUMNS,
    AddCalculatedMembers([Measures].Members) ON ROWS
FROM [Vols]
```

Requête MDX 29

Mesures	Temps			
	1er trimestre	2ieme trimestre	3ieme trimestre	4ieme trimestre
Nombre de vols	354	348	346	340
Duree cumulee	1,264	1,193	1,198	1,150
Chiffre d'affaires induit	1,313,745	1,256,180	1,234,610	1,198,965
Cout total induit	1,074,836	1,027,264	1,009,808	980,712
Marge induite	18.19%	18.22%	18.21%	18.20%

7.12 – Rendre les membres calculés adaptables au contexte

Parmi les fonctionnalités très intéressantes dans le MDX, c'est le fait de pouvoir conditionner les calculs MDX avec le IIF (Immediate IF) qui retourne l'une des deux valeurs déterminées par un test logique.

Dans l'exemple qui suit, nous construisons plusieurs Membres que nous composons ensemble, plus 1 :

- Q-1 = Quantités à la même période l'année précédente
- EVOL = Evolution par rapport à A-1
- % = % d'évolution par rapport à A-1
- PERF = cette mesure renvoie une chaîne de caractère qui nous permet d'appliquer des règles de gestion liées à des conditions

```

With
Member [Measures].[Q-1]
as
(
    PARALLELPERIOD([Temps].[Annee],1,[Temps].CurrentMember) ,
    [Measures].[Nombre de vols]
)

Member [Measures].[EVOL]
as
(
    ([Temps].CurrentMember,[Measures].[Nombre de vols] )
    -
    (
        PARALLELPERIOD([Temps].[Annee],1,[Temps].CurrentMember) ,
        [Measures].[Nombre de vols]
    )
) , solve_order = 1

Member [Measures].[%]
as
(
    [Measures].[EVOL] / (
        PARALLELPERIOD([Temps].[Annee],1,[Temps].CurrentMember) ,
        [Measures].[Nombre de vols]
    )
),FORMAT_STRING = "#.00%", solve_order = 0

Member [Measures].[PERF] as
iif( [Measures].[%] < 0 , "-",
    iif( [Measures].[%] > 0 and [Measures].[%] < 0.15 , "+",
        iif([Measures].[%] >= 0.15 and [Measures].[%] < 0.3 , "++", "Une meilleur
performance")
    )
)

Select
    { ([Measures].[Nombre de vols]), [Measures].[Q-1],[Measures].[EVOL]],
    [Measures].[%] ,[Measures].[PERF] } on columns,

    Descendants([Temps].[Annee].[2001],[Temps].[Mois])
on rows
from [Vols]
    
```

Requête MDX 30

Results:

	[Measures].[Nombre de vols]	[Measures].[Q-1]	[Measures].[EVOL]	[Measures].[%]	[Measures].[PERF]
[Temps].[Tout].[2001].[Janvier]	155	117	38	32.48%	Une meilleur performance
[Temps].[Tout].[2001].[Fevrier]	125	140	-15	-10.71%	-
[Temps].[Tout].[2001].[Mars]	123	97	26	26.80%	++
[Temps].[Tout].[2001].[Avril]	126	108	18	16.67%	++
[Temps].[Tout].[2001].[Mai]	150	120	30	25.00%	++
[Temps].[Tout].[2001].[Juin]	117	120	-3	-2.50%	-
[Temps].[Tout].[2001].[Juillet]	125	113	12	10.62%	+
[Temps].[Tout].[2001].[Aout]	136	140	-4	-2.86%	-
[Temps].[Tout].[2001].[Septembre]	129	93	36	38.71%	Une meilleur performance
[Temps].[Tout].[2001].[Octobre]	144	114	30	26.32%	++
[Temps].[Tout].[2001].[Novembre]	126	129	-3	-2.33%	-
[Temps].[Tout].[2001].[Decembre]	131	109	22	20.18%	++

La fonction IIF renvoie des valeurs différentes suivant l'évaluation d'une expression conditionnelle qui constitue son premier paramètre. Si elle est évaluée à vraie, c'est la valeur contenue dans le deuxième paramètre qui est renvoyé. Sinon, c'est celle contenue dans le dernier paramètre qui est renvoyé.

La requête MDX précédente permet d'illustrer que le MDX contient un grand nombre de fonctions d'analyses avancées. Ici la fonction ParallelPeriod est utilisée pour récupérer la période équivalente à la période de la requête (2001) l'année auparavant (2000). L'intérêt de cette fonction est quelle est générique quel que soit la période choisie en filtre !

Le côté « magique » c'est que si on prend soin de rendre souple ses membres calculés, ils peuvent s'adapter à tous les contextes. Ici, nous faisons la même requête mais au niveau des semaines :

```
With
Member [Measures].[Q-1]
as
(
    PARALLELPERIOD([Temps].[Annee],1,[Temps].CurrentMember) ,
    [Measures].[Nombre de vols]
)

Member [Measures].[EVOL]
as
(
    ([Temps].CurrentMember,[Measures].[Nombre de vols] )
    -
    (
        PARALLELPERIOD([Temps].[Annee],1,[Temps].CurrentMember) ,
        [Measures].[Nombre de vols]
    )
) , solve_order = 1

Member [Measures].[P]
as
(
    [Measures].[EVOL] / (
        PARALLELPERIOD([Temps].[Annee],1,[Temps].CurrentMember) ,
        [Measures].[Nombre de vols]
    )
),FORMAT_STRING = "#.00%", solve_order = 0

Member [Measures].[PERF] as
iif( [Measures].[P] < 0 , "-",
    iif( [Measures].[P] > 0 and [Measures].[P] < 0.15 , "+",
        iif([Measures].[P] >= 0.15 and [Measures].[P] < 0.3 , "++", "Une meilleur
performance")
    )
)

Select
{ ([Measures].[Nombre de vols]), [Measures].[Q-1],[Measures].[EVOL]),
[Measures].[P] ,[Measures].[PERF] } on columns,
Descendants([Temps].[Annee].[2001],[Temps].[Semaine])
on rows
from [Vols]
```

Requête MDX 31

Mesures					
Temps	Nombre de vols	Q-1	EVOL	%	PERF
1	28	4	24	600.00%	Une meilleur performance
2	39	33	6	18.18%	++
3	37	25	12	48.00%	Une meilleur performance
4	31	25	6	24.00%	++
5	20	30	-10	-33.33%	-
6	27	23	4	17.39%	++
7	38	34	4	11.76%	+
8	30	28	2	7.14%	+
9	30	32	-2	-6.25%	-
10	31	23	8	34.78%	Une meilleur performance
11	41	28	13	46.43%	Une meilleur performance
12	32	29	3	10.34%	+
13	19	17	2	11.76%	+
14	33	31	2	6.45%	+
15	29	32	-3	-9.37%	-

...

7.13 – Classements, régression, moyennes mobiles, etc.

La fonction `LastPeriod(Num [, M])` retourne un jeu composé de membres précédents qui inclus le membre courant. Elle est utile pour les moyennes mobiles par sa combinaison avec la fonction `Avg`. A noter que cette fonction ne semble pas fonctionner avec Mondrian. On peut cependant calculer une moyenne mobile avec `Yld`, `Tail`, `Count`, `CurrentMember`, `NextMember`.

La fonction `Rank(T, J)` permet de retourner la position ordinale d'un tuple dans un jeu. Elle peut aussi servir pour convertir une période de temps en nombres séquentiels pour la régression.

La fonction `LinRegPoint(Num, J, Num, Num)` permet de calculer un point dans une ligne de tendance.

La fonction `LinRegIntercept` retourne la valeur de la constante b de la tendance linéaire $y=ax+b$.

La fonction `LinRegR2` retourne le coefficient de détermination R2 de la régression linéaire. D'autres fonctions sont aussi disponibles : `LinRegSlope` et `LinRegVariance`.

Parmi d'autres fonctions statistiques, citons `FirstQ`, `ThirdQ`, `Median`, `Stddev`, `StdDevP`, `Var`, `VarP`, `Covariance`, `CovarianceN`.

```
WITH
Member [Measures].X AS
'Rank([Temps],[Mois].Members)'
Member [Measures].Tendance AS
'LinRegPoint(Measures.X, [Mois].Members,[Nombre de vols], Measures.X)'
MEMBER [Measures].[Moyenne mobile sur 3 mois centree sur le mois courant] AS 'IIF
(COUNT(Tail(YTD([Temps].CurrentMember.NextMember),3))=3,Avg(Tail(YTD([Temps].CurrentM
ember.NextMember),3), [Nombre de vols]),0)'
MEMBER [Measures].[Composantes de la Moyenne mobile sur 3 mois] AS
'SetToStr(Tail(YTD([Temps].CurrentMember.NextMember),3))'

Select
{ ([Measures].[Nombre de vols]), [Measures].[Moyenne mobile sur 3 mois centree
sur le mois courant],
[Measures].X, [Measures].Tendance } on columns,
Descendants([Temps].[Annee].[2002],[Temps].[Mois]) on rows
from [Vols]
```

Requête MDX 32

Results:

	[Mesures].[Nombre de vols]	[Mesures].[Moyenne mobile sur 3 mois centree sur le mois courant]	[Mesures].[X]	[Mesures].[Tendance]
[Temps].[Tout].[2002].[Janvier]	143		25	127
[Temps].[Tout].[2002].[Fevrier]	110	124	26	127
[Temps].[Tout].[2002].[Mars]	118	132	27	127
[Temps].[Tout].[2002].[Avril]	167	136	28	127
[Temps].[Tout].[2002].[Mai]	123	137	29	128
[Temps].[Tout].[2002].[Juin]	122	128	30	128
[Temps].[Tout].[2002].[Juillet]	140	123	31	128
[Temps].[Tout].[2002].[Aout]	107	122	32	129
[Temps].[Tout].[2002].[Septembre]	118	118	33	129
[Temps].[Tout].[2002].[Octobre]	129	121	34	129
[Temps].[Tout].[2002].[Novembre]	116	121	35	129
[Temps].[Tout].[2002].[Decembre]	119		36	130

Pour une liste des fonctions MDX implantées dans Mondrian : <http://mondrian.pentaho.org/documentation/mdx.php>

Pour la spécification MDX de Microsoft en français : [http://msdn2.microsoft.com/fr-fr/library/ms145506\(fr-fr,SQL.90\).aspx](http://msdn2.microsoft.com/fr-fr/library/ms145506(fr-fr,SQL.90).aspx)

8. Schéma / configuration du serveur Mondrian

```
<Schema name="Aeroclub_OLAP">
  <!-- Shared dimensions -->
  <Dimension name="Temps">
    <Hierarchy hasAll="true" allMemberName="Tout" primaryKey="IDTPS">
      <Table name="DIM_TEMPS">
        <Level name="Annee" column="ANNEE" levelType="TimeYears" type="Numeric" uniqueMembers="true"/>
        <Level name="Mois" column="MOIS_EN_LETTE" ordinalColumn="MOIS" nameColumn="MOIS_EN_LETTE" levelType="TimeMonths" uniqueMembers="false" type="String">
          <Property name="Numero" column="MOIS" type="Numeric"/>
          <Property name="Libelle" column="LIBELLE_MOIS" type="String"/>
        <Level>
          <Level name="Semaine" column="SEMAINE" type="Numeric" uniqueMembers="false" levelType="TimeWeeks">
            <Property name="Identifiant" column="IDTPS" type="Numeric"/>
            <Property name="Libelle" column="LIBELLE_SEMAINE" type="String"/>
          <Level>
            <Hierarchy>
              <Dimension>
                <Dimension name="Avions">
                  <Hierarchy hasAll="true" allMemberName="Tous les avions" primaryKey="NUMAVION">
                    <Table name="DIM_ENGIN">
                      <Level name="Famille de l'avion" column="FAMILLEAVION" uniqueMembers="true"/>
                      <Level name="Type de l'avion" column="TYPEAVION" uniqueMembers="true"/>
                      <Level name="Numero d'immatriculation de l'avion" column="NUMAVION" uniqueMembers="true"/>
                    <Hierarchy>
                      <Dimension>
                        <Dimension name="Adherents">
                          <Hierarchy hasAll="true" allMemberName="Tous les adherents" primaryKey="NUMADHERENT">
                            <Table name="DIM_ADH">
                              <Level name="Instructeur suiveur" column="NOMCOMPLETINSTRUCTEUR" type="String" ordinalColumn="NUMINSTRUCTEUR" nameColumn="NOMCOMPLETINSTRUCTEUR" uniqueMembers="true">
                                <Property name="Numero de l'instructeur" column="NUMINSTRUCTEUR" type="Numeric"/>
                              <Level>
                                <Level name="Adherent" column="NOMCOMPLETADHERENT" type="String" ordinalColumn="NUMADHERENT" nameColumn="NOMCOMPLETADHERENT" uniqueMembers="true">
                                  <Property name="Numero de l'adherent" column="NUMADHERENT" type="Numeric"/>
                                <Level>
                                  <Hierarchy>
                                    <Dimension>
                                      <!-- Cube -->
                                      <Cube name="Vols">
                                        <Table name="FAITS_VOLS">
                                          <AggName name="AGG_TOUT_VOLS">
                                            <AggFactCount column="FACT_COUNT"/>
                                            <AggIgnoreColumn column="ID"/>
                                            <AggIgnoreColumn column="MARGE"/>
                                            <AggMeasure name="[Mesures].[Nombre de vols]" column="NBVOLS"/>
                                            <AggMeasure name="[Mesures].[Duree cumulee]" column="DUREECUMULEE"/>
                                            <AggMeasure name="[Mesures].[Chiffre d'affaires induit]" column="CAINDUIT"/>
                                            <AggMeasure name="[Mesures].[Cout total induit]" column="COUTTOTAL"/>
                                          <AggName>
                                            <AggName name="AGG_ANNEE_VOLS"><AggName>
                                            <AggName name="AGG_SUIVEUR_VOLS"><AggName>
                                            <AggName name="AGG_TYPE_VOLS"><AggName>
                                          <Table>
                                            <DimensionUsage name="Temps" source="Temps" foreignKey="IDTPS"/>
                                            <DimensionUsage name="Avions" source="Avions" foreignKey="NUMAVION"/>
                                            <DimensionUsage name="Adherents" source="Adherents" foreignKey="NUMADHERENT"/>
                                            <Measure name="Nombre de vols" column="NBVOLS" datatype="Integer" aggregator="sum" formatString="#,###"/>
                                            <Measure name="Duree cumulee" column="DUREECUMULEE" datatype="Integer" aggregator="sum" formatString="#,###"/>
                                            <Measure name="Chiffre d'affaires induit" column="CAINDUIT" datatype="Numeric" aggregator="sum" formatString="###,###,###,###"/>
                                            <Measure name="Cout total induit" column="COUTTOTAL" datatype="Numeric" aggregator="sum" formatString="###,###,###,###"/>
                                            <CalculatedMember name="Marge induite" dimension="Measures" datatype="Numeric" formula="([Mesures].[Chiffre d'affaires induit]-[Mesures].[Cout total induit]) / [Mesures].[Chiffre d'affaires induit]">
                                            <CalculatedMemberProperty name="FORMAT_STRING" value="#.00%"/>
                                          <CalculatedMember>
                                        <Cube>
                                      <!-- Roles -->
                                      <Role name="Administrateur">
                                        <SchemaGrant access="all"/>
                                      <Role>
                                    </Schema>
                                  </Dimension>
                                </Level>
                              </Level>
                            </Hierarchy>
                          </Dimension>
                        </Level>
                      </Level>
                    </Hierarchy>
                  </Dimension>
                </Level>
              </Table>
            </Hierarchy>
          </Level>
        </Table>
      </Hierarchy>
    </Dimension>
  </Schema>
```

Un schéma (Cf. [ce document](#)) définit une base de données multidimensionnelle. Il contient le modèle logique des cubes, des hiérarchies, des membres et la mise en correspondance (mapping) qui transforme ce modèle logique en un modèle physique. Le modèle logique consiste à définir les structures utilisées dans une requête MDX : cubes, dimensions, hiérarchies, niveaux et membres. Il donne également la source des données représentées dans le modèle logique. C'est le schéma en étoile ou en flocon qui est traduit par un ensemble de tables relationnelles. Les schémas ROLAP sous Mondrian sont représentés à l'aide

d'un document XML. Un moyen de créer un tel schéma est d'éditer un document XML et de respecter la syntaxe proposées dans Mondrian. Les plus importants des composants du schéma sont les cubes, les indicateurs et les dimensions.

Rappelons qu'un cube est une collection de dimensions et d'indicateurs dans un domaine particulier. Un indicateur est une quantité qui vous intéresse, que vous souhaitez observer (par exemple, le montant des ventes, nombre de produits inventoriés, etc.). Une dimension est un attribut, ou un ensemble d'attributs, à travers lesquels sont observées les indicateurs. Par exemple, vous pouvez être intéressés à observer la vente des produits selon leurs couleurs, le sexe du client et le magasin où sont vendus ces produits. La couleur du produit, le sexe du client et le magasin de vente sont des dimensions.

Un cube (<Cube>) est une collection de mesures et de dimensions, identifiée par un nom. Les dimensions et les mesures ont la table de faits en commun (dans l'exemple, la table de faits est "FAITS_VOLS"). La table de faits contient les colonnes à partir desquelles les mesures sont calculées et des références vers les tables contenant les dimensions. La table de faits est définie en utilisant <Table>. Si la table de faits n'est pas dans le schéma par défaut, vous pouvez fournir explicitement son schéma en utilisant l'attribut "schema". Par exemple, <Table schema="dmart" name="sales_fact_1997"/>. Vous pouvez aussi utiliser <View> et <Join> pour construire des commandes SQL encore plus complexes.

Chaque indicateur (<Measure>) a un nom, une colonne de correspondance dans la table de faits, un opérateur d'agrégation. L'opérateur d'agrégation est souvent "sum", mais d'autres opérateurs comme "count", "min", "max", "avg" et "distinct-count" peuvent être utilisés. L'opérateur "distinct-count" a des limitations si le cube contient une hiérarchie parent-fils.

L'attribut optionnel "datatype" spécifie comment les valeurs des cellules seront représentées dans le cache de Mondrian et comment elles seront retournées via XMLA. L'attribut "datatype" peut avoir pour valeur "String", "Integer", "Numeric", "Boolean", "Date", "Time", et "Timestamp". La valeur par défaut est "Numeric" à l'exception des opérateurs "count" et "distinct-count" qui ont une valeur par défaut "Integer".

Un autre attribut optionnel est "formatString". Il donne le format à utiliser lors de l'affichage des données. Par exemple, la mesure "Unit Sales" est affichée sans décimales, alors que "Store Sales" est affichées avec deux décimales. L'emplacement des caractères ',' et '.' dépend des notations adoptées dans chaque région du monde pour afficher les décimales. Une mesure peut utiliser un attribut "caption" (étiquette en français) qui sera retourné à la place du nom lors d'un appel à l'aide de la méthode Member.getCaption() dans une requête MDX. La définition d'une étiquette a un sens lorsque les caractères comme Σ ou Π sont utilisés : <Measure name="Sum X" column="sum_x" aggregator="sum" caption=" Σ X"/>

Un membre est un point dans une dimension déterminé par les valeurs d'attributs de cette dimension. La hiérarchie "Gender" a deux membres 'M' et 'F'. Une hiérarchie est un ensemble de membres organisés selon une structure appropriée pour l'analyse. Par exemple, les villes peuvent être regroupées par région et les régions par pays. Les mesures sont agrégées pour chaque niveau de la hiérarchie. Les ventes d'un pays sont calculées à partir des ventes de ses régions. Un niveau est une collection de membres qui ont la même distance de la racine de la hiérarchie. Une dimension est une collection de hiérarchies selon laquelle les faits sont observés. Ci-dessous un exemple d'une représentation XML de la dimension "Gender" :

```
<Dimension name="Gender" foreignKey="customer_id">
  <Hierarchy hasAll="true" primaryKey="customer_id">
    <Table name="customer"/>
    <Level name="Gender" column="gender" uniqueMembers="true"/>
  </Hierarchy>
</Dimension>
```

Cette dimension a une seule hiérarchie et un seul niveau. La dimension prend ses valeurs à partir de la colonne "gender" de la table "customer". La colonne "gender" a deux valeurs 'F' et 'M'. La dimension "Gender" a donc deux membres "[Gender].[F]" et "[Gender].[M]". Pour chaque vente, la dimension "Gender" donne le sexe du client ayant réalisé un achat. Cela s'exprime par la jointure entre la table de faits "sales_fact_1997" et la table de dimensions "customer" sur l'attribut "customer_id".

Une dimension peut contenir plus d'une hiérarchie. La dimension "Time", ci-dessous, contient deux hiérarchies : "Year, Quarter, Month" et "Year, Week, Day".

```
<Dimension name="Time" foreignKey="time_id">
  <Hierarchy hasAll="false" primaryKey="time_id">
    <Table name="time_by_day"/>
    <Level name="Year" column="the_year" type="Numeric" uniqueMembers="true"/>
    <Level name="Quarter" column="quarter" uniqueMembers="false"/>
    <Level name="Month" column="month_of_year" type="Numeric" uniqueMembers="false"/>
  </Hierarchy>
```

```
<Hierarchy name="Time Weekly" hasAll="false" primaryKey="time_id">
  <Table name="time_by_week"/>
  <Level name="Year" column="the_year" type="Numeric" uniqueMembers="true"/>
  <Level name="Week" column="week" uniqueMembers="false"/>
  <Level name="Day" column="day_of_week" type="String" uniqueMembers="false"/>
</Hierarchy>
</Dimension>
```

Une dimension est jointe avec un cube à l'aide de deux colonnes : une colonne dans la table de faits et l'autre dans la table de dimensions. L'élément `<Dimension>` a une clé étrangère (l'attribut `foreignKey`), qui est le même dans la table de faits.

L'élément `<Hierarchy>` a une clé primaire (l'attribut `primaryKey`). Si une hiérarchie est organisée selon plusieurs tables, vous pouvez utiliser l'attribut `primaryKeyTable` pour lever toute ambiguïté.

L'attribut `uniqueMembers` est utilisé pour optimiser les commandes SQL. Si vous savez que les valeurs d'un niveau d'une table de dimensions sont uniques sur toutes les valeurs d'un niveau parent, il faut alors mettre l'attribut `uniqueMembers="true"` et `"false"` sinon. Par exemple, dans la dimension "Time", la hiérarchie "[Year].[Month]" peut avoir la valeur `uniqueMembers="false"` au niveau de "Month" car un même mois peut être présent dans différentes années. D'une autre part, si vous avez la hiérarchie "[Product Class].[Product Name]" et que vous êtes sûr que "[Product Name]" est unique, alors vous pouvez mettre `uniqueMembers="true"`. Dans le cas contraire, mettez `"false"`. Dans le niveau le plus haut de la hiérarchie, l'attribut `uniqueMembers="true"` car il n'y a aucun niveau parent.

Par défaut, toute hiérarchie contient un niveau haut appelé 'All', qui contient un seul membre appelé '(All {hierarchyName})'. Ce membre est le parent de tous les autres membre de la hiérarchie et représente l'agrégation totale sur cette dimension. C'est le membre par défaut de la hiérarchie. Il est le membre utilisé pour calculer les valeurs d'une cellule lorsque cette hiérarchie n'est pas incluse dans un axe ou un "slicer".

Si l'élément `<Hierarchy>` a `hasAll="false"`, le niveau 'All' est supprimé. Le membre par défaut d'une telle dimension est le premier membre du premier niveau. Par exemple, dans la hiérarchie "Time", le membre par défaut serait la première année de la hiérarchie. Le changement du membre par défaut peut prêter à confusion, alors vous devrez le laisser toujours vrai (`hasAll="true"`).

L'élément `<Hierarchy>` a également l'attribut `"defaultMember"`, utilisé pour changer le membre par défaut de la hiérarchie. Ci-dessous, un exemple.

```
<Dimension name="Time" type="TimeDimension" foreignKey="time_id"> <Hierarchy
hasAll="false" primaryKey="time_id" defaultMember="[Time].[1997].[Q1].[1]"/> ...
```

La dimension temps est basée sur l'année, le mois, la semaine et le jour. Elle est implantée d'une manière spéciale sous Mondrian pour prendre en compte la spécificité des fonctions liées aux temps comme :

- ParallelPeriod([level[, index[, member]])
- PeriodsToDate([level[, member]])
- WTD([member])
- MTD([member])
- QTD([member])
- YTD([member])
- LastPeriod(index[, member])

La dimension temps a un attribut "Type" dont la valeur est "TimeDimension". Le rôle d'un niveau d'une dimension temps est indiqué par l'attribut "levelType", dont les valeurs permises sont les suivantes :

Valeur de levelType	Explication
TimeYears	niveau représentant les années
TimeQuarters	niveau représentant les trimestres
TimeMonths	niveau représentant le mois
TimeDays	niveau représentant les jours

9. Requêtes XMLA.

Un entrepôt de données relationnel se résume en une architecture trois tiers :

- Une base relationnelle qui stocke et structure les données ;
- La couche d'analyse OLAP en tant que telle (ou serveur OLAP) ;
- Les interfaces client.

Au sein d'une solution de business intelligence, l'OLAP fait donc figure de logique applicative, couche prenant en charge les requêtes et les traitements métiers.

Les requêtes SQL, adaptées aux systèmes traditionnels OLTP (On-Line Transactional Processing), peuvent difficilement supporter les requêtes OLAP (Voir cependant la partie OLAP de SQL99). En effet, les requêtes décisionnelles sont complexes (plusieurs agrégations et jointures, multidimensionnelles...) et elles opèrent sur des grand volumes de données.

MDX est un langage de requête créé pour l'OLAP. Mondrian implémente le langage MDX. Il charge les données à partir d'une base de données relationnelles et agrège les données dans une mémoire cache. Par ailleurs, Mondrian supporte d'autres spécifications OLAP comme XMLA et JOLAP

Le système OLAP Mondrian se compose de quatre couches allant de l'utilisateur final vers le centre des données. Ces couches (voir la figure ci-dessous) sont : la couche de présentation, la couche de calcul et d'agrégation, et la couche de stockage. Mondrian se charge du calcul et de l'agrégation des données. La partie de calcul analyse, valide et exécute des requêtes MDX. Une requête est évaluée dans des phases multiples. Ce sont d'abord les axes qui sont calculés, puis les valeurs des cellules dans les axes. Un optimiseur de requêtes autorise l'application à manipuler les requêtes existantes dans le cache, plutôt que la construction d'une déclaration MDX à partir de rien pour chaque demande afin d'améliorer les performances. Une agrégation est un ensemble de valeurs de mesures (cellules) dans la mémoire, qualifiée par un ensemble de valeurs de dimensions colonnes. La couche de calcul envoie des requêtes pour des ensembles de cellules. Si les cellules requises ne sont pas dans le cache (mémoire tampon), on envoie une requête à la couche de stockage.

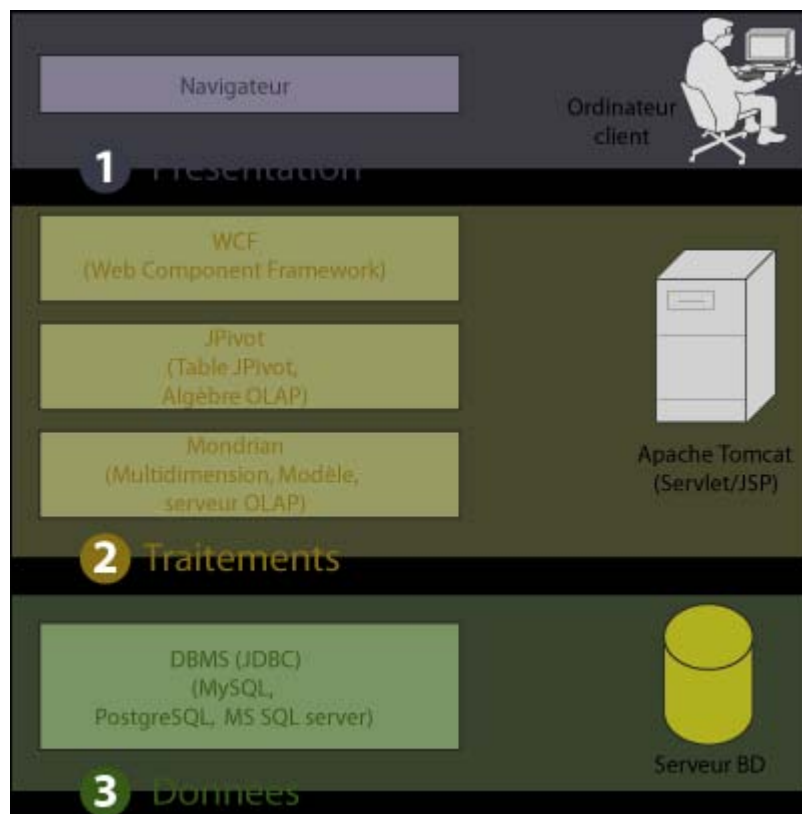


Figure 13 Architecture Mondrian

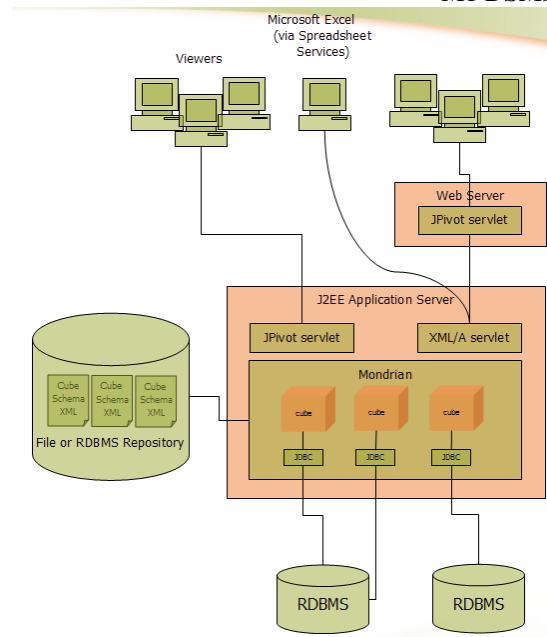
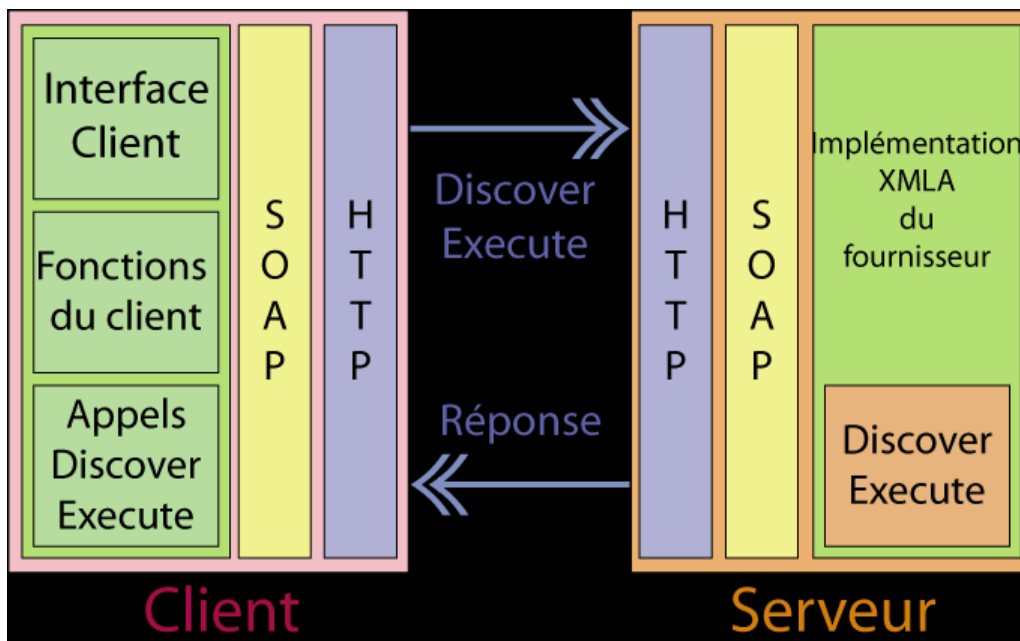


Figure 14 Exemple de déploiement de Mondrian dans un Système d'Information Décisionnel

XMLA (XML for Analysis) est une API XML s'appuyant sur le protocole de communication SOAP. Il a été mis au point par Microsoft, Hyperion et SAS afin de normaliser l'accès aux bases multidimensionnelles. De nombreux acteurs du marché de la BI l'implémentent désormais. Il est notamment largement utilisé en interne par le moteur de SQL Server 2005 pour l'administration des bases, la gestion des sécurités ou encore la modification des données...

XMLA permet de communiquer avec des sources de données multidimensionnelles. La communication se passe via des normes Web comme SOAP (Simple Object Access Protocol) et XML (voir la figure ci-dessous). Le langage d'interrogation des données utilisé est MDX.



L'un des points forts de XMLA est le fait qu'il simplifie la récupération des données par rapport à l'utilisation d'interfaces propriétaires complexes. XMLA comporte seulement deux méthodes : Discover et Execute. Les requêtes se font avec SOAP: on transmet un document XML au serveur qui donne la réponse sous la forme d'un document XML.

- La première méthode sert à récupérer des métadonnées qui décrivent les services pris en charge par le fournisseur XMLA (dans notre cas Mondrian).
- La deuxième sert à exécuter des requêtes et à retourner les données correspondantes.

La méthode *Discover* recherche des méta-données, comme la liste des bases données disponibles ou des détails sur des objets donnés. Les informations recherchées par *Discover* dépendent de la valeur des paramètres passés.

Un appel SOAP est destiné à récupérer une liste de sources de données du serveur. Son code XML est donné ci-dessous.

```
<Discover>
  <RequestType>...</RequestType>
  <Restrictions>...</Restrictions>
  <Properties>...</Properties>
</Discover>
```

Le premier paramètre, *RequestType*, détermine le type d'information renvoyé par la méthode *Discover*. Les types disponibles vous permettent d'obtenir une liste des sources de données disponibles sur le serveur (*DISCOVER_DATASOURCES*), une listes des propriétés concernant une source de données spécifique sur le serveur (*DISCOVER_PROPERTIES*), une liste des types de requêtes pris en charge (*DISCOVER_SCHEMA_ROWSETS*), une liste des mots-clés pris en charge (*DISCOVER_KEYWORDS*), ainsi qu'une constante d'ensemble de lignes de schéma (*schema rowset*) permettant de récupérer le schéma d'un type de données. Nous étudierons dans la suite les différents types de requêtes.

Le deuxième paramètre, *Restrictions*, permet de définir des conditions sur les données retournées par *Discover*. Le paramètre *RequestType* dans l'appel de la méthode *Discover* détermine les champs que le paramètre *Restrictions* peut filtrer. Si vous souhaitez retourner toutes les données d'un appel *RequestType*, laissez le paramètre *Restrictions* vide.

Pour filtrer les données à renvoyer par le serveur, utilisez la syntaxe XML qui suit.

```
<Restrictions>
  <RestrictionList> ... liste_des_restrictions ... </RestrictionList>
</Restrictions>
```

Le paramètre *Properties* apporte des informations supplémentaires sur la requête. Par exemple, *Timeout* spécifie la durée en secondes pendant laquelle le serveur attendra l'aboutissement de la requête *Discover* avant de retourner un message d'expiration de délai. L'ordre de spécification des propriétés n'a pas d'importance. Si vous ne précisez pas de valeurs pour *Properties*, l'appel *Discover* utilisera la valeur par défaut.

Le code XML qui suit montre comment ajouter des propriétés.

```
<Properties>
  <PropertyList> ... Liste_des_propriétés ... </PropertyList>
</Properties>
```

9.1 – DISCOVER_SCHEMA_ROWSETS

Pour avoir la liste des métadonnées que le serveur peut fournir, il faut faire un appel *DISCOVER_SCHEMA_ROWSETS* comme suit.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <Discover xmlns="urn:schemas-microsoft-com:xml-analysis">
      <RequestType>DISCOVER_SCHEMA_ROWSETS</RequestType>
      <Restrictions> </Restrictions>
      <Properties> </Properties>
    </Discover>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requête XML/A 1.

9.2 – DISCOVER_DATASOURCES

Pour découvrir les sources de données, il faut faire une appel *Discover* en mettant dans *RequestType* comme paramètre *DISCOVER_DATASOURCES*. Ci-dessous un exemple d'une requête.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <Discover xmlns="urn:schemas-microsoft-com:xml-analysis">
      <RequestType>DISCOVER_DATASOURCES</RequestType>
      <Restrictions></Restrictions>
      <Properties></Properties>
    </Discover>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requête XML/A 2.

La réponse obtenue à cet appel Discover est le document XML qui suit.

```
<SOAP-ENV:Envelope SOAPENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Header></SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <cxm1a:DiscoverResponse>
      <cxm1a:return>
        <root>
          <xsd:schema targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
            elementFormDefault="qualified">
            <xsd:element name="root">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="row" type="row" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
            <xsd:simpleType name="uuid">
              <xsd:restriction base="xsd:string">
                <xsd:pattern value="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}"/>
              </xsd:restriction>
            </xsd:simpleType>
            <xsd:complexType name="row">
              <xsd:sequence>
                <xsd:element sql:field="DataSourceName" name="DataSourceName"
                  type="xsd:string"/>
                <xsd:element sql:field="DataSourceDescription" name="DataSourceDescription"
                  type="xsd:string" minOccurs="0"/>
                <xsd:element sql:field="URL" name="URL" type="xsd:string" minOccurs="0"/>
                <xsd:element sql:field="DataSourceInfo" name="DataSourceInfo"
                  type="xsd:string" minOccurs="0"/>
                <xsd:element sql:field="ProviderName" name="ProviderName" type="xsd:string"
                  minOccurs="0"/>
                <xsd:element sql:field="ProviderType" name="ProviderType" type="xsd:string"
                  maxOccurs="unbounded"/>
                <xsd:element sql:field="AuthenticationMode" name="AuthenticationMode"
                  type="xsd:string"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:schema>
          <row>
            <DataSourceName>Provider=Mondrian;DataSource=MondrianFoodMart;
            </DataSourceName>
            <DataSourceDescription>Mondrian FoodMart Data Warehouse
            </DataSourceDescription>
            <URL>http://localhost:8080/mondrian/xmla</URL>
            <DataSourceInfo>MondrianFoodMart</DataSourceInfo>
            <ProviderName>Mondrian</ProviderName>
            <ProviderType>MDP</ProviderType>
            <AuthenticationMode>Unauthenticated</AuthenticationMode>
```

```
</row>
</root>
</cxmmla:return>
</cxmmla:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Le document ci-dessus, nous renseigne sur le nom de la source de données, sa description, l'URL du serveur, des informations sur la source, le nom du fournisseur de cette source, son type et son mode d'authentification. A noter qu'avec la version installée de Mondrian à l'IUT (2004), la réponse nous signale que ce n'est pas encore implanté.

9.3 – MDSHEMA_CUBES

Une RequestType de type MDSHEMA_CUBES demande de renvoyer les cubes qui sont stockés sur le serveur. Pour que la requête fonctionne, il faut donner des informations supplémentaires sur la source de données. Ces informations peuvent être obtenues en faisant un appel DISCOVER_DATASOURCES. Les informations sur CATALOG_NAME, SCHEMA_NAME, CUBE_NAME et CUBE_TYPE peuvent être utilisées pour filtrer les réponses.

Ci-dessous est présenté un appel MDSHEMA_CUBES.

```
<Discover xmlns="urn:schemas-microsoft-com:xml-analysis">
  <RequestType>MDSHEMA_CUBES</RequestType>
  <Restrictions> <!-- <RestrictionList>
    <CUBE_NAME>Vols</CUBE_NAME>
  </RestrictionList> -->
</Restrictions>
<Properties>
  <PropertyList>
    <DataSourceInfo>Aeroclub_OLAP</DataSourceInfo>
  </PropertyList>
</Properties>
</Discover>
```

Requête XML/A 3.

La réponse obtenue est une liste de cubes (un cube pour chaque balise <row>) et des informations concernant ces cubes. L'extrait XML de la réponse donne les informations concernant le cube "Vols".

```
<row>
  <CATALOG_NAME>Aeroclub_OLAP</CATALOG_NAME>
  <SCHEMA_NAME>Aeroclub_OLAP</SCHEMA_NAME>
  <CUBE_NAME>Vols</CUBE_NAME>
  <CUBE_TYPE>CUBE</CUBE_TYPE>
  <IS_DRILLTHROUGH_ENABLED>true</IS_DRILLTHROUGH_ENABLED>
  <IS_WRITE_ENABLED>false</IS_WRITE_ENABLED>
  <IS_LINKABLE>false</IS_LINKABLE>
  <IS_SQL_ALLOWED>false</IS_SQL_ALLOWED>
</row>
```

9.4 – MDSHEMA_DIMENSIONS

Une requête MDSHEMA_DIMENSIONS permet d'obtenir des informations sur les dimensions ou une dimension particulière. Les informations concernant CATALOG_NAME, SCHEMA_NAME, CUBE_NAME, DIMENSION_NAME et DIMENSION_UNIQUE_NAME peuvent être utilisées pour filtrer les réponses.

La requête suivante retourne les informations sur les dimensions du cube "Vols".

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <Discover xmlns="urn:schemas-microsoft-com:xml-analysis">
```



```
<RequestType>MDSHEMA_DIMENSIONS</RequestType>
<Restrictions>
  <RestrictionList>
    <CUBE_NAME>Vols</CUBE_NAME>
  </RestrictionList>
</Restrictions>
<Properties>
  <PropertyList>
    <DataSourceInfo>Aeroclub_OLAP</DataSourceInfo>
  </PropertyList>
</Properties>
</Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requête XML/A 4.

9.5 – MDSHEMA_HIERARCHIES

Un appel MDSHEMA_HIERARCHIES permet d'avoir des informations sur les hiérarchies de chaque dimension ou d'une hiérarchie particulière. Les informations concernant CATALOG_NAME, SCHEMA_NAME, CUBE_NAME, DIMENSION_UNIQUE_NAME, HIERARCHY_NAME et HIERARCHY_UNIQUE_NAME peuvent jouer le rôle d'un filtre.

La requête suivante cherche les hiérarchies du cube "Vols".

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <Discover xmlns="urn:schemas-microsoft-com:xml-analysis">
      <RequestType>MDSHEMA_HIERARCHIES</RequestType>
      <Restrictions>
        <RestrictionList>
          <CUBE_NAME>Vols</CUBE_NAME>
        </RestrictionList>
      </Restrictions>
      <Properties>
        <PropertyList>
          <DataSourceInfo>Aeroclub_OLAP</DataSourceInfo>
        </PropertyList>
      </Properties>
    </Discover>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requête XML/A 5.

9.6 – MDSHEMA_LEVELS

Un appel MDSHEMA_LEVELS permet d'avoir des informations sur les niveaux de chaque hiérarchie ou niveau d'un particulier. Les informations concernant CATALOG_NAME, SCHEMA_NAME, CUBE_NAME, DIMENSION_UNIQUE_NAME, HIERARCHY_UNIQUE_NAME, LEVEL_NAME et LEVEL_UNIQUE_NAME peuvent être utilisées pour filtrer la réponse.

La requête suivante retourne les niveaux de la dimensions " Temps" du cube "Vols".

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <RequestType>MDSHEMA_LEVELS</RequestType>
      <Restrictions>
        <RestrictionList>
```

```
<CATALOG_NAME>Aeroclub_OLAP</CATALOG_NAME>
<DIMENSION_UNIQUE_NAME>[ Temps ]</DIMENSION_UNIQUE_NAME>
</RestrictionList>
</Restrictions>
<Properties>
  <PropertyList>
    <DataSourceInfo>Aeroclub_OLAP</DataSourceInfo>
    <Catalog>Aeroclub_OLAP</Catalog>
    <Format>Tabular</Format>
  </PropertyList>
</Properties>
</Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requête XML/A 6.

9.7 – MDSHEMA_MEMBERS

Un appel MDSHEMA_MEMBERS recherche les informations concernant les membres de chaque niveau ou d'un membre particulier. Les informations concernant CATALOG_NAME, SCHEMA_NAME, CUBE_NAME, DIMENSION_UNIQUE_NAME, HIERARCHY_UNIQUE_NAME, LEVEL_UNIQUE_NAME, LEVEL_NUMBER, MEMBER_NAME, MEMBER_UNIQUE_NAME, MEMBER_TYPE, MEMBER_CAPTION et TREE_OP peuvent être utilisées comme un filtre. Si vous ne mettez pas de restrictions, vous obtiendrez tous les membres de la dimension All.

La requête suivante retourne les niveaux de la hiérarchie "Temps" du cube "Vols".

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <RequestType>MDSHEMA_MEMBERS</RequestType>
      <Restrictions>
        <RestrictionList>
          <CATALOG_NAME>Aeroclub_OLAP</CATALOG_NAME>
          <HIERARCHY_UNIQUE_NAME>[ Temps ]</HIERARCHY_UNIQUE_NAME>
        </RestrictionList>
      </Restrictions>
      <Properties>
        <PropertyList>
          <DataSourceInfo>Aeroclub_OLAP</DataSourceInfo>
          <Catalog>Aeroclub_OLAP</Catalog>
          <Format>Tabular</Format>
        </PropertyList>
      </Properties>
    </Discover>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requête XML/A 7.

9.8 – MDSHEMA_MEASURES

MDSHEMA_MEASURES permet d'obtenir des informations sur les mesures. Les valeurs données aux CATALOG_NAME, SCHEMA_NAME, CUBE_NAME, MEASURE_NAME et MEASURE_UNIQUE_NAME peuvent être utilisées pour filtrer la réponse retournée par le système.

L'exemple de la requête ci-dessous, recherche toutes les mesures du cube "Vols".

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```
<SOAP-ENV:Body>
<Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <RequestType>MDSHEMA_MEASURES</RequestType>
  <Restrictions>
    <RestrictionList>
      <CATALOG_NAME>Aeroclub_OLAP</CATALOG_NAME>
    </RestrictionList>
  </Restrictions>
  <Properties>
    <PropertyList>
      <DataSourceInfo>Aeroclub_OLAP</DataSourceInfo>
    </PropertyList>
  </Properties>
</Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requête XML/A 8.

9.9 – Méthode Execute (Mondrian)

La méthode Execute permet d'exécuter des requêtes MDX via XMLA. Un appel Execute doit contenir les informations nécessaires pour établir une connexion au serveur. Le code XML ci-dessous donne la syntaxe d'une méthode Execute.

```
<Execute xmlns="urn:schemas-microsoft-com:xml-analysis">
  <Command>
    <Statement> Requête_MDX </Statement>
  </Command>
  <Properties>
    <PropertyList> Liste_d_informations_nécessaire_pour_se_connecter
      [Liste_d_informations_concernant_le_de_retour_des_résultats]
    </PropertyList>
  </Properties>
</Execute>
```

Le code XML qui suit est un appel Execute de la requête MDX :

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <Execute xmlns="urn:schemas-microsoft-com:xml-analysis"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <Command>
        <Statement>select [Measures].allmembers on Columns from Vols</Statement>
      </Command>
      <Properties>
        <PropertyList>
          <DataSourceInfo>Aeroclub_OLAP</DataSourceInfo>
          <Catalog>Aeroclub_OLAP</Catalog>
          <Format>Multidimensional</Format>
          <AxisFormat>TupleFormat</AxisFormat>
        </PropertyList>
      </Properties>
    </Execute>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requête XML/A 9.

Vous remarquez aussi que le données sont affichées sous format multidimensionnel comme souhaité dans la requête. On peut choisir l'affichage tabulaire en remplaçant <Format>Multidimensional</Format> par <Format>Tabular</Format>.

A noter qu'à l'heure actuelle (Septembre 2015), Mondrian n'intègre pas directement la fonctionnalité de mise à jour des données d'un cube par les utilisateurs (*writeback*), fonctionnalité intéressante dans une analyse budgétaire par une méthode What-if.

L'application web de démonstration est basée sur Mondrian 1.0. Cette version trop vieille ne permet pas aux clients XMLA extérieurs de s'y connecter. Sur le serveur de la plateforme décisionnelle de l'UBS (bureau à distance), une distribution Mondrian packagée avec la solution pentaho 3.8.0 est aussi disponible.

Pour interroger ce serveur Mondrian via XML/A, divers clients open-source lourds et légers sont disponibles :

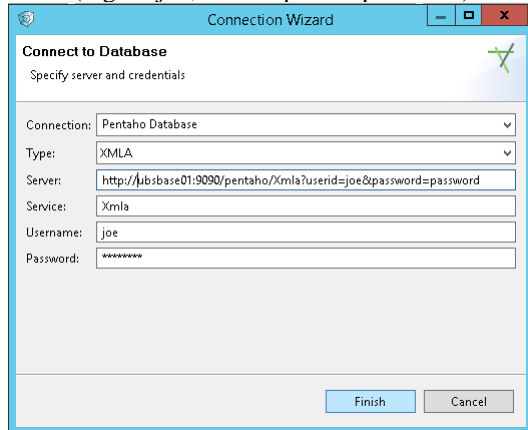
En lourd :

- [PalOOCa](#), extension pour OpenOffice.org, Apache OpenOffice et LibreOffice ;
- [JPalo](#) ;
- [iReport](#) ;
- Connecteur [XMLAConnect](#) est une implémentation de la spécification OLEDB pour OLAP (ODBO). Le fournisseur se connecte à des sources de données XMLA. Utilisé par MS Excel.

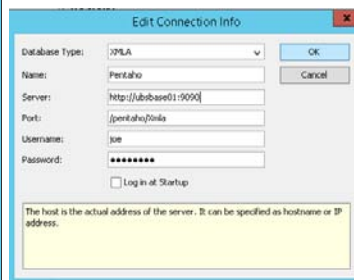
En Web :

- JPivot/Pentaho ;
- PAT (Pentaho Analysis Tools) ;
- saiku ;
- [Xmla4js](#) ;
- [JPalo Web Client](#).

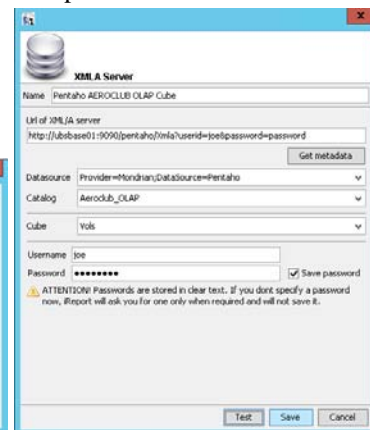
Voici quelques écrans de configurations XMLA : <http://ubsbase04:9090/pentaho/Xmla?userid=joe&password=password>
JPalo (login : joe, mot de passe : password)



PalOOCa

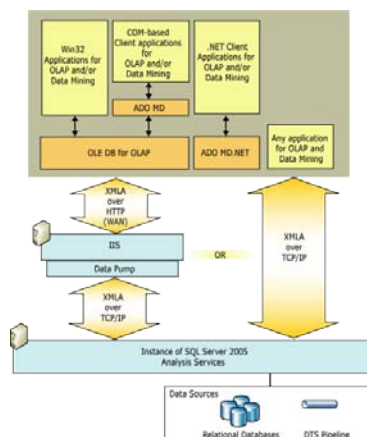


iReport



9.10 – Méthode Execute (Microsoft SQL Server Analysis Services 2008r2)

La méthode **Execute** permet non seulement de récupérer des données mais aussi modifier les cubes OLAP via l'ordre MDX UPDATE CUBE. Pour mémoire, l'implantation de Microsoft du langage MDX a une partie spécifique de définition d'un cube, reprise ni par Sas Institute, ni par Mondrian.



Sur le serveur décisionnel statdec2.univ-ubs.fr, la pompe OLAP qui permet le requetage *XMLA over HTTP*, est configurée pour répondre à l'adresse (methode Post) : <http://UBSBASE04:81/olap/msmdpump.dll>

Un moyen de l'intégrer à Jpivot est de faire la requête `MD_sqlserverXMLAFoodMart.jsp` dans le répertoire `WEB-INF/queries/` et de l'invoquer avec : `JPivot pivot table`.

```
<%@ page session="true" contentType="text/html; charset=ISO-8859-1" %>
<%@ taglib uri="http://www.tonbeller.com/jpivot" prefix="jp" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<jp:xmqlaQuery id="query01" uri="http://UBSBASE4:81/olap/msmdpump.dll"
dataSource="Provider=MSOLAP.4;Data Source=local;"
catalog="Foodmart 2000 time plus 10">
    with Member [Measures].[Warehouse Margin] as '[Measures].[Warehouse Sales]-
[Measures].[Warehouse Cost]' select {[([Time].[Quarter].Members)} on columns,
{([Warehouse].[Country].USA)} on rows from Warehouse Where ([Measures].[Warehouse
Margin])
</jp:xmqlaQuery>
<c:set var="title01" scope="session">MS SQL Server Analysis Services Cube:
[Sales](Foodmart 2000 time plus 10)</c:set>
```

Un autre moyen d'interroger le cube est d'utiliser le plug-in Palo pour OpenOffice.org PalOOCa installé sur le serveur statdec3.univ-ubs.fr. avec database type : XMLA, Server : <http://UBSBASE04:81>, port : /olap/msmdpump.dll, username : EtudiantXX, password : P@ssw0rd.

A noter que la méthode Execute accepte aussi une requête en DMX (Data Mining Extensions), un langage de requête pour les modèles de Data Mining pris en charge par le produit Microsoft SQL Server Analysis Services. Comme le SQL, il prend en charge un langage de définition de données, un langage de manipulation de données et un langage de requête de données, tous les trois avec une syntaxe à la SQL. DMX est utilisé pour créer et faire apprendre des modèles d'exploration de données, et pour les afficher ou les décrire, pour les gérer et pour faire des prédictions à partir d'eux. Voici une requête de prédiction singleton concernant la cliente donnée pour savoir si elle sera intéressée par les produits de prêt à domicile :

```
SELECT [Loan Seeker], PredictProbability([Loan Seeker]) FROM [Decision Tree] NATURAL
PREDICTION JOIN (SELECT 35 AS [Age], 'Y' AS [House Owner], 'M' AS [Marital Status],
'F' AS [Gender], 2 AS [Number Cars Owned], 2 AS [Total Children], 18 AS [Total Years
of Education] )
```

DMX dans XML/A est un des standards de web services pour le data mining. Un autre standard pour les web services en fouille de données est JDMWS (Java Data Mining Web Services) dont les requêtes sont elles aussi formulée en SOAP et qui a aussi une définition WSDL. Voici une requête du fournisseur de modèles de scores et de classification [ADAPA®](#) publiés en tant que web services sur le cloud conforme à ce standard :

```
<soap:Body xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<jdmws:executeTaskElement xmlns:jdm="http://www.jsr-73.org/2004/JDMSchema"
xmlns:jdmws="http://www.jsr-73.org/2004/webservices"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<task xsi:type="jdm:RecordApplyTask" modelName="Iris_NN">
<jdm:recordValue name="sepal_length">
<jdm:value xsi:type="jdm:DecimalValue" decimal="5.1" />
</jdm:recordValue>
<jdm:recordValue name="sepal_width">
<jdm:value xsi:type="jdm:DecimalValue" decimal="3.5" />
</jdm:recordValue>
<jdm:recordValue name="petal_length">
<jdm:value xsi:type="jdm:DecimalValue" decimal="1.4" />
</jdm:recordValue>
<jdm:recordValue name="petal_width">
<jdm:value xsi:type="jdm:DecimalValue" decimal="0.2" />
</jdm:recordValue>
</task>
</jdmws:executeTaskElement>
```

</soap:Body> .

9.11 – Méthode Execute (SAS Institute)

SAS Institute propose un appel de processus stockés via la méthode XMLA. D'un point de vue multidimensionnel, aujourd'hui, SAS ne propose pas d'accès direct aux données multidimensionnelles tel que présenté dans ce document. Cependant le processus stockés peut récupérer via une requête MDX les données [PROC SQL; CONNECT TO OLAP(); QUIT;] du serveur SAS OLAP ou modifier depuis SAS V9.2 le cube OLAP [PROC OLAP cube=Cube Name <DELETE OPTION>;]. Depuis la version 9.2, les processus stockés sont publiables en tant que web services sans passer par XML/A. La proc SOAP permet d'invoquer un web service depuis SAS *Foundation*. La publication se fait via un assistant qui utilise Apache Axis 2. Depuis la version 9.3 de SAS, la publication de chaque processus stockée est systématique et automatique. Elle se fait en utilisant les web services du framework Spring. L'assistant permet de grouper plusieurs processus stockés en un seul endpoint.

9.12 – Exercice

L'exercice suivant vous permet de manipuler des requêtes XMLA pour découvrir les métadonnées et d'exécuter des requête MDX via XML. Pour répondre aux questions ci-dessous, utilisez l'outil « XML for Analysis tester » de Mondrian.

1. Vérifier que le cube "Vols" existe.
 2. Quelles sont les dimensions du cube "Vols" ?
 3. Combien de hiérarchies a la dimension "Temps" du cube "Vols" ?
 4. Donnez la structure de chacune des hiérarchies que vous avez trouvée pour la dimension "Temps" ?
 5. Quelles sont les indicateurs du cube "Vols" ?
 6. Proposez les requêtes équivalentes aux requêtes SQL 1999 en XML/A. Les fonctions CrossJoin, Union, Hierarchize et Children pourront être utilisées pour la partie MDX.
- En dehors de l'outil « XML for Analysis tester » de Mondrian :
7. Sous Eclipse, importer l'archive war de mondrian comme projet java web dynamique et déployez le dans un serveur tomcat 5.5. Construire et exécuter le projet sur le serveur.
 8. Ecrivez à l'aide de [olap4j](#) un client XML/A qui envoie des requêtes à notre serveur.
 9. Déroutez sous Eclipse les requêtes XML/A du client pour les afficher avec le moniteur TCP/IP intégré d'Eclipse.
 10. Etudier le fichier wsdl de la [spécification de XML/A](#). Editez le sous Eclipse pour l'adapter à notre serveur.

10. Eléments de savoir-faire : installation locale d'un serveur Mondrian XMLA.

A noter qu'une vidéo de l'installation est téléchargeable [ici](#), traite des points 9.1 à 9.9.

10.1 –Installation des fichiers

Copie et décompression des fichiers du TP

Copie des fichiers nécessaires au TP

Avec la séquence de touches [Ctrl+C], [Ctrl+V], copier les fichiers *.zip & *.war du répertoire
G:\vannes\prof\lcsd01\MD\5sms vers P:\votreEspaceDeTP\natifwin.

De fait, remplacer dans toute cette présentation d:\natifwin (ici répertoire de destination) par
P:\votreEspaceDeTP\natifwin

Décompression des fichiers :

Décompresser dans votre répertoire natifwin tous les fichiers zip.

Utilisez pour cela PowerArchiver 2009, installé à l'UBS.

Il suffit de sélectionner le fichier zip et l'ouvrir par le menu contextuel.

Puis cliquer sur le bouton "Extract" et choisir pour destination P:\votreEspaceDeTP\natifwin.

10.2 –Invite de commandes - Environnement

Edition et exécution de commandes

Mise à jour de vos variables d'environnement

Ouvrir dans le bloc note le fichier P:\votreEspaceDeTP\natifwin\setJavaANT.bat

Modifier la variable d'environnement NATIFWIN

Mettre la valeur P:\votreEspaceDeTP\natifwin

Sauvegarder le fichier.

Ouverture d'une invite de commande

Utilisez la commande de Windows Démarrer / Exécuter ...

Entrez "CMD" et utilisez le bouton [OK]

Se positionner dans l'invite de commande au niveau du répertoire P:\votreEspaceDeTP\natifwin.

Exécutez le fichier de commandes P:\votreEspaceDeTP\natifwin\setJavaANT.bat :

Les variables d'environnement JAVA_HOME, NATIFWIN sont disponible pour cette invite de commandes.

10.3 –Invite de commandes - Démarrage du serveur Tomcat

Déploiement d'une application web lors du démarrage du serveur

Déploiement sur le serveur Tomcat de l'archive web (*.war) csdia3

Copier via l'explorateur Windows le fichier P:\votreEspaceDeTP\natifwin\csdia3.war dans
P:\votreEspaceDeTP\natifwin\apache-tomcat-5.5.28\webapps.

Démarrer le serveur Tomcat en exécutant le fichier P:\votreEspaceDeTP\natifwin\apache-tomcat-5.5.28\bin\startup.bat.

Votre serveur Tomcat procède au déploiement de l'archive csdia3.war en la décompressant.

Attendre le message "Server startup in ... ms" : votre serveur Tomcat peut répondre à vos requêtes.

10.4 –Invite de commandes - Test de l'application web

L'application web permet l'apprentissage de SQL 99 et de MDX

Test de votre serveur OLAP Mondrian

Ouvrir dans un navigateur l'URL "http://localhost:8080/csdia3/".

Testez jpivot avec des données de tests. Il faut procéder 2 fois avant de réussir le test : jpivot est long à mettre en place ce qui provoque un premier échec.

Testez jpivot avec des données provenant d'un cube mondrian créé par Michel Dubois.

Faites une navigation OLAP. Affichez la requête MDX.

Apprentissage MDX

Utilisez l'exemple créé par Michel Dubois.

Lisez les explications MDX du sujet du TP.

Toutes les requêtes ont un numéro dans la légende.

Ce numéro correspond au numéro dans la liste de sélection.

Appuyez sur le bouton "show query" pour afficher la requête de la sélection.

Appuyez sur le bouton "process MDX query" pour exécuter la requête.

Un tableau (simple au début, croisé ensuite) constitue le résultat.

Faire cela pour les 22 requêtes.

10.5 –Invite de commandes - Arrêt du serveur Tomcat

Arrêt sans risque du serveur Tomcat

Arrêter le serveur Tomcat

en exécutant le fichier P:\votreEspaceDeTP\natifwin\apache-tomcat-5.5.28\bin\shutdown.bat ou
%NATIFWIN%\apache-tomcat-5.5.28\bin\shutdown.

Redémarrage du serveur Tomcat

Le redémarrage du serveur Tomcat se fait par la commande %NATIFWIN%\apache-tomcat-5.5.28\bin\startup.

10.6 –Eclipse - Déclaration d'un serveur Tomcat

Ajout dans votre workspace Eclipse d'un serveur Tomcat

Remarque : lors de l'ouverture d'Eclipse vous avez choisi comme workspace : P:\votreEspaceDeTP\workspace.

Déclaration d'un serveur Tomcat dans le module WTP d'éclipse

à l'aide de la commande Windows\Préférences\Servers\Runtime Environments

On ajoute un serveur Tomcat 5.5.

On déclare comme répertoire d'installation P:\votreEspaceDeTP\natifwin\apache-tomcat-5.5.28.

10.7 –Eclipse - Importation d'une application java web

Importation dans votre workspace Eclipse de l'archive war en projet java web dynamique

On importe une archive WAR dans notre workspace.

Choisir le fichier P:\votreEspaceDeTP\natifwin\csdia3.war.

Il suffit de cliquer sur [Finish] sans sélectionner aucune archive JAR : on ne veut pas créer un projet Eclipse par bibliothèque java utilisée par l'application web.

Le projet Java Web Dynamique csdia3 (nom de l'archive war) est en train d'être créé dans votre workspace.

10.8 –Eclipse - Déploiement du projet dans Tomcat et test

La publication de votre projet web dynamique précède le démarrage du serveur de votre workspace.

Déploiement du projet dans Tomcat et test

Déployer votre projet csdia3 sur votre serveur Tomcat déjà déclaré dans votre workspace.

Ouvrir l'explorateur de projet Eclipse. Sélectionner le projet csdia3.

Dans le menu contextuel obtenu par le clic du bouton droit de la souris, choisir la commande Run AS / Run on server.

Vérifier que csdia3 se trouve dans la partie à droite. Si ce n'est pas le cas, faites le basculer à droite avec le bouton [Add >].

Attendre la fin de la publication (déploiement) du projet actuel sur le serveur tomcat.

On bascule sur le "workbench Eclipse".

Une fenêtre navigateur interne est ouverte sur notre application web.

Les serveurs déclarés dans notre workspace sont affichés en bas. On nous signale que Tomcat est démarré.

10.9 –Test de l'application web d'interrogation XML/A.

Un premier contact avec les messages XML du protocole SOAP

Les requêtes XML/A ne sont pas forcément et n'ont pas les numéros des requêtes de la section 8 de votre sujet de TP.

On remarque que le protocole HTTP a été gommé : on n'a que le corps HTTP *i.e.* le document sans les entêtes). Normalement SOAP est basé sur HTTP 1.1 (recommandation WS-I).

Remarque : SOAP (Simple Object Access Protocol) est un protocole d'échange inter-applications indépendant de toute plateforme, basé sur le langage XML. Un appel de service SOAP est un flux ASCII encadré dans des balises XML et transporté dans le protocole HTTP.

Est-ce que cette page JSP (application client) que l'on manipule fait réellement des échanges réseaux avec le fournisseur XML/A qui appartient à la même application java web (archive war) ? Mystère ... Vous pouvez consulter, à partir de l'index de l'archive web, le code source de la page JSP mais cela ne vous renseignera pas.

Après avoir étudié cette section et exécuté les requêtes XML/A, formulez vos propres requêtes XML/A pour répondre aux questions 1 à 6 du paragraphe 8.12.

A noter que le point 7 a déjà été traité.

10.10 – Programmation de clients XML/A

Utilisation de l'API de JPivot et de l'API Olap4j avec le pilote XML/A livré avec sa distribution.

La vidéo n°2, téléchargeable [ici](#), traite des sujets suivants :

- Exécution des clients JPivot et olap4j
- Interception des messages entre les clients XML/A et le fournisseur XML/A via le moniteur TCP-IP
- Modification des clients XML/A pour envoyer les requêtes au moniteur
- Etudes des messages SOAP
- Ouverture du fichier WSDL dans l'éditeur WSDL d'Eclipse
- Adaptation naïve du WSDL pour Mondrian
- Test du fichier WSDL dans l'explorateur des web services d'Eclipse (partie spécialisée WSDL). Un aller-retour a été nécessaire car on avait oublié de sauvegarder dans la vidéo le fichier wsdl !
- Le test échoue.

A noter que vous pouvez télécharger et installer sur votre espace de TP un client XML/A tels que [iReport](#) de JasperSoft. Choisissez la version générique. Une copie locale est téléchargeable [ici](#).

10.11 – Edition du fichier WSDL

La transformation naïve de "xsd :xmlDOM" en "xsd :string" n'est pas concluante.

Une recherche internet nous révèle que "xsd :xmlDOM" du toolkit Microsoft semble correspondre à "xml-type :Vector" de Apache 2. En vous aidant des nouveaux fichiers WSDL de Microsoft et de SAS Institute, faite la correction et testez avec succès votre fichier mondrian.wsdl dans l'explorateur de web services d'Eclipse.

Remarquons que bien que le fournisseur XML/A soit dans l'impossibilité de répondre à une requête MDX, ce dernier ne considère pas cela comme une erreur SOAP (code HTTP de la classe 500 et bloc Fault).

La vidéo n°3, téléchargeable [ici](#), traite de cette correction et du test avec succès.