# Couplage d'un HyperCube SOLAP et d'un outil de visualisation

Michaël Tranchant

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHÔNE-ALPES

CENTRE D'ENSEIGNEMENT DE GRENOBLE

---

MÉMOIRE

présenté par **Michaël Tranchant**

en vue d'obtenir

LE DIPLÔME D'INGENIEUR C.N.A.M.

en INFORMATIQUE

---

# Couplage d'un HyperCube SOLAP et d'un outil de visualisation

Soutenu le 22 février 2013

---

JURY

| | |
|---|---|
| Président : | M. Eric Gressier-Soudan |
| Membres : | M. Claude Genier |
| | M. Jean-Pierre Giraudin |
| | M. Jérôme Gensel |
| | M. Thierry Humbert |

**CONSERVATOIRE NATIONAL DES ARTS ET METIERS**

**CENTRE REGIONAL RHÔNE-ALPES**

**CENTRE D'ENSEIGNEMENT DE GRENOBLE**

---

MÉMOIRE

présenté par **Michaël Tranchant**

en vue d'obtenir

LE DIPLÔME D'INGENIEUR C.N.A.M.

en INFORMATIQUE

---

# Couplage d'un HyperCube SOLAP et d'un outil de visualisation

Soutenu le 22 février 2013

---

Les travaux relatifs à ce mémoire ont été effectués dans l'équipe STEAMER du LIG (Laboratoire d'Informatique de Grenoble) sous la direction de M. Jérôme GENSEL.

# Remerciements

Je tiens à adresser les premiers remerciements aux personnes qui me font l'honneur de participer au jury de ce mémoire : Monsieur Eric Gressier-Soudan, professeur au CNAM Paris et président de ce jury, Monsieur Claude Genier, responsable régional EICNAM, Monsieur Jean-Pierre Giraudin, professeur à l'université de Grenoble, Monsieur Jérôme Gensel, professeur à l'université Pierre Mendès France de Grenoble, Monsieur Thierry Humbert, responsable du pôle ESIA de la société EdifiXio.

Je tiens tout particulièrement à remercier mon tuteur, Monsieur Jérôme Gensel, professeur à l'université Pierre Mendès France de Grenoble, pour m'avoir proposé de travailler dans les domaines de l'informatique décisionnelle et de la géomatique ainsi que pour m'avoir accueilli au sein de l'équipe STEAMER dont il a la direction.

Je remercie également les membres de cette équipe pour leur accueil et leur disponibilité. Je remercie particulièrement Benoit Le Rubrus, pour ses conseils, son soutien et ses relectures ; Laurent Poulenard, Anton Telechev, Anthony Hombiat et Philippe Genoud pour nos échanges dans le cadre du projet ESPON 2013 Database ; Mesdames Paule-Annick Davoine et Cécile Saint-Mars pour leurs recommandations en sémiologie.

Merci aussi à Monsieur Thierry Humbert, de la société EdifiXio, pour s'être montré compréhensif et m'avoir encouragé tout au long de ma formation CNAM ; à l'équipe COPAIN de l'IRSTEA de Clermont-Ferrand et plus particulièrement à Sandro Bimonte pour nos discussions autour de SOLAP ; à la société Intelli³, et en particulier à Martin Nadeau, pour leur disponibilité et leur amabilité ; à Alban Chazot, de l'équipe PimLIG, pour ses connaissances en système.

Un grand merci à l'association des ingénieurs du CNAM (AI-CNAM) pour leurs services de relectures et de soutenances à l'oral, et plus particulièrement à Dominique Fournier.

Enfin, je tiens à remercier mon amie et mon fils pour m'avoir entouré de leur affection et accordé leur soutien indéfectible au cours de ma formation CNAM et encore plus pendant la réalisation de ce mémoire.

# Sommaire

**Introduction** ..... **1**

**Partie I - Présentation** ..... **3**

A. Contexte ..... 3

1. Groupe de recherche HyperCarte ..... 3
2. Programme ESPON 2013 ..... 5
3. Directive INSPIRE ..... 6
4. Informatique décisionnelle ..... 7
5. Visualisation des données ..... 8

B. Enjeux et objectifs du projet ..... 8

C. Plan du mémoire ..... 9

**Partie II - Etat de l'art** ..... **11**

A. Informatique géodécisionnelle ..... 11

1. Principes de l'informatique décisionnelle ..... 11
2. Modélisation multidimensionnelle ..... 12
3. Architecture de l'informatique géodécisionnelle ..... 15

a. Sources de données ..... 16

b. Outils d'extraction, transformation et chargement ..... 16

c. Entrepôt de données ..... 17

d. Traitement analytique en ligne OLAP ..... 24

i. Opérateurs OLAP ..... 25

ii. SOLAP : l'OLAP appliqué aux données spatiales ..... 27

iii. XML for Analysis, une méthode d'accès ..... 28

iv. MultiDimensional eXpressions ..... 29

e. Outils de visualisation ..... 33

f. Métadonnées ..... 33

B. Visualisation de l'information ..... 34
1. Visualisation de données provenant de systèmes décisionnels ..... 34
a. Moyens ..... 34
b. Outils ..... 36
2. Particularités de l'information géographique ..... 41
a. Définition et rappels ..... 41
b. Carte ..... 42
c. Outils ..... 45
d. Conclusion ..... 49
C. Synthèse ..... 50

**Partie III. Démarche et proposition ..... 51**

A. Besoins ..... 51
B. Proposition ..... 52
1. Vue globale ..... 52
2. Architecture ..... 52
a. Architecture métier ..... 52
b. Architecture logicielle ..... 53
3. Démarche ..... 54
a. Découpage en tâches ..... 54
b. Méthodologie ..... 56
c. Maquettes et prototypes ..... 58
C. Décisions préalables ..... 59
1. Gestion des données ..... 59
a. Stockage des données spatiales ..... 59
b. Stockage des données du cube ..... 61
c. Accès aux données ..... 61
2. Serveur SOLAP ..... 62
3. Java et serveur d'application ..... 63
4. Fonctionnement global ..... 64
D. Feuille de route ..... 64

**Partie IV. Réalisation** ..... **67**

A. Outils et méthodes de développement ..... 67

1. Gestion de projets Maven ..... 67

2. Tests unitaires JUnit ..... 69

3. Intégration continue Jenkins ..... 70

B. Chaîne géodécisionnelle ..... 71

1. Modélisation du cube ..... 71

a. Expérimentation ..... 71

b. Travaux de l'UAB ..... 71

c. Notre hypercube ..... 76

2. Spatial OLAP ..... 77

C. Serveur HyperAtlas³ ..... 79

1. Vue d'ensemble ..... 79

2. Modules ..... 80

a. Service ..... 80

i. Architecture ..... 81

ii. Technologies ..... 82

iii. Génie logiciel ..... 83

b. XmlaClient ..... 84

c. Business unit ..... 86

i. Sources multiples ..... 86

ii. Décision de spatialisation des requêtes ..... 87

iii. Cache ..... 88

d. MDXBuilder ..... 88

i. Architecture ..... 88

ii. Construction automatique des requêtes ..... 90

iii. Spatialisation ..... 94

e. MapBuilder ..... 96

i. Interprétation des données ..... 96

ii. Remaniement d'HyperAtlas ..... 97

iii. Génération de carte ..... 97

iv. Représentation des données ..... 98

f. Commons ..... 102

3. Documentation ..... 104

D. Client HyperAtlas³ ..... 104
1. Technologie ..... 105
a. Connexion aux services ..... 105
b. Génie logiciel ..... 106
2. Interface ..... 108
a. Sélection des niveaux et paramétrage des axes ..... 110
b. Tableau croisé multidimensionnel ..... 112
c. Représentation par diagrammes ..... 114
d. Rendu cartographique ..... 114
3. Guide utilisateur ..... 116

**Partie V - Conclusion ..... 117**

A. Rappel des objectifs ..... 117
B. Synthèse des réalisations ..... 117
C. Synthèse globale ..... 119
D. Perspectives ..... 120
E. Bilan personnel ..... 121

**Annexes ..... 123**

**Glossaire ..... 137**

**Bibliographie ..... 139**

# Liste des figures

**Figure I-1 :** Découpage hiérarchique des données ..... 5

**Figure II-1 :** Dimensions et leur hiérarchie respective ..... 13

**Figure II-2 :** Niveaux et membres de la dimension temporelle ..... 13

**Figure II-3 :** Représentation multidimensionnelle ..... 14

**Figure II-4 :** Représentation en système de coordonnées d'un cube contenant les données agrégées ..... 15

**Figure II-5 :** Ensemble des briques composant un système décisionnel ..... 16

**Figure II-6 :** Architecture de l'approche ROLAP ..... 18

**Figure II-7 :** Architecture de l'approche MOLAP ..... 19

**Figure II-8 :** Architecture de l'approche HOLAP ..... 20

**Figure II-9 :** Structure en étoile ..... 21

**Figure II-10 :** Modèle physique de données d'une dimension « Vendeur » en schéma Etoile ..... 22

**Figure II-11 :** Structure en flocon ..... 22

**Figure II-12 :** Représentation physique d'une dimension « Vendeur » en schéma Flocon ..... 23

**Figure II-13 :** Structure mixte ..... 23

**Figure II-14 :** Structure en constellation ..... 23

**Figure II-15 :** Opérateurs de forage et de remontée ..... 25

**Figure II-16 :** Opérateur de forage latéral ..... 26

**Figure II-17 :** Opérateur pivot ..... 26

**Figure II-18 :** Opérateur de perçage ..... 26

**Figure II-19 :** Opérateurs de taillage ..... 27

**Figure II-20 :** Représentation des données spatiales, de gauche à droite, respectivement descriptive, géométrique et mixte ..... 27

**Figure II-21 :** Exemple de n-uplet à trois dimensions ..... 30

**Figure II-22 :** Exemple de requête MDX mettant en œuvre une fonction de niveau. ..... 30

**Figure II-23 :** Exemple de requête MDX mettant en œuvre un opérateur d'ensemble. ..... 31

**Figure II-24 :** Exemple, sous Microsoft Excel, de tableau croisé à trois dimensions : NUTS et Bassins versants en lignes, mesures en colonnes ..... 35

**Figure II-25 :** Interface de JPivot ..... 37

**Figure II-26 :** Interface de Saiku, vue tabulaire ..... 38

**Figure II-27 :** Interface de Saiku, vue graphique ..... 38

**Figure II-28 :** Interface de JRubik ..... 39

**Figure II-29 :** Interface d'icCube ..... 40

**Figure II-30 :** Représentation UML des données géographiques en mode vecteur selon l'ISO19107 ..... 42

**Figure II-31 :** Cartes à symboles : cercles, pictogrammes, diagrammes circulaires et à barres verticales ..... 43

**Figure II-32 :** Cartes choroplèthes : plage simple teinte et plage bipolaire inversée ..... 44

**Figure II-33 :** Cartes à symboles de tailles proportionnels et choroplèthe à plage simple teinte dans HyperAtlas ..... 46

**Figure II-34 :** Cartes de synthèses des trois écarts, deux à deux. Il s'agit de cartes choroplèthes à plusieurs teintes dans HyperAtlas. ..... 46

**Figure II-35 :** Carte choroplèthe dans Map4Decision ..... 47

**Figure II-36 :** Représentation de polluants par diagrammes circulaires et par multicartes dans Map4Decision ..... 48

**Figure III-1 :** Vue métier de l'application ..... 53

**Figure III-2 :** Deux architectures logicielles possibles : « client serveur » et « n tiers » ..... 53

**Figure III-3 :** Vue logicielle après les premières itérations ..... 57

**Figure III-4 :** Vue logicielle : modules enrichis ..... 57

**Figure III-5 :** Vue logicielle : modules spécialisés ..... 57

**Figure III-6 :** Première maquette de l'interface manipulable par le client ..... 58

**Figure III-7 :** Accès aux données spatiales « à la volée » ..... 61

**Figure III-8 :** Accès aux données spatiales par une connexion annexe ..... 62

**Figure III-9 :** Proposition de fonctionnement des briques logicielles ..... 64

**Figure IV-1 :** Architecture type d'un projet généré par Maven, depuis l'IDE Eclipse. ..... 68

**Figure IV-2 :** Transfert des données socio-économiques et environnementales aux supports géographiques ..... 72

**Figure IV-3 :** Superposition et intersection des objets géographiques....73

**Figure IV-4 :** Zone maximum....74

**Figure IV-5 :** Illustration du calcul proportionnel....74

**Figure IV-6 :** Calcul proportionnel....74

**Figure IV-7 :** Calcul proportionnel et pondéré....75

**Figure IV-8 :** Structure mixte mise en place pour notre entrepôt....76

**Figure IV-9 :** Positionnement du serveur HyperAtlas³ vis-à-vis des autres éléments....79

**Figure IV-10** : Schématisation des modules d'HyperAtlas³....80

**Figure IV-11 :** Diagramme de classes simplifié référençant les annotations des interfaces et classes impliquées dans les services Web....82

**Figure IV-12 :** Diagramme de séquence simplifié du service getSchemaInfo....84

**Figure IV-13 :** Diagramme de classe partiel du module XmlaClient illustrant le patron de conception « fabrique »....85

**Figure IV-14 :** Gestion d'une requête par le service executeRequest....87

**Figure IV-15 :** Diagramme de classes du module « MDXBuilder »....89

**Figure IV-16 :** Diagramme de flux simplifié pour la construction d'un axe....93

**Figure IV-17 :** Extraction des géométries des pays (en lignes) sur la dernière colonne (Geom)....94

**Figure IV-18 :** Emplacements des informations de géométries....96

**Figure IV-19 :** Etapes de génération d'une carte. D'abord, fond de carte ; puis aplat des cartes choroplèthes, symboles de tailles proportionnelles ; enfin légende....98

**Figure IV-20 :** Recalcul de la surface occupée par les indicateurs lors d'un grossissement....98

**Figure IV-21 :** Carte à disques de taille proportionnelle....99

**Figure IV-22 :** Représentation par diagrammes circulaires des données de la France et de la Belgique à des fins de comparaisons....100

**Figure IV-23 :** Carte à diagrammes en crête de coq....101

**Figure IV-24 :** Hiérarchie des classes constituant la méta-structure....102

**Figure IV-25 :** Diagramme de séquence simplifié des services exposés par HyperAtlas³ et des objets échangés....103

**Figure IV-26 :** Configurations possibles pour l'interconnexion du client et du serveur....105

**Figure IV-27 :** Diagramme de séquence du patron MVC....108

**Figure IV-28 :** Architecture du patron MVC....108

**Figure IV-29 :** Vue complète de l'interface principale du client ..... 109

**Figure IV-30 :** Menu contextuel d'un niveau assigné à l'axe « ligne » ..... 110

**Figure IV-31 :** Menu contextuel d'un niveau non assigné ..... 110

**Figure IV-32 :** Contrainte sur le déplacement de niveaux ..... 111

**Figure IV-33 :** Métadonnées du niveau NUTS 0 ..... 111

**Figure IV-34 :** Principales options du client HyperAtlas[3] ..... 111

**Figure IV-35 :** Hiérarchisation des niveaux dans les axes ..... 113

**Figure IV-36 :** Sélection et classement des membres d'un niveau ..... 113

**Figure IV-37 :** Impact des niveaux en colonne et en ligne sur l'abscisse et l'ordonnée du diagramme ..... 114

**Figure IV-38 :** Outils généraux du bloc carte ..... 115

**Figure IV-39 :** Outils dédiés à la génération de cartes à symboles à taille proportionnelle ..... 115

**Figure IV-40 :** Outils dédiés à la génération de cartes choroplèthes ..... 115

**Figure IV-41 :** Affichage du guide utilisateur dans un navigateur ..... 116

**Figure V-1 :** Répartition des tests par modules ..... 118

**Figure A-C-1 :** Page d'accueil du site ..... 129

**Figure A-C-2 :** Page décrivant les dépendances entre les modules ..... 129

**Figure A-C-3 :** JavaDoc et diagramme de classe interactif de la classe MdxCondition ..... 130

**Figure A-C-4 :** Synthèse de la couverture de code par les tests ..... 130

**Figure A-C-5 :** Informations détaillées, module par module ..... 131

# Liste des tableaux

**Tableau II-1 :** Extraction d'une table de dimension « Vendeur », limitée à cinq entrées. ..... 22

**Tableau II-2 :** Extractions des tables factices de dimension dans le cas d'un schéma Flocon. ..... 23

**Tableau II-3 :** Liste des messages « discover » les plus rencontrés ..... 29

**Tableau II-4 :** Exemple de résultat à la requête de la figure II-23 ..... 32

**Tableau III-1 :** Liste des tâches déterminées pour mener à bien le projet ..... 56

**Tableau IV-1 :** Données intégrées à l'entrepôt ..... 75

**Tableau IV-2 :** Fichier XML simplifié de description du contenu de l'entrepôt ..... 78

**Tableau IV-3 :** Calculs effectués pour la génération du diagramme circulaire ..... 100

**Tableau IV-4 :** Description des principales options d'HyperAtlas[3] ..... 112

**Tableau V-1 :** Nombre de classes Java par modules ..... 118

# Conventions d'écriture, sigles et acronymes

Les expressions en langues étrangères sont indiquées en italique.

Afin de permettre une compréhension plus aisée, certains termes courants du jargon informatique sont utilisés sans traduction française, c'est le cas, par exemple, du mot « Web ».

Les termes suivis d'une étoile (exemple : géomatique*) sont définis dans le glossaire.

Tous les sites Internet cités dans les notes de bas de pages ont été consultés entre octobre 2011 et juillet 2012.

Certains néologismes pourront être utilisés car faisant partie du langage quotidien du laboratoire, comme par exemple, le mot « géodécisionnel » signifiant le décisionnel appliqué à la géographie.

Ci-dessous, vous trouverez une liste alphabétique des sigles et acronymes utilisés dans ce document :

**AEE** Agence Européenne pour l'Environnement ;
**ANSI** *American National Standards Institute* ;
**ANT** Another Neat Tool ;
**API** *Application Programming Interface* ;
**APT** *Almost Plain Text* ;
**ASCII** *American Standard Code for Information Interchange* ;
**ATM** *Analyse Territoriale Multiscalaire* ;
**BI** *Business Intelligence* ;
**CCR** Centre Commun de Recherche ;
**CI** *Continous Integration* ;
**CLC** *Corine Land Cover* ;
**CNAM** Conservatoire National des Arts et Métiers ;
**CNRS** Centre National de la Recherche Scientifique ;
**CSS** *Cascading Style Sheet* ;
**DOLAP** *Desktop On-Line Analytical Processing* ;
**EAR** *Enterprise Archive* ;
**EJB** *Enterprise JavaBean* ;
**ESPON** *European Observation Network for Territorial Development and Cohesion* ;
**ETL** *Extract Transform Load* ;
**GML** *Geography Markup Language* ;
**GPS** *Global Positioning System* ;

**GWT** *Google Web Toolkit* ;
**HOLAP** *Hybrid On-Line Analytical Processing* ;
**HTML** *HyperText Markup Language* ;
**HTTP** *HyperText Transfer Protocol* ;
**IA** Intelligence Artificielle ;
**IGN** Institut Géographique National ;
**IHM** Interface Homme-Machine ;
**INP** Institut National Polytechnique ;
**INRIA** Institut National de la Recherche en Informatique et Automatisme ;
**INSPIRE** *Infrastructure for Spatial Information in the European Community* ;
**ISO** *International Organization for Standardization* ;
**JAR** *Java Archive* ;
**JDBC** *Java DataBase Connectivity* ;
**JEE** *Java Enterprise Edition* ;
**JMS** *Java Message Service* ;
**JNDI** *Java Naming and Directory Interface* ;
**JSF** *Java Server Faces* ;
**JSP** *Java Server Page* ;
**JSR** *Java Specification Request* ;
**LIG** Laboratoire d'Informatique de Grenoble ;
**LUZ** *Large Urban Zones* ;
**M2M** *Machine to Machine* ;
**M4D** *Multi Dimensional Database Design and Development* ;
**MBA** *Master of Business Administration* ;
**MDX** *MultiDimensional eXpression* ;
**MESCAL** *Middleware Efficiency Scalable* ;
**MIME** *Multipurpose Internet Mail Extensions* ;
**MOLAP** *Multidimensional On-Line Analytical Processing* ;
**MTOM** *Message Transmission Optimization Mechnism* ;
**MVC** Modèle Vue Contrôleur ;
**NUTS** Nomenclature d'Unités Territoriales Statistiques ;
**OGC** *Open Geospatial Consortium* ;
**OLAP** *On-Line Analytical Processing* ;
**OLTP** *On-Line Transaction Processing* ;
**PARIS** Pour l'Avancement des Recherches en Interaction Spatiale ;
**PIB** Produit Intérieur Brut ;
**POM** *Project Object Model* ;
**RAD** *Rapid Application Development* ;
**REST** *Representational State Transfer* ;
**RIA** *Rich Internet Application* ;
**RIATE** Réseau Interdisciplinaire pour l'Aménagement du Territoire ;
**RMI** *Remote Method Invocation* ;

**ROA** *Resource Oriented Architecture* ;
**ROLAP** *Relational On-Line Analytical Processing* ;
**SDI** *Spatial Data Infrastructure* ;
**SFS** *Simple Features Specifications* ;
**SGBD** Système de Gestion de Bases de Données ;
**SGBDR** Système de Gestion de Bases de Données Relationel ;
**SI** Système d'Information ;
**SIG** Système d'Information Géographique ;
**SLR** *Spatial Level Reference* ;
**SOA** *Service Oriented Service* ;
**SOAP** *Simple Object Access Protocol* ;
**SOLAP** *Spatial On-Line Analytical Processing* ;
**SQL** *Structured Query Language* ;
**SRS** *Spatial Reference System* ;
**STEAMER** *Spatio-TEmporal information, Adaptability, Multimedia and KnowlEdge Representation* ;
**SVN** *SubVersion* ;
**TDD** *Test Driven Development* ;
**TEST** Travail d'Étude et de Synthèse Technique (UE CNAM ENG111) ;
**UAB** *Universitat Autònoma de Barcelona* ;
**UML** *Unified Modeling Language* ;
**UMR** Unité Mixte de Recherche ;
**UMS** Unité Mixte de Service ;
**URL** *Uniform Resource Locator* ;
**WAR** *Web Archive* ;
**WKT** *Well-Known Text* ;
**WMS** *Web Map Service* ;
**WS** *Web Service* ;
**WS-I** *Web Service Interoperability* ;
**WSDL** *Web Services Description Language* ;
**XML** *eXtended Markup Language* ;
**XMLA** *XML for Analysis* ;
**XOLAP** *XML On-Line Analytical Processing* ;
**XP** *eXtreme Programming*.

# Introduction

À la suite de l'avènement de l'Internet et des réseaux en général, les échanges d'informations se sont démocratisés. La conséquence en a été un accroissement significatif du volume des données accessibles, corollaire toujours d'actualité de nos jours. Nos sociétés et organismes publics croulent sous l'information et, faute de pouvoir la traiter efficacement, passent à côté d'opportunités ou pire, prennent de mauvaises décisions, ce qui peut avoir un impact important sur leurs performances.

Depuis le milieu des années 90, l'informatique décisionnelle offre des solutions pour la mise en valeur de ces données, dans le but de les transformer en une mine d'informations. Malgré tout, la dimension spatiale est restée très longtemps sous-exploitée, faute de moyens pour la traiter. Pour en profiter, l'informatique décisionnelle s'est adaptée et offre désormais un outil sur mesure, le traitement spatial et analytique en ligne, plus largement connu sous le sigle SOLAP ou *Spatial OLAP*. Cette technologie assez récente se positionne à l'intersection des systèmes d'informations géographiques et des solutions d'aide à la décision. Le mariage de ces deux mondes, ajouté aux capacités de la géomatique*, discipline se situant au croisement de l'informatique et de la géographie, permettrait une représentation cartographique de résultats issus de l'hypercube. De confidentiel, le sujet est alors devenu hautement stratégique. Il apporte un avantage concurrentiel incontestable aux entreprises le mettant à profit, s'offrant, par exemple, un outil d'étude de leur zone de chalandise. Il aide aussi les organismes d'états : les services de veille sanitaire dans le suivi de progression de maladies ou ceux responsables des voies de transport par l'étude des localisations des populations. Tout ceci est rendu possible par le croisement de nombreuses données, provenant de systèmes hétérogènes, possédant une perspective spatiale.

Ce mémoire a pour objectif la mise en place d'une partie de l'infrastructure de *Business Intelligence* capable d'accueillir au moins une dimension spatiale. Un outil de visualisation doit être pensé puis développé pour être à même de manipuler et de représenter les données issues de ce système.

# Partie I - Présentation

**Cette première partie introduit le cadre de notre stage d'ingénieur en présentant d'abord le contexte dans lequel il est effectué, les contraintes auxquelles il est soumis et les domaines dans lesquels il s'inscrit.**

## *A. Contexte*

Notre projet, effectué au laboratoire d'informatique de Grenoble (LIG), au sein de l'équipe STEAMER, s'insère dans une démarche expérimentale en marge des développements commandités par les traditionnels clients de l'équipe. Cependant, il profite de l'expérience et des connaissances acquises des précédentes réalisations. Nous présentons ici cet environnement.

### *1. Groupe de recherche HyperCarte*

Créé en 1996, le groupe de recherche « HyperCarte » a pour objectif de mettre au point des outils interactifs de production, de représentation et d'interrogation cartographique de phénomènes socio-économiques, environnementaux, épidémiologiques, sans que la liste soit exhaustive, et ce, sur différents territoires. En résultent la conception et le développement de modules d'analyse cartographique fondamentaux, formant le socle d'une plateforme logicielle accessible à des profils d'utilisateurs aux niveaux géographiques et statistiques hétérogènes.

Le concept « HyperCarte » repose sur l'hypothèse centrale que « toute spatialisation d'un phénomène social peut faire l'objet d'un nombre infini de représentations » [IMAG, 2011], les représentations en résultant dépendant de la nature des phénomènes représentés, des hypothèses du concepteur de la carte, des attentes et du contexte des utilisateurs finaux de l'information cartographique [Andrienko, 1999].

Ce groupement de recherche est constitué de géographes, de statisticiens et d'informaticiens, réunis en quatre équipes autour du thème de l'analyse territoriale et spatiale multiscalaire*.

**RIATE (UMS 2414)**

Le réseau interdisciplinaire pour l'aménagement du territoire (RIATE) a pour rôle de diffuser les bases de données territoriales européennes auprès des équipes de recherche françaises. Elle est le point focal français du projet ESPON (*European Observation Network for Territorial Development and Cohesion*). Celui-ci sera le sujet de la section 2 (page 5).

**Géographie-cités (CNRS UMR 8504)**

Au sein de Géographie-cités, l'équipe PARIS, acronyme de « Pour l'Avancement des Recherches en Interaction Spatiale », apporte ses connaissances dans les domaines de la géographie et des sciences sociales.

## LIG STEAMER (UMR 5217)

Le laboratoire d'informatique de Grenoble, créé en 2007, est un groupement d'équipes de recherche. Il est sous l'égide du Centre National de la Recherche Scientifique (CNRS), Institut National Polytechnique (INP) de Grenoble, de l'Institut National de Recherche en Informatique et Automatisme (INRIA), des universités Joseph Fourier et Pierre Mendès France.

L'équipe STEAMER[1] est l'une des 32 équipes du LIG. Ses recherches portent sur les systèmes d'information en ligne, ayant pour but le traitement d'informations multimédia à références spatiales et temporelles. Les travaux de recherche sont validés par le développement d'applications, dont la conception repose sur des formalismes structurés (ontologie, objet) ou semi-structurés (XML). Le champ d'application des travaux est dans le domaine de la géomatique.

## LIG MESCAL (UMR 5132)

À l'instar de STEAMER, l'équipe « *Middleware Efficienly SCALable* » (MESCAL) est partie intégrante du LIG. Elle gère des architectures distribuées à grande échelle, permettant la répartition et la parallélisation de calculs.

## Réalisations d'HyperCarte

Parmi les réalisations du groupe HyperCarte, les logiciels HyperAtlas et HyperAdmin peuvent être retenus de par leur importance. Les résultats des travaux et développements produits au sein de ces équipes sont décrits ci-après.

## HyperAtlas

HyperAtlas est la concrétisation de quelques idées du groupe de recherche « HyperCarte ». Les développements ont été menés par des stages mémoires CNAM successifs, dont les derniers sont : Christine Plumejeaud [Plumejeaud, 2007], Christophe Chabert [Chabert, 2007], Raphaël Thomas [Thomas, 2008], Benoit Le Rubrus [Le Rubrus, 2011] et Laurent Poulenard [Poulenard, 2011].

HyperAtlas est un module d'analyse territoriale multiscalaire (ATM). Il permet la manipulation des données, notamment socio-économiques, et leur visualisation sur différentes cartes. Il s'agit d'un outil polymorphe. Il peut, en effet, être décliné de plusieurs façons, selon l'application qui doit en être faite :

- Application de bureau (*standalone* [Le Rubrus, 2011]) ;
- Application manipulable en ligne (applet et *webcontainer* [Le Rubrus, 2011]) ;
- Service Web[2] *Web Map Service* (WMS) [Poulenard, 2011].

HyperAtlas (en version 2 depuis mars 2011) peut, en outre, être personnalisé, tant au niveau du style et des données, que du comportement par défaut. Ainsi, il existe notamment deux versions de l'application : « standard » et « ESPON », le second embarquant un habillage aux couleurs du drapeau européen ainsi qu'une configuration par défaut résolument tournée vers l'Europe des 28*.

[1] STEAMER : *Spatio-TEmporal information, Adaptability, Multimedia and KnowlEdge Representation.*

[2] L'expression « service Web » sera utilisée régulièrement le long de ce mémoire, au détriment de la forme française, plus lourde, « service de la toile ».

### HyperAdmin

Ce logiciel est le module d'administration fournissant les fichiers `.hyp`, jeux de données nécessaires à HyperAtlas. Ils concentrent les informations de localisation, de structure et les indicateurs socio-économiques géoréférencés (stocks). HyperAdmin traite les données avec un « emboîtement hiérarchique strict » : pays, grandes régions, régions, départements, etc. comme l'illustre la figure I-1.

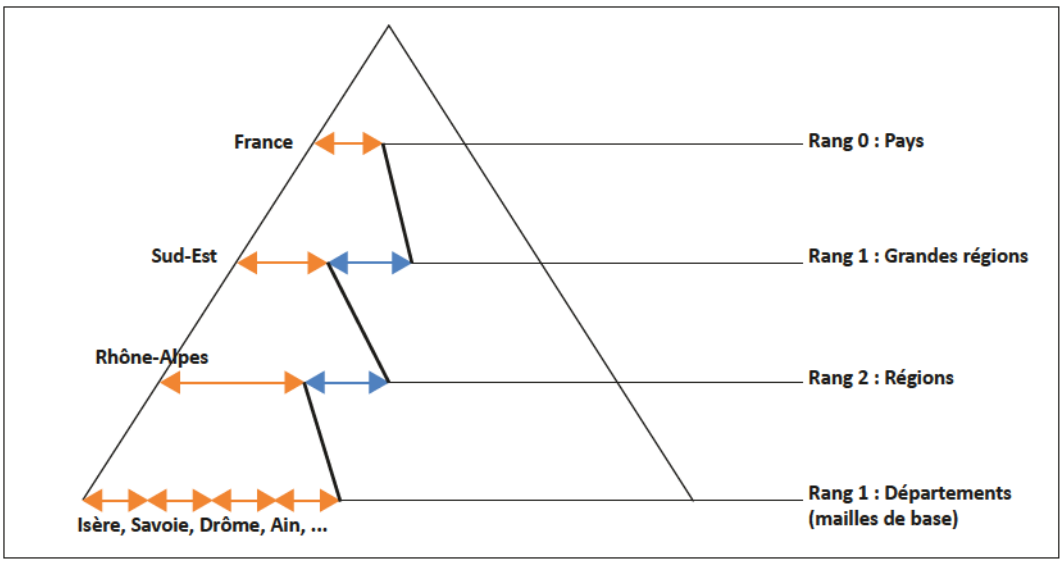



Figure I-1 : Découpage hiérarchique des données [IMAG, 2011]

### HyperSmooth

Le module HyperSmooth propose une représentation continue des phénomènes spatiaux, s'affranchissant ainsi des limites administratives. Il repose sur une estimation probabiliste (dite « de potentiel ») basée sur l'échantillonnage territoriale de données. Ce module n'est plus maintenu depuis 2007, il n'en sera pas plus fait état dans ce mémoire.

### Hyantes

Il s'agit d'une bibliothèque, au sens informatique du terme, proposant des méthodes de lissage liées à l'estimation de densité multiscalaire par voisinage. Ce module a été développé par l'équipe MESCAL.

## *2. Programme ESPON 2013*

ESPON (*European Observation Network for Territorial Development and Cohesion*) est un réseau d'étude destiné à l'observation de l'espace communautaire européen. Soutenu et financé par les États membres de l'Union Européenne, ainsi que par des États partenaires (la Norvège, la Suisse, l'Islande et le Lichtenstein), le programme éponyme, version 2013, a pour mission le support à l'élaboration d'une politique de cohésion territoriale et au développement harmonieux du territoire européen. Pour atteindre ces objectifs, des données comparables, des faits, des analyses et des scénarios sur les dynamiques territoriales sont mises à disposition. Il en va de même pour toutes les informations révélant le potentiel de développement des régions et des grands territoires contribuant à la compétitivité européenne, à la coopération territoriale et au développement durable et équilibré.

C'est dans ce cadre que l'équipe STEAMER intervient, en particulier sur les projets *ESPON 2013 Database* et *ESPON HyperAtlas Update*.

### ESPON 2013 Database

ESPON considère la base de données comme « l'élément central de la plateforme scientifique [...] et comme l'outil fournissant les entrées pour les analyses fondées sur des indicateurs territoriaux » [ESPON, 2012]. Dans ce contexte, il est aisé de comprendre combien il est crucial de maintenir et développer cette base de données, dont le nombre de contributeurs n'a cessé de croitre depuis sa création. Elle a pour objectif de stocker toutes les données des projets ESPON.

Le développement de la phase 2 débuté en 2011, sous le nom *Multi Dimensional Database Design and Development* (M4D), prévoit, pour ce qui est des données, l'inclusion de nouvelles sources notamment en provenance d'EUROSTAT et de l'Agence Européenne pour l'Environnement (AEE). Les objectifs de cette gestion de nouvelles informations permettront de replacer et de situer l'Europe par rapport à ses voisins, ainsi que dans le monde ; elles aideront à maîtriser au mieux l'émergence d'un nouveau paradigme en matière énergétique et de ses impacts sur l'environnement.

Divers types d'objets géographiques doivent être pris en compte. En effet, jusqu'à présent, le découpage hiérarchique de référence était le NUTS (Nomenclature d'Unités Territoriales Statistiques), comme il a été brièvement décrit dans la section HyperAdmin, basé sur les divisions administratives. Il serait alors question d'ajouter de nouveaux découpages.

Au niveau technologique, la base de données doit être capable de stocker et gérer une quantité toujours plus grande d'informations, tout en en assurant la qualité. Ces données doivent être accessibles facilement, par le biais d'une interface légère en ligne, ainsi que selon les recommandations d'INSPIRE (*Infrastructure for Spatial Information in the European Community*), plus longuement évoquées dans la section suivante.

Parce qu'il s'agit de projets mûrs, les différents livrables se doivent d'être pleinement fonctionnels.

### ESPON HyperAtlas Update

La version livrée en avril 2011, dont les évolutions ont été développées par Benoit Le Rubrus [Le Rubrus, 2011], a permis de voir l'ajout de nouvelles fonctionnalités avancées telles que la boîte à moustaches, les courbes de Lorenz, le coefficient de Gini, l'index de Hoover et les cartes de redistributions. Cette mise à jour a impacté à la fois le programme utilisateur HyperAtlas par l'ajout des écrans de visualisation de ces fonctionnalités, et le logiciel de génération des fichiers `.hyp` HyperAdmin par l'adjonction d'une matrice de distances entre les villes, nécessaire aux calculs de proximité, par exemple. Cette partie du programme a été honorée.

Notre projet s'inscrit, à terme, au carrefour des deux projets, utilisant à la fois les données en provenance de la base de données ESPON et profitant du cœur de métier d'HyperAtlas, à savoir la production de cartes.

## 3. Directive INSPIRE

La directive INSPIRE[3] (directive européenne 2007/2/CE du 14 mars 2007, ordonnance 2010-1232, ratifiée en France par la loi 2011-12 du 5 janvier 2011) vise à favoriser l'échange de données numériques, environnementales, géographiquement localisées, au sein de la communauté européenne, en établissant une infrastructure d'information géographique (dont l'acronyme anglo-

[3] Directive INSPIRE complète disponible à cette adresse :
http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2007:108:0001:0014:FR:PDF

saxon est SDI *Spatial Data Infrastructure*). Elle a pour mission générale de favoriser la connaissance, l'analyse et le suivi des territoires, en particulier dans le domaine du développement durable et de l'environnement. D'après [Leobet & Merrien, 2011], les objectifs sont de :

- Faciliter la prise de décision dans un cadre démocratique ;
- Décloisonner les informations entre les autorités publiques ;
- Permettre la mise en place de meilleurs services au citoyen ;
- Favoriser la croissance économique [...] au travers du développement non seulement du secteur de l'information géographique, mais aussi des nombreuses activités qui ont besoin d'utiliser des données géographiques pour créer de nouveaux services.

Concrètement, les autorités publiques sont tenues de recenser les données qu'elles détiennent, qui entrent dans le champ de la directive et d'établir les métadonnées sur ces informations selon les normes énoncées dans la directive. Enfin, une fois ces étapes validées, cinq services doivent être édités afin de permettre leur exploitation sur Internet :

- Services de recherche : permettre l'identification des séries et services de données géographiques sur la base des métadonnées correspondantes ;
- Services de consultation : visualiser les données, en autoriser la superposition avec d'autres sources, proposer des outils afin de donner la possibilité de se déplacer, de changer d'échelle, d'afficher les légendes, etc. ;
- Services de téléchargement : récupérer les données elles-mêmes, pour un traitement tiers ;
- Services de transformation : transformer des séries de données géographiques en vue d'une interopérabilité maximale ;
- Services d'appel : gérer l'appel de services pour leur réutilisation dans des applications en ligne.

Cette directive implique une profonde refonte de certains systèmes d'informations des autorités publiques concernées, notamment rendre les données et les métadonnées accessibles au public par leur publication sur Internet tout en permettant de les croiser entre elles.

Notre projet peut s'inscrire dans le cadre de cette directive dans le sens où il transforme et stocke des données, cependant le domaine de l'informatique décisionnelle en reste un peu en marge. En effet, cette dernière ne fait que réutiliser les données précédemment partagées.

## *4. Informatique décisionnelle*

Dans un monde où l'information est abondante, voire envahissante, noyant les décideurs sous un flux inutilement conséquent, l'informatique décisionnelle apporte une solution à sa synthèse. Elle désigne en effet un ensemble de techniques et d'outils informatiques utilisés dans l'objectif d'aider la prise de décisions. Ces outils, mis bout à bout, forment une chaîne décisionnelle, afin de gérer un flot de données, de leur source, jusqu'à leur restitution sous forme analysable, notamment par le biais de tableaux de synthèse et de graphiques.

Apparue dans les années 90, cette discipline connait un essor considérable depuis la démocratisation d'Internet et le partage d'informations à grande échelle, la généralisation de la communication entre serveurs et l'arrivée à maturité d'algorithmes issus du domaine de l'intelligence artificielle. Un environnement est alors pensé et construit afin que les décideurs, principaux utilisateurs de ces solutions, puissent manipuler eux-mêmes les outils mis à leur disposition. Ils contrôlent ainsi les entrées et sorties des rapports générés et peuvent, en toute autonomie, générer ceux qui les guideront dans leurs décisions.

Reposant sur une imbrication de logiciels, ces technologies manipulent de grandes quantités de données, tant au niveau physique par la lecture sur disque au travers du système de gestion de bases de données, que logique, par les calculs en mémoire. Cette caractéristique implique que les données constituant le cœur du système décisionnel soient organisées selon un modèle spécifique, et que les différents composants de cet environnement soient décorrélés des systèmes opérationnels, sous peine d'en consommer toutes les ressources et d'en dégrader les performances.

Enfin, pour exploiter aisément ces technologies, une interface doit permettre l'accès aux données collectées et en proposer une représentation.

### *5. Visualisation des données*

Pour permettre leur lecture et leur interprétation, les données doivent être présentées et mises en forme. Selon leur type et le message qui doit être délivré, plusieurs solutions sont envisageables : tableau, graphique et carte. Si les deux premiers semblent naturels dans le paysage des logiciels actuels, le dernier est toutefois beaucoup plus rare. La raison provient du fait que les données géo-référencées ont longtemps été sous-exploitées. Désormais, la tendance s'inverse et de plus en plus d'applications mettent à disposition une représentation cartographique de ces informations.

Les tableaux croisés sont généralement le moyen le plus classique pour la représentation des données du cube. Ils permettent une vision synthétique des dimensions entrant en jeu et une définition rapide des membres qui forment un résultat. Les diagrammes sont largement utilisés aux travers des systèmes d'information, et ce, depuis très longtemps. Il en existe de nombreuses sortes : à barres, linéaires, circulaires, etc. Tous ont pour point commun d'offrir une lecture simplifiée des informations tabulaires en en dégageant rapidement les tendances et/ou valeurs exceptionnelles, selon le but désiré. Enfin, les cartes exploitent les données spatialement localisées dans l'objectif de les resituer sur un planisphère ou les unes par rapport aux autres. Les multiples représentations (cartes à symboles proportionnels, choroplèthes, etc.) proposent des points de vue adaptés aux requêtes. Elles sont utilisées pour faire émerger la répartition géographique des informations, aussi bien de manière unitaire, que dans leurs ratios.

## *B. Enjeux et objectifs du projet*

La technologie SOLAP, émergente, permet selon Yvan Bédard une manipulation performante de données géographiquement localisées, en mettant en jeu les principes de l'informatique décisionnelle [Bédard *et al.*, 2005]. Il s'agit donc d'une mise en exergue de ces informations afin d'en tirer le maximum : en l'extrapolant, en cherchant des correspondances entre les données, etc.

En analysant le titre de notre sujet, nous pouvons dégager deux objectifs principaux à ce projet : l'« hypercube SOLAP » et « l'outil de visualisation ».

La première partie implique une application dont les tâches suivantes peuvent former une feuille de route générale :

- La construction et l'alimentation de l'entrepôt de données ;
- Le choix puis la mise en place d'une architecture *Spatial OLAP* ;
- La modélisation de l'hypercube qui abritera, à plus long terme, les données extraites de la base de données ESPON DB, conforme à la directive INSPIRE ;
- L'ouverture des données via une exposition de services mis à disposition sur un réseau (services Web sur Internet, par exemple).

La seconde met en jeu le client du « cube », comme moyen :

- D'interroger le serveur ;
- De restituer les informations que ce dernier a transmis, quelles qu'elles soient ;
- D'offrir à l'utilisateur des représentations de ces données conformes à ses attentes.

Les outils utilisés et développés doivent permettre les opérations classiques du traitement analytique en ligne, décrits dans la section consacrée à l'informatique géodécisionnelle, ainsi que la présentation des données, sous plusieurs formes, dont cartographique.

## *C. Plan du mémoire*

Outre cette partie introductive situant le contexte et les domaines relatifs à notre projet, ce mémoire s'articule autour de quatre autres parties.

La partie II dresse un état de l'art des deux principaux axes impliqués dans la création d'un environnement géodécisionnel. Il est d'abord question des principes fonctionnels de l'informatique décisionnelle, puis des standards régissant la visualisation de l'information en général et des données spatiales en particulier.

La partie III présente la proposition que nous avons rédigée afin d'atteindre les objectifs du sujet et répondre aux besoins exprimés. Nous y abordons l'architecture retenue pour l'application ainsi que le découpage en grandes phases de notre projet. Les décisions constituant le socle de nos futurs développements sont détaillées et justifiées.

La partie IV décrit le travail effectué pour mener à bien le projet. Elle est basée sur la proposition avancée dans la partie précédente. Elle est elle-même composée de quatre chapitres correspondant respectivement à la présentation des outils utilisés, à la mise en place de la chaîne décisionnelle, aux développements relatifs au serveur puis au client formant notre logiciel.

La partie V conclut ce mémoire par une synthèse technique des réalisations élaborées au cours de la partie précédente, propose une ouverture sur les possibilités offertes par notre prototype et ses possibilités d'amélioration. Enfin, un bilan personnel clôturera ce mémoire.

# Partie II - Etat de l'art

**La géomatique est une science reconnue depuis 1994[4]. Avec la généralisation de l'utilisation de l'informatique, la production cartographique n'a pas échappé à son automatisation. En plus de représentations cartographiques d'éléments et phénomènes naturels, elle est capable de montrer des infrastructures et des activités économiques, humaines et sociales.**

L'informatique décisionnelle apporte son lot de réponses et d'outils à la géomatique en déclinant le traitement analytique en ligne (dont l'acronyme anglo-saxon est OLAP pour *On-Line Analytical Processing*) en version spatiale : le néologisme « géodécisionnel », largement employé dans ce mémoire, est né en même temps que le *Spatial OLAP* (SOLAP). Cependant, elle soulève aussi de nombreuses questions que nous allons tenter de résoudre.

Cet état de l'art présente tout d'abord le fonctionnement de cette informatique géodécisionnelle, dont les outils SOLAP. Ce chapitre se place dans la continuité de notre travail de synthèse, effectué dans le cadre de l'unité « Information et communication pour l'ingénieur » du CNAM [Tranchant, 2011]. Ce domaine est très vaste et toujours en pleine expansion afin de couvrir les besoins en sans cesse évolution des utilisateurs. Notre périmètre d'étude couvre des bases théoriques aux applications usuelles. Nous évoquons aussi brièvement quelques applications alternatives, notamment dans la partie OLAP.

Un statut sur les concepts de visualisation de l'information, aussi bien en vue tabulaire que cartographique, constitue notre second axe. En effet, récupérer de l'information n'est pas une fin en soi, mais un élément constituant d'une démarche plus vaste, dont la visualisation des résultats peut être l'un des objectifs les plus évidents. Pour compléter cette partie sur la restitution des données, notre état de l'art passe en revue les méthodes et usages tournés vers la production de cartes. Pour dégager ces us et illustrer nos propos, quelques solutions sont étudiées.

Une synthèse conclura cette partie.

## *A. Informatique géodécisionnelle*

Au croisement de l'informatique décisionnelle et des systèmes d'informations géographiques (SIG), l'informatique géodécisionnelle émerge peu à peu comme une discipline à part entière. Ce chapitre dresse un état scientifique et technique de ce domaine tout en décrivant et expliquant les notions qui lui sont relatives ou voisines.

### *1. Principes de l'informatique décisionnelle*

L'informatique décisionnelle, aussi nommée *Business Intelligence* (BI), répond aux problématiques posées par le croisement de données hétérogènes, destructurées et éparses rencontrées dans les différents systèmes d'informations (SI) des entreprises. Or, l'une des plus grandes richesses

[4] Journal Officiel du 14 février 1994.

d'une société étant justement ces informations, le fait de pouvoir les synthétiser afin de dégager les grandes tendances d'évolution de son métier, d'en tenter une prédiction afin de répondre au mieux aux demandes du marché qu'elle occupe ou qu'elle vise donne un avantage concurrentiel important à toute entreprise ayant mis en place ce genre de structure. S'adressant principalement aux dirigeants, elle n'a d'autre finalité que de délivrer des informations pertinentes afin de leur permettre une prise de décision efficace.

Cette discipline a connu un essor particulier dans les années 90, avec la création d'environnements spécialisés dans l'aide à la décision, ainsi qu'avec l'arrivée à maturité des algorithmes permettant l'extraction d'informations à partir de données brutes. Ces algorithmes proviennent des mondes de la statistique et de la recherche dans le domaine de l'intelligence artificielle (IA).

Lorsqu'une entité décide la mise en place d'une telle architecture, les outils technologiques qui font partie de la chaîne décisionnelle sont déployés en accord avec la stratégie globale de l'entreprise. Le choix des logiciels, les données et leurs sources sont autant de variables d'ajustement pouvant influer sur la qualité des réponses données.

En outre, cet environnement décisionnel doit répondre à plusieurs exigences :

- Simplicité : l'outil doit être simple et ergonomique, aucune connaissance en informatique ne doit être pré-requise ;
- Rapidité : les réponses aux requêtes doivent être quasiment instantanées ;
- Manipulation de gros volumes de données : pour sortir des tendances de progression, de vente sur cinq ans par exemple, les données de toutes les transactions doivent être agrégées, occasionnant le traitement de nombreuses données ;
- Indépendant du système de production : pour effectuer les requêtes, un système décisionnel dédié est interrogé et celui de production n'est pas impacté ;
- Sécurité : le système d'aide à la prise de décision ne doit généralement être accessible qu'aux dirigeants ;
- Fiable : confiance est le maître mot. En effet, il est impossible de prendre une décision si l'on ne peut donner un total crédit au système.

L'informatique décisionnelle permet la sélection des informations opérationnelles pertinentes pour l'entreprise. Elles sont collectées via des outils d'extraction, transformation et chargement, où elles sont normalisées pour alimenter un entrepôt de données. De ce concept est née la notion de modélisation dimensionnelle. Cette dernière est fondamentale pour honorer les exigences de rapidité, de facilité d'analyse et de manipulation.

## *2. Modélisation multidimensionnelle*

Afin de répondre à un besoin analytique, en sachant que le modèle relationnel et ses principes de normalisation y trouvent leurs limites (notamment en termes de performances d'accès), le modèle multidimensionnel offre une solution intuitive et complète aux exigences inhérentes à l'informatique décisionnelle.

Le modèle multidimensionnel est la combinaison de tables de dimensions et de faits. Le fait (aussi appelé indicateur) est le sujet de l'analyse et l'élément central de la structure. Il est formé de mesures, généralement numériques, renseignées de manière continue. Ces mesures permettent de résumer un grand nombre d'enregistrements des données sources en quelques-uns. Le fait est analysé selon des perspectives, nommées dimensions. Celles-ci, par leurs croisements, donnent le contexte de la valeur du fait. Chacune contient une structure hiérarchique, dont voici un exemple :

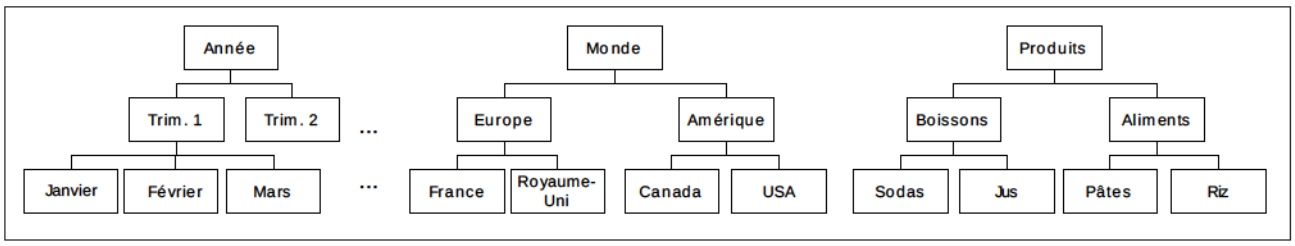



Figure II-1 : Dimensions et leur hiérarchie respective (Source: [Dinimant, 2009])

La figure II-1 présente trois dimensions (temporelle, spatiale, produit), chacune ayant sa propre hiérarchie :

- Temporelle : les années sont découpées en trimestres, eux-mêmes découpés en mois ;
- Spatiale : le monde est divisé en continents, subdivisés en pays ;
- Produit : les produits sont scindés en catégories, puis en sous-catégories.

Comme nous pouvons le voir, les dimensions contiennent des membres, hiérarchisés en niveaux. En reprenant un exemple complet sur la dimension temporelle uniquement, nous obtenons la figure suivante :

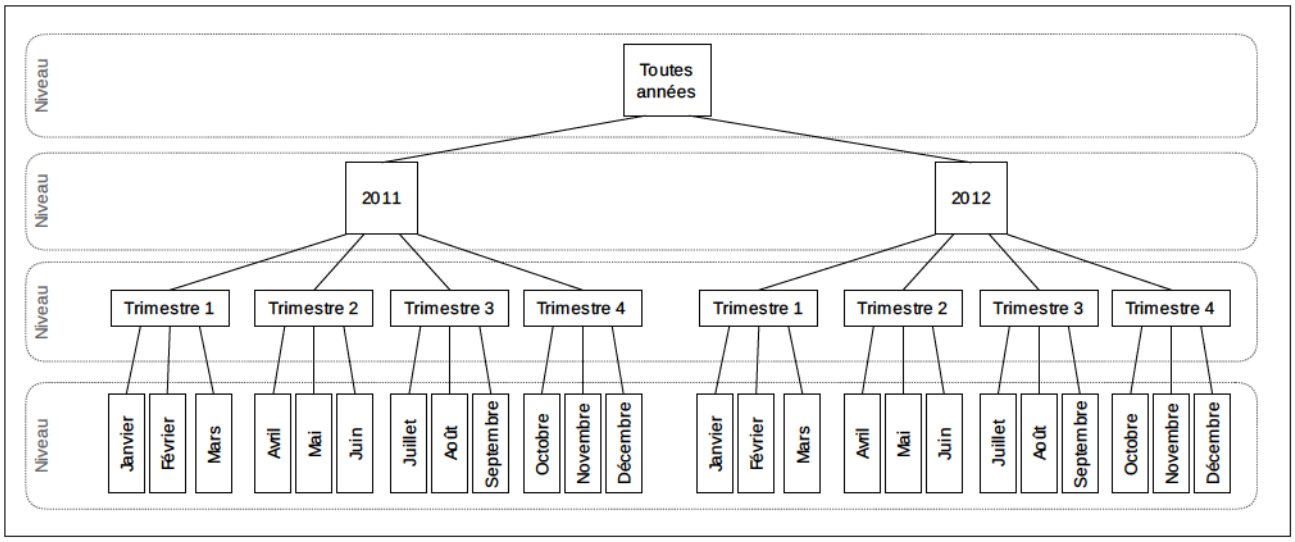



Figure II-2 : Niveaux et membres de la dimension temporelle

Tous les rectangles sont les membres de la dimension, divisés en niveaux. Si l'on prend comme référence la branche gauche de la figure II-2, les membres « Janvier », « Février » et « Mars » sont des enfants du membre « Trimestre 1 », lui-même enfant de « 2011 ».

Le niveau le plus bas de la hiérarchie impose la granularité maximale du cube. En effet, les données sont généralement agrégées au niveau le plus petit/précis de la structure multidimensionnelle. Toujours dans l'exemple illustré dans la figure II-2, la granularité de la dimension temporelle est le mois. Ceci n'exclut cependant pas que des données plus précises encore existent : nous évoquerons ce point dans la section *Hybrid OLAP* (HOLAP).

Certaines dimensions peuvent posséder plusieurs hiérarchies. Cette flexibilité permet d'organiser un même axe de différentes façons. Un exemple classique concerne l'axe temporel : un utilisateur peut être intéressé par la visualisation de l'année calendaire ou par celle de la période fiscale. La hiérarchie permet d'analyser efficacement et naturellement sous plusieurs angles le même type d'information.

Il y a quelques cas particuliers autour des dimensions. Les dimensions dégénérées sont référencées directement dans le fait : il n'y a plus de table de dimension mais des clés existent toujours dans la table de mesure, se suffisant à elles-mêmes [Becker, 2003]. Les dimensions de type « fourre-tout » (*junk*) contiennent, par définition, des informations diverses qui n'ont pas forcément une cohérence, mais qui peuvent constituer de bons filtres [Ross, 2003]. Les dimensions à évolution lente prennent en charge les changements dans le temps. Trois solutions s'offrent alors pour prendre en compte cette modification :

- L'écrasement de l'ancienne valeur : il s'agit de mettre à jour l'enregistrement. Ceci implique la perte de l'historique ;
- L'ajout d'un nouveau membre dans la dimension : les deux valeurs (l'ancienne et la nouvelle) coexistent, les clés sont couplées à un numéro de version ou à un marqueur indiquant s'il s'agit de la valeur la plus récente ;
- L'ajout d'un nouveau champ dans la dimension : permet l'enregistrement sur le même membre des valeurs actuelle et d'origine ou actuelle et précédente [Donsez, 2006], [Browning & Mundy, 2001].

Le recours à une mini-dimension à évolution lente peut être une quatrième solution. Il permet de déporter la problématique de versionnement à une autre table, liée à celle qui aurait été confrontée à l'évolution, mais occasionnant moins d'enregistrements. Cette dernière solution permet de gérer plus aisément ce qu'on nomme dimensions à évolution rapide, sans faire exploser le nombre d'enregistrements dans la table principale de la dimension.

Les dimensions s'articulent autour du fait afin d'offrir des perspectives d'analyses multiples, pouvant être sollicitées selon les besoins de l'analyse. Par exemple, un fait référencé par trois dimensions donnera l'impression d'interroger un cube, comme l'illustre la figure II-3 :

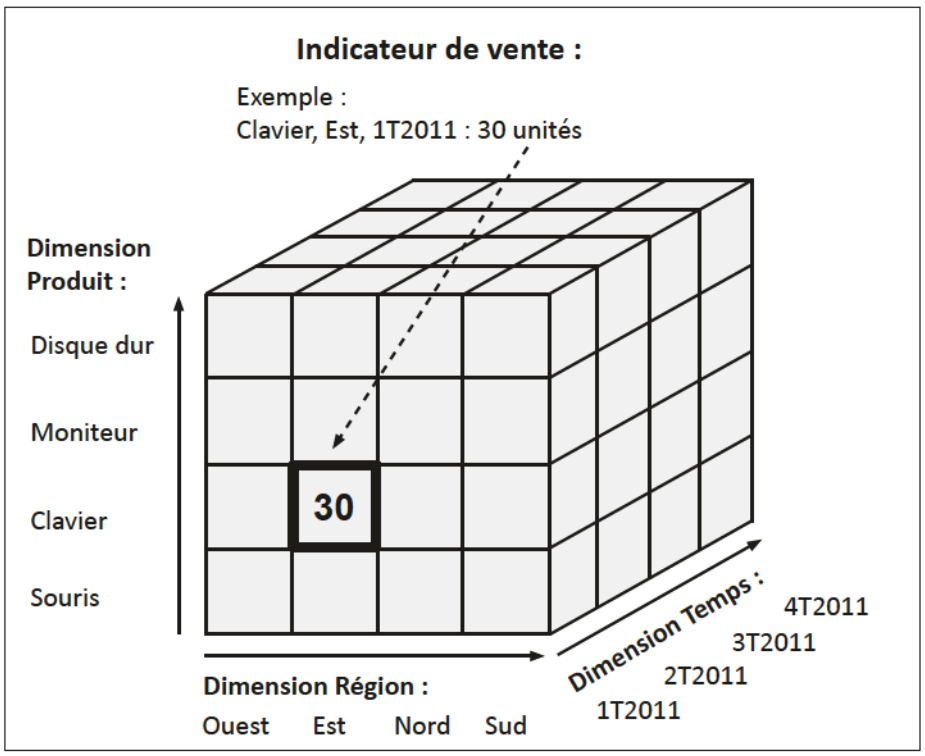



**Figure II-3 : Représentation multidimensionnelle : le cube offre trois perspectives d'analyse (Source : [Meier, 2006])**

Cette figure illustre la réponse (« 30 unités ») à l'interrogation « combien d'unités de clavier ont été vendues dans la région Est au cours du premier trimestre 2011 ? ».

Il peut y avoir de deux à *n* dimensions ; un modèle avec plus de trois dimensions étant nommé « hypercube » ou « *n*-cube ». Par souci de généralisation, on parle simplement de « cube ».

Ce mode de stockage multidimensionnel favorise un accès rapide à l'information, les fonctions (somme, moyenne, minimum, maximum, comptage, comptage distinctif) étant plus facilement réalisables lorsque les données sont normalisées et pré-calculées. Il existe en effet, deux règles pour la gestion des agrégats :

- Calcul à la volée : les fonctions sont exécutées lors de l'interrogation, sur la totalité des données sélectionnées ;
- Lecture et calcul sur les données pré-agrégées : lors du chargement des données dans le cube, la fonction est exécutée et son résultat est stocké à des niveaux hiérarchiques stratégiques. Il ne reste alors qu'à lire l'information existante, ou à exécuter la fonction sur ces mêmes données, dans le cas où la requête ne correspondrait pas tout à fait aux agrégats.

La figure II-4 illustre ce dernier mode de fonctionnement. Comme pour le calcul à la volée, les données détaillées sont bien entendu présentes. Les agrégats sont effectués à chaque niveau hiérarchique : mis en exergue par le gras sur le schéma, les sommes par trimestres, années, localisations (Europe, Monde) et par types (Boissons, etc.)

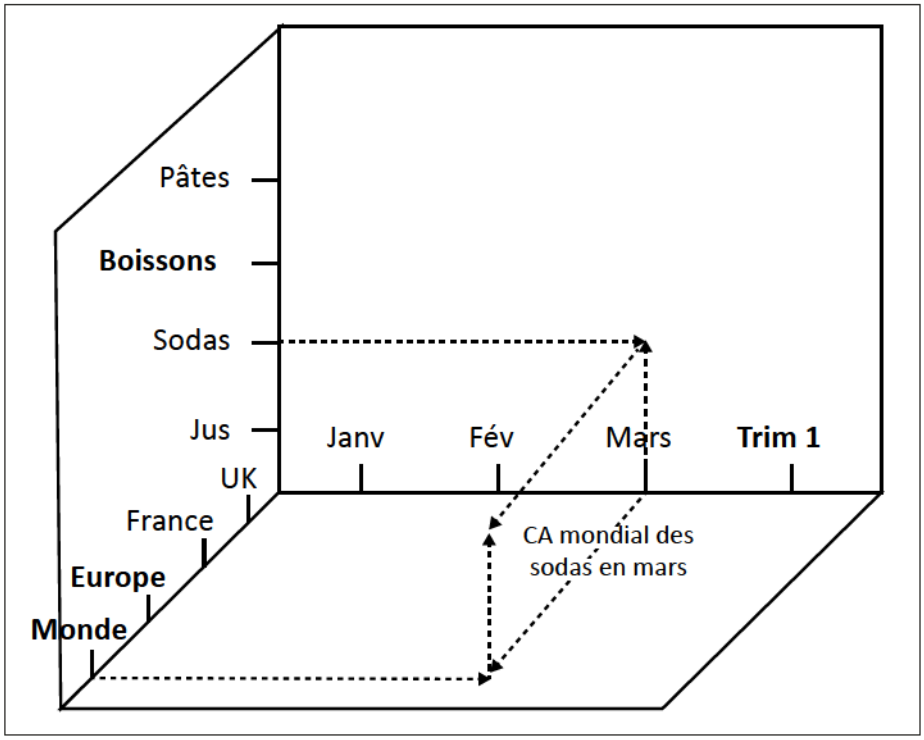



**Figure II-4 : Représentation en système de coordonnées d'un cube contenant les données agrégées (Source: [Dinimant , 2009])**

Un système complet a vu le jour en même temps que l'invention du concept d'informatique décisionnelle, avec ses principes, son architecture et son langage d'interrogation, MDX (*MultiDimentional eXpressions*). La section suivante aborde ces particularités.

## *3. Architecture de l'informatique géodécisionnelle*

L'architecture d'un système décisionnel est organisée comme une chaîne : localisation des données sources ; extraction, nettoyage, transformation et chargement dans un entrepôt ; interrogation des bases de données, mise sous forme multidimensionnelle ; et enfin, service à des outils d'analyse ou de fouille.

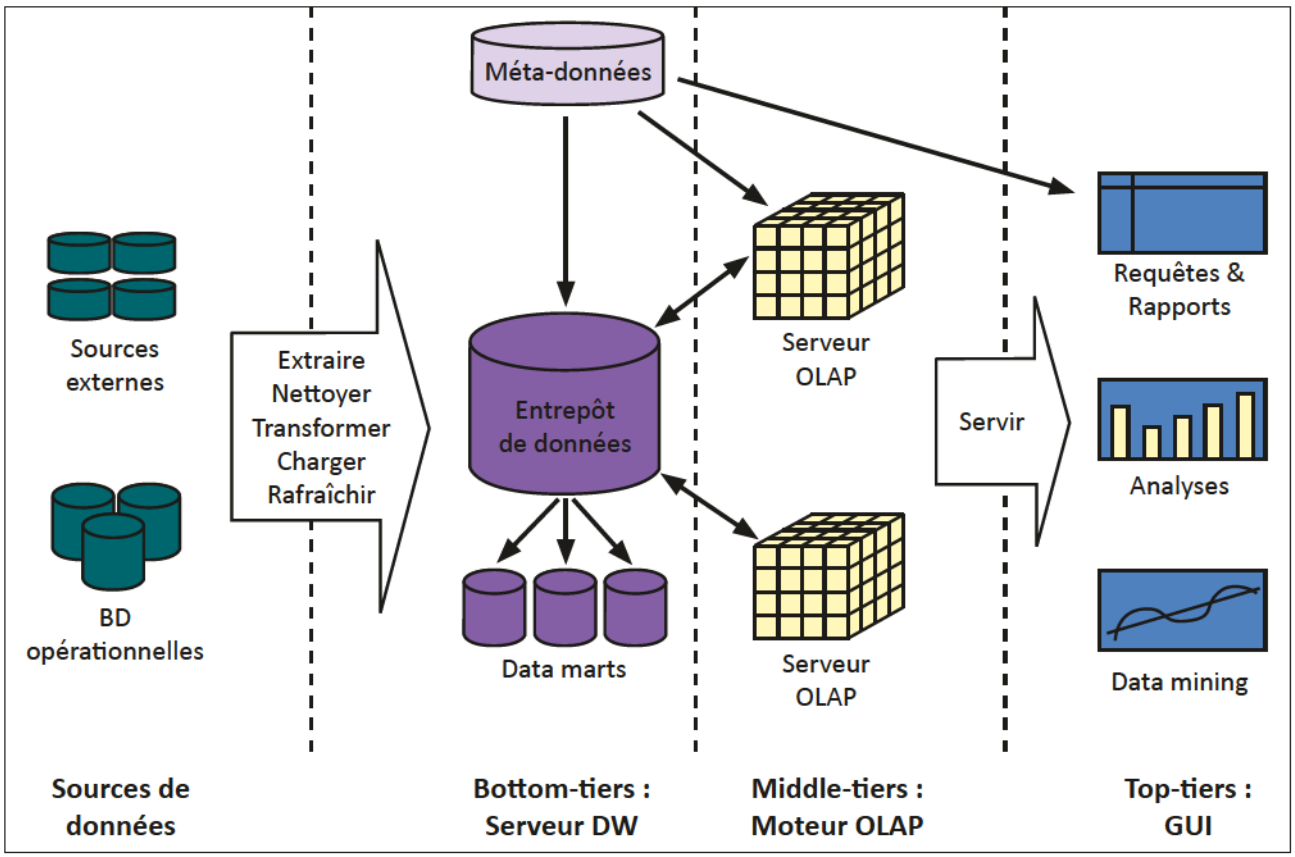



Figure II-5 : Ensemble des briques composant un système décisionnel (Source : [Lebrun & Charrier, 2008])

La figure II-5 ci-dessus illustre cette structure. Cette sous-section propose une analyse plus poussée des différents éléments constituant l'environnement décisionnel.

### *a. Sources de données*

Afin d'alimenter l'entrepôt et/ou les marchés de données, des informations doivent être collectées dans les divers systèmes où elles sont exploitées ou mises à disposition. Il s'agit généralement de données internes à l'organisme, issues des environnements transactionnels des applications opérationnelles. Les différents services d'une société utilisent plusieurs programmes, chacun ayant systématiquement un stockage de données (base de données relationnelle, fichier XML (*eXtended Markup Language*), format propriétaire, etc.). Hétérogènes, dispersées, difficiles à atteindre, ces informations sont cependant d'une importance capitale pour le système décisionnel, leur acquisition par les outils ETL est la source de nombreux aménagements dans le système d'information : changement d'encodage, ouverture de liens d'accès aux SGBD, adaptation du réseau et mise en place d'authentifications [Tranchant, 2011].

### *b. Outils d'extraction, transformation et chargement*

Désignés par le sigle anglo-saxon ETL (*Extract Transform Load*), ces outils sont la porte d'entrée des données et sont cruciaux pour s'assurer un entrepôt à jour. Ils extraient les données des systèmes composites sources, puis leur appliquent des transformations en vue de les rendre intégrables dans la structure multidimensionnelle.

Les processus couverts par le terme ETL sont complexes et doivent faire l'objet d'une attention particulière : il en va en effet de la crédibilité des données qui seront ensuite exploitées par l'OLAP et qui serviront de base aux décisions des analystes. Afin d'atteindre le but désiré en matière

d'organisation et de granularité des données, ainsi qu'au niveau des performances, de nombreuses opérations doivent être appliquées [Tranchant, 2011] :

- Filtrage : identifie les données aberrantes ;
- Dédoublonnage : empêche l'insertion en de multiples exemplaires de la même information, issue de plusieurs sources de données différentes ;
- Formatage : arrange les données afin qu'elles partagent toutes le même référentiel (même unité, même multiplicateur), transforme les informations codifiées (abréviations) pour les unifier, standardise les dates ;
- Dénormalisation : redonde l'information. Les sources étant généralement issues de bases relationnelles s'appuyant sur la troisième forme normale*, l'outil « aplanit » les relations entre entités afin de n'en garder que des données pertinentes (les clés primaires des SGBDR sont inutiles) ;
- Synchronisation : garantit la cohérence des agrégats. Les mêmes éléments peuvent avoir des représentations différentes selon leurs sources, il s'agit ici de les détecter et de les traiter en conséquence ;
- Désagrégation : transforme les données en changeant leur support. On peut ainsi passer un indicateur de haut niveau (une donnée d'une granularité faible) sur un niveau plus petit en le pondérant, grâce à d'autres informations. Déterminer le taux de pollution d'une commune à partir de celui d'un bassin versant, en le pondérant par la superficie de ladite commune, par exemple ;
- Agrégation : effectue une ou plusieurs opérations sur les données, parmi lesquelles on retrouve la somme, la somme cumulée, la moyenne, le comptage, le comptage distinct, etc.

Avec le traitement de plus en plus généralisé des données spatiales, les ETL se sont adaptés et une nouvelle catégorie est apparue. Il s'agit des ETL dits « spatiaux », qui gèrent nativement les géométries nécessaires aux outils SOLAP et ainsi permettent l'interrogation du cube par des opérateurs topologiques, par exemple.

### *c. Entrepôt de données*

Cœur du système décisionnel, l'entrepôt de données est « une base de données architecturée pour des requêtes et des analyses, plutôt que pour le traitement transactionnel des données », et les résultats de ces requêtes doivent être obtenus rapidement [Noirault, 2006], [Bédard *et al.*, 1998].

Un entrepôt de données doit être, selon [Inmon, 2005] :

- Orienté sujet : chaque zone (fait) de l'entrepôt est organisée pour répondre à une problématique métier, comme par exemple, les habitudes d'achat des clients, ou l'efficacité d'une équipe ;
- Intégré : toute entrée de l'entrepôt est uniforme. Les valeurs sont toutes exprimées dans la même unité : les centimètres et les pouces, par exemple, sont convertis dans une même unité, celle qui sera définie par l'architecte du cube. Ceci oblige à déporter la lourdeur des conversions à l'étape de l'ETL ;
- Historisé : chaque donnée est horodatée et ne sera jamais effacée ni modifiée (voir « non volatile »). Ceci permet à l'utilisateur du système de constater une progression, comprendre les tendances, etc. ;
- Non volatile : ici synonyme de « lecture seule ». Pouvoir modifier une donnée reviendrait à avoir la possibilité de modifier le passé, ce qui n'est jamais une bonne chose. Les données sont associées à un instant, valables uniquement à un moment donné.

L'entrepôt possède néanmoins plusieurs types de fonctionnement. En effet, selon le système de gestion de données sur lequel il est construit, on parle de différentes approches.

### Approche ROLAP

La première, et plus répandue, est l'approche *Relational OLAP* (ROLAP). Basée sur un SGBD relationnel, il pourrait s'agir d'un modèle entité/association classique, s'il n'y avait pas cet agencement particulier des tables. Ces dispositions sont décrites après la description des différentes approches dans la sous-section « modèles appliqués à l'approche ROLAP » (page 21).

Le principal avantage de ROLAP réside dans le fait que la majorité des entreprises possède une grande expertise (à l'utilisation et à l'administration) des bases de données relationnelles. De plus, ces logiciels ont fait leurs preuves notamment sur le plan des performances, et sont, de surcroît, normalisés.

L'inconvénient de cette implantation est dû à la nature même des SGBDR : les coûts des opérations de jointure entre des tables contenant beaucoup d'enregistrements sont considérables. Afin de réduire au maximum le nombre de jonctions, on a recours à la dénormalisation, concept clé de ROLAP, mais paradoxe fondamental avec la troisième forme normale pourtant base des performances du modèle relationnel. Autre atout dans la réduction des coûts d'exécution, la redondance, qui est le résultat de la matérialisation de vues, stockant des données agrégées fréquemment utilisées.

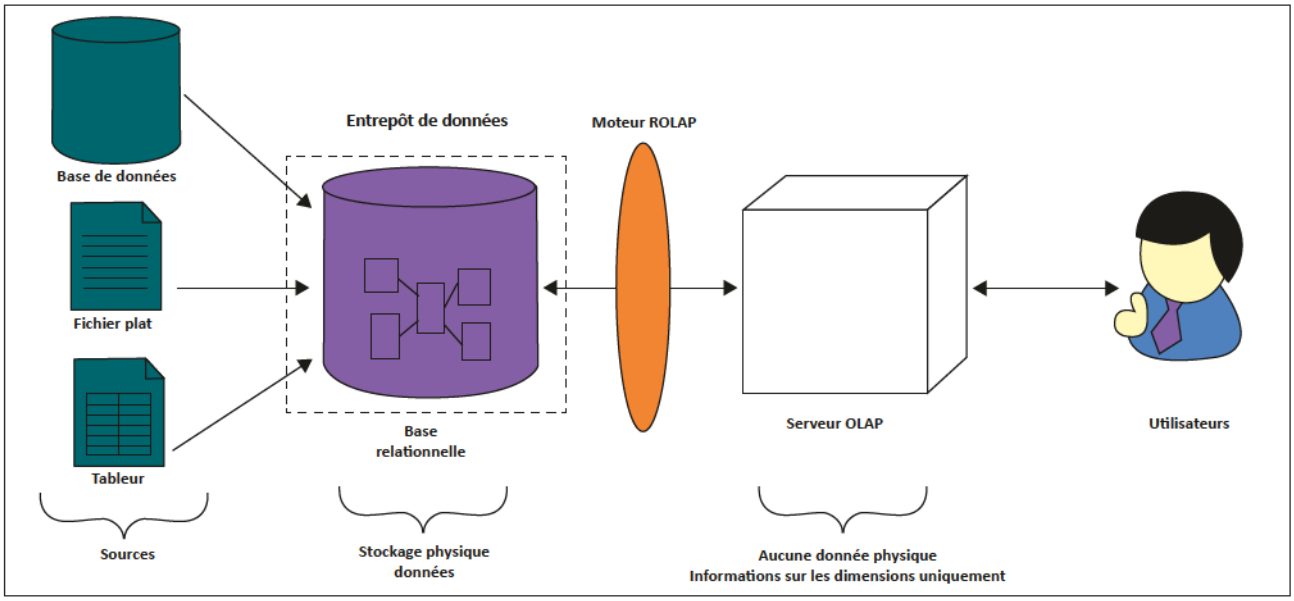



Figure II-6 : Architecture de l'approche ROLAP (Inspirée de [Développez.net, 2006])

D'un point de vue architecturel, l'utilisation de ROLAP requiert une couche applicative permettant la simulation de la multidimensionnalité entre le SGBD et le « navigateur d'agrégats » à proprement parlé, comme l'illustre la figure II-6.

### Approche MOLAP

Le *Multidimensional OLAP*, quant à lui, est basé sur un modèle logique ad hoc lui permettant d'opérer de manière optimale toute manipulation multidimensionnelle. Les données sont physiquement stockées sous forme de tableaux, et leur accès s'effectue directement via leur position (algorithmes *Grid-files*, *R*-trees* et *UB-trees*) [Techtarget, 2009]. Ainsi, modèles logiques et physiques convergent.

MOLAP a été créé pour palier aux manques de ROLAP : il s'exonère donc de la lourdeur des jointures des requêtes relationnelles, c'est pourquoi les performances en lecture sont généralement bien

meilleures [Jensen & Torben, 2001]. De plus, sur toutes les dimensions, les agrégations sont précalculées, améliorant encore les temps d'accès. Enfin, les données étant déjà sous la forme multidimensionnelle, l'accès y est direct (figure II-7).

Il existe une grande diversité de fonctionnement entre les outils MOLAP : l'absence de normes et de standards a laissé le champ libre aux très grands acteurs du domaine. Ce manque est l'un des inconvénients majeurs de cette approche. L'autre point sensible concerne la gestion de l'absence de données (la valeur nulle) [Serna Encinas, 2005]. Enfin, le niveau de détail, dans le cube, sera plus limité que dans le modèle ROLAP. En effet, du fait de la redondance d'information occasionnée par les agrégations, et de la grande cardinalité que l'on peut rencontrer dans les structures relationnelles pour arriver à une granularité assez fine, les limites techniques du modèle multidimensionnel sont rapidement atteintes.

Enfin, il n'existe pas, à l'heure où nous écrivons ces lignes, de SGBD multidimensionnel à même de traiter l'information spatiale.

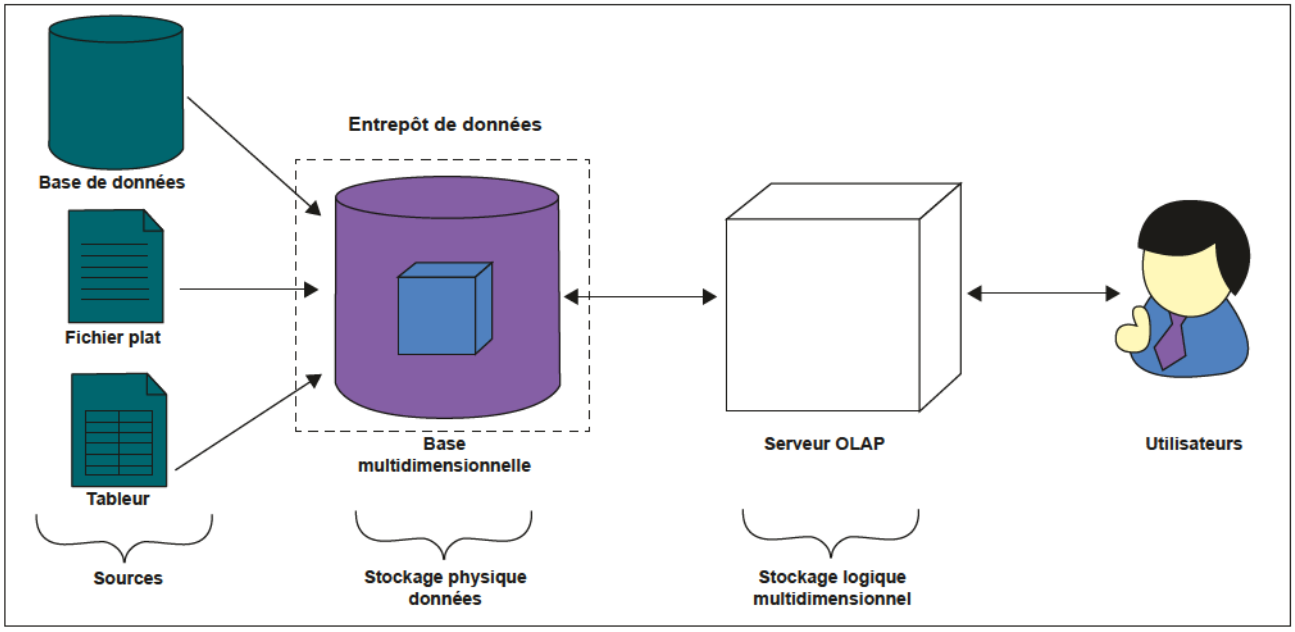



Figure II-7 : Architecture de l'approche MOLAP

### Approche HOLAP

L'*Hybrid OLAP* est la solution à mi-chemin des deux premières, prenant à son compte les avantages de chacune : il a été conçu pour combiner les grandes capacités de stockage de ROLAP et la supériorité de MOLAP en terme d'exécution [Techtarget, 2000].

Cette approche stocke, d'une part, les données granulaires dans une base relationnelle, s'appuyant ainsi sur les avantages de la standardisation et les capacités de gestion de stockage de grands volumes d'informations de ROLAP ; d'autre part, construit (logiquement et physiquement) un système multidimensionnel répondant aux requêtes les plus fréquentes, en les pré-calculant, profitant de cette façon de la rapidité de réponse offerte par MOLAP. Le système bascule sur le fonctionnement relationnel de manière transparente dès qu'il ne lui est pas possible de répondre à la demande grâce au modèle multidimensionnel. Cette approche permet d'avoir accès à des données bien plus fines (dont la granularité est plus grande), tout en ne sacrifiant pas aux performances.

Les inconvénients de cette approche (figure II-8) résident dans la très forte redondance d'informations et la maîtrise de la cohérence entre les deux modèles. La gestion spatiale de HOLAP souffre des mêmes limites que MOLAP.

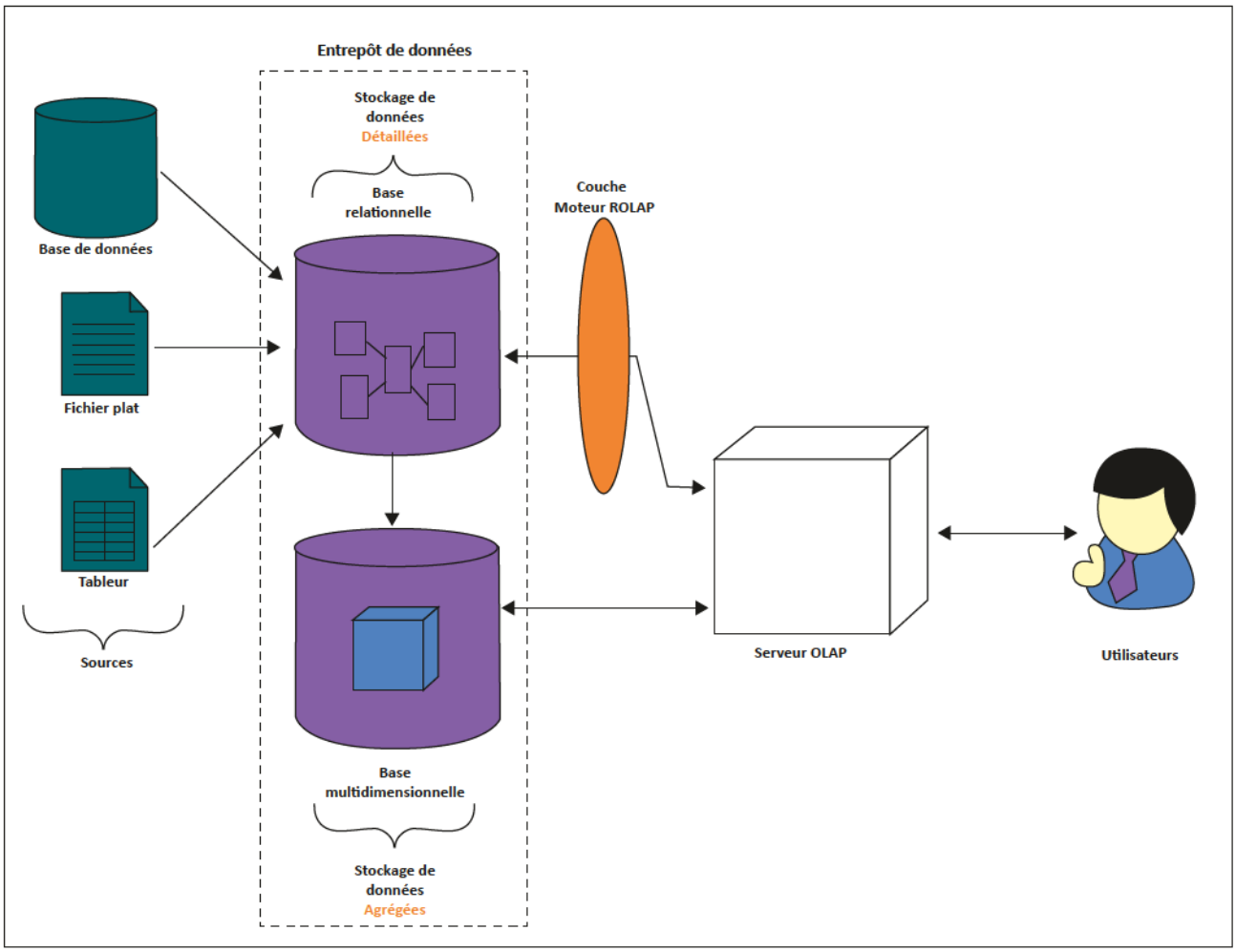



Figure II-8 : Architecture de l'approche HOLAP (Inspirée de [Développez.net, 2006])

### Autres approches

Dans l'écosystème de l'informatique décisionnelle, certaines architectures n'ont pas, à proprement parlé, d'entrepôt de données dédié à l'environnement : les requêtes sont effectuées directement dans les systèmes hébergeant les sources opérationnelles. Ceci occasionne généralement de mauvaises performances (extraction de données compliquée, nombreux supports à gérer, canaux d'accès inadaptés, aucun précalcul). Au vue de la volumétrie de données concernées par notre projet, nous nous passerons de ce type de fonctionnement, et ne sera pas plus décrit dans notre mémoire.

Signalons cependant brièvement trois autres « approches », qui n'ont comme point commun que de contenir OLAP dans leur nom tant elles sont différentes les unes par rapport aux autres.

D'abord, le *Desktop OLAP* (DOLAP), cas particulier de fonctionnement de ROLAP et MOLAP, où il s'agit d'un entrepôt de données miniature non connecté aux sources (typiquement un marché de données), embarqué sur un poste nomade. Ainsi l'utilisateur peut profiter de l'analyse sans pour autant devoir être connecté au serveur hébergeant l'entrepôt (en déplacement, hors ligne).

Ensuite, XML-OLAP (aussi appelé XOLAP). Il s'agit ici essentiellement d'un système de stockage, basé sur des fichiers XML. Aucun système commercial[5] n'est encore fondé sur cette technologie, dont les recherches commencent seulement à aboutir [Choquet, 2006], [Choquet & Boussaïd, 2007], [Hachicha *et al.*, 2007].

Pour finir, l'OLAP sémantique propose, de son côté, une vision avancée de l'exploitation de l'OLAP beaucoup plus orientée vers la fouille de données (*datamining*) et la recherche de règles [Loudcher, 2011].

En conclusion, les principales approches OLAP permettent de répondre, généralement, aux problématiques courantes des entreprises ou organisations qui souhaitent mettre en place un système d'aide à la décision.

### Modèles appliqués à l'approche ROLAP

Les bases de données relationnelles n'embarquent pas nativement les concepts de « dimensions », de « mesures » ou de « hiérarchies », il est alors indispensable d'émuler ces principes via une disposition particulière des outils disponibles : tables, attributs, relations et contraintes d'intégrité.

Le fait, élément central des structures, est stocké dans une table. Cette table possède généralement une clé primaire qui lui est propre et de clés étrangères référençant les dimensions dont le fait dépend. Les clés étrangères sont indexées afin d'optimiser les performances des requêtes. Le fait abrite aussi les attributs destinés aux mesures. Notons tout de même qu'il arrive parfois que la clé primaire soit construite grâce aux clés, impliquant que toutes les combinaisons d'utilisation des dimensions soient uniques.

Il existe quatre modèles pour organiser les dimensions autour des faits [Tranchant, 2011].

### Modèle en étoile

La structure en étoile tire son nom de sa configuration, au centre, la table des faits, autour, les dimensions, contenant les attributs (voir figure II-9). Cette forme est la plus communément rencontrée, lorsque l'approche ROLAP est adoptée. Ce choix est généralement justifié par la simplicité de sa mise en place, le recours à un nombre limité de jointures (restreint à celles entre le fait et chacune des dimensions).

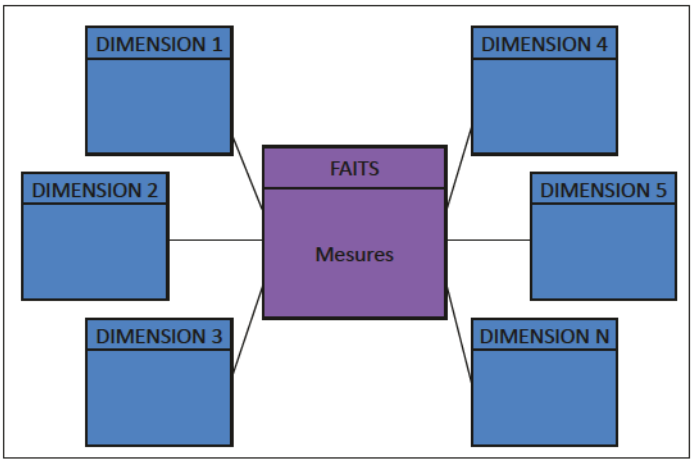



**Figure II-9 : Structure en étoile (Source : [Bédard, 2008])**

[5] Seuls deux logiciels libres nommés XML-OLAP (abandonné en juin 2006) et Open XML-Indri (arrêté en 2009) mettent en œuvre cette approche : http://sourceforge.net/projects/xml-olap/ http://projets-gmi.iup.univ-avignon.fr/projets/proj0809/M1/p08/oxid/

Cette disposition implique une forte dénormalisation, imposant la duplication des attributs de niveaux supérieurs. En effet, pour introduire les niveaux d'une hiérarchie, les données doivent être répétées autant de fois qu'il y a de lignes dans cette dimension, comme le montrent, en exemple, le tableau II-1 et la figure II-10, reproduisant une extraction de la table d'une dimension représentant les vendeurs d'une société .

**Table « Vendeur »**

| id | nom | service |
|---|---|---|
| 1 | Marcel PATALI | Hi-Fi |
| 2 | Marie CUZA | Hi-Fi |
| 3 | Roméo TAJOI | Hi-Fi |
| 4 | Dorine SELENA | Puériculture |
| 5 | Ramon PEREZ | Puériculture |

Tableau II-1 : Extraction d'une table de dimension « Vendeur », limitée à cinq entrées

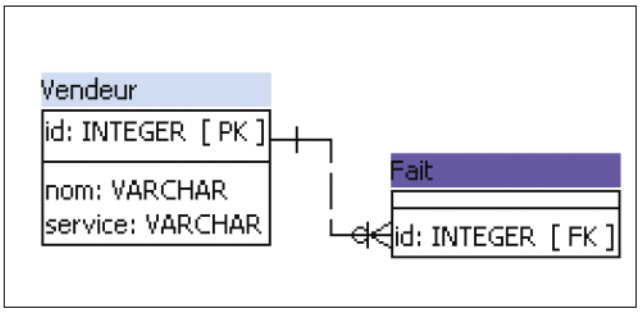



Figure II-10 : Modèle physique de données d'une dimension « Vendeur » en schéma étoile

Cet exemple illustre la duplication des informations « service », données qui auraient été normalisées dans le cas du schéma flocon.

### Modèle en flocon

La structure en flocon (figure II-11) est dérivée de celle en étoile : au centre, la table de faits, autour, celles des dimensions. Ces dernières sont décomposées suivant les niveaux de leur hiérarchie et sont plus normalisées que pour le schéma étoile. Ainsi, dans chaque dimension, une table représente un niveau.

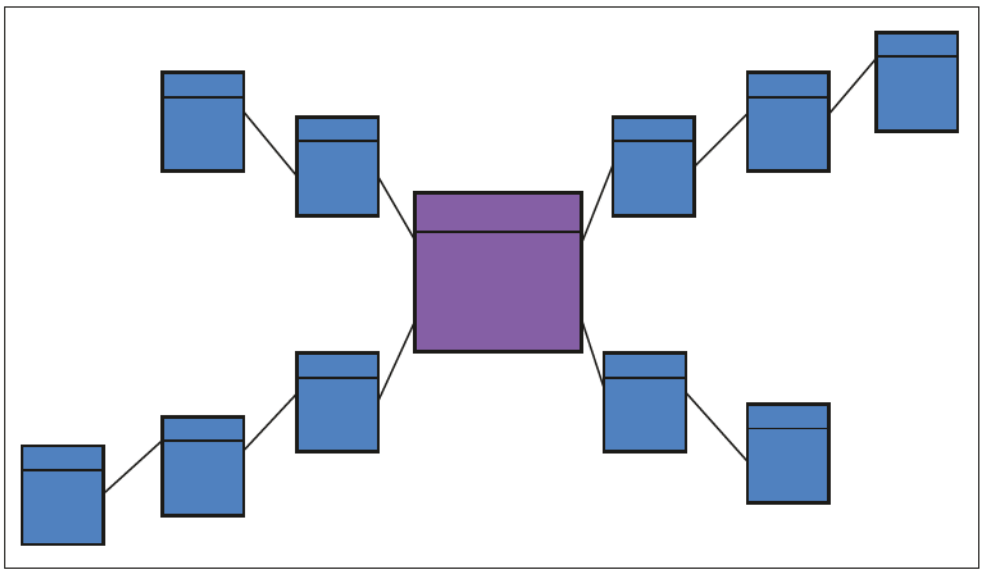


Figure II-11 : Structure en flocon (Source : [Bédard, 2008])

Pour reprendre l'exemple de la dimension « vendeur » du schéma étoile, nous trouverions ici deux tables : d'abord « vendeur », au plus près du fait, puis « service », mère de la première. Cette architecture est illustrée par le tableau II-2 et la figure II-12 ci-contre.

**Table « Service »**

| service_id | nom |
|---|---|
| 1 | Hi-Fi |
| 2 | Puériculture |

**Table « Vendeur »**

| id | nom | service_id |
|---|---|---|
| 1 | Marcel PATALI | 1 |
| 2 | Marie CUZA | 1 |
| 3 | Roméo TAJOI | 1 |
| 4 | Dorine SELENA | 2 |
| 5 | Ramon PEREZ | 2 |

Tableau II-2 : Extractions des tables factices de la dimension « Vendeur » dans le cas d'un schéma flocon

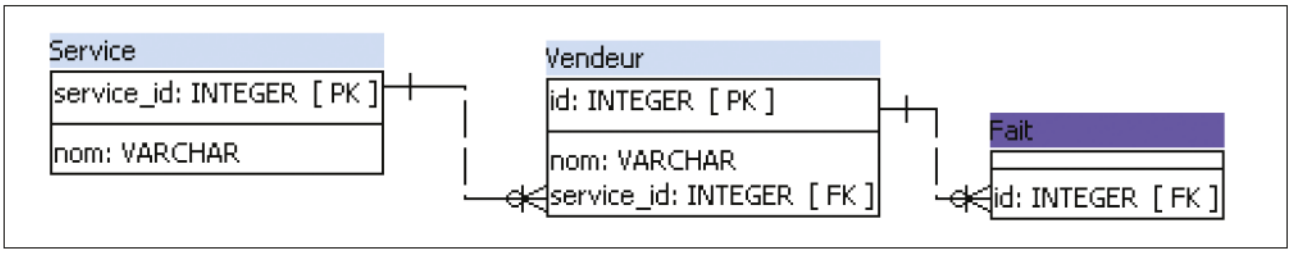



Figure II-12 : Représentation physique d'une dimension « Vendeur » en schéma flocon

Bien moins adopté par le modèle décisionnel à cause des jointures nécessaires à son exploitation, occasionnant de fait une consommation accrue de ressources lors d'une interrogation, ce schéma est cependant une solution à la non-redondance d'informations. Il est parfois utile d'éviter les doublons, c'est notamment le cas lorsque ces informations sont volumineuses.

### Modèle mixte

Résultat de la combinaison des structures en étoile et en flocon, cette solution, illustrée par la figure II-13, exploite les avantages des deux : certaines dimensions sont normalisées, généralement les plus grandes tables.

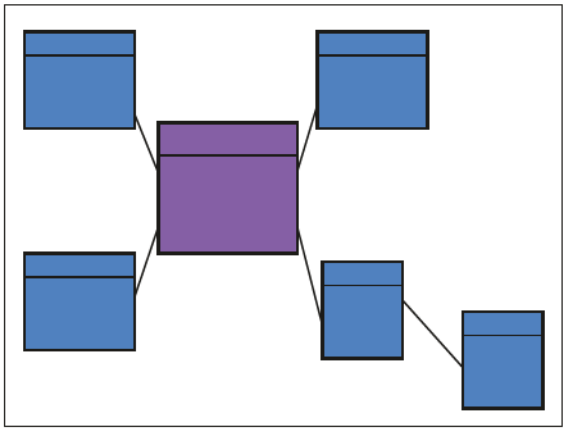


Figure II-13 : Structure mixte (Source : [Bédard, 2008])

### Modèle en constellation

Dans le cas particulier de la structure en constellation, plusieurs faits partagent certaines tables de dimensions (figure II-14). Ces dimensions sont qualifiées de conformes. Autour de ces faits, le schéma utilisé peut-être en étoile, en flocon ou mixte. L'utilisation de dimensions conformes permet d'obtenir une grande cohérence entre les vues destinées aux utilisateurs et oblige à une homogénéité des attributs et des mesures.

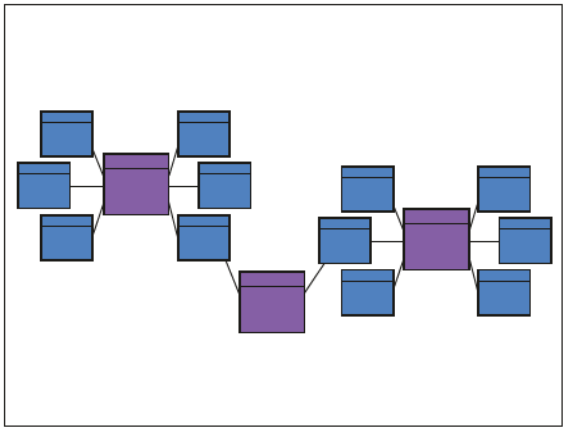


Figure II-14 : Structure en constellation (Source : [Bédard, 2008)

**Marchés de données**

Plus généralement désignés par le terme anglo-saxon *datamarts*, les marchés de données sont des entrepôts spécialisés sur un thème ou un métier et de plus petites tailles. Yvan Bédard définit que « l'entrepôt [...] est prévu pour l'entreprise dans son ensemble alors que le marché de données est sectoriel (il peut être un sous-ensemble exact ou modifié de l'entrepôt de données) » [Rougé-Libourel, 2008].

Ces *datamarts* sont généralement utilisés dans les très grands systèmes, où de nombreux utilisateurs doivent pouvoir effectuer leurs analyses, sans impacter les autres. La segmentation a ces avantages de cloisonner les accès (inutile au directeur des achats de parcourir les données des ressources humaines), et de permettre de répartir la charge occasionnée par les interrogations des entrepôts sur plusieurs serveurs.

À l'instar du modèle en constellation, impliquant des dimensions conformes au sein d'un même entrepôt, la conformité des dimensions, et, qui plus est, des faits, permet une grande uniformité des vues d'un marché de données à un autre. Un fait conforme est un « fait ayant la même définition dans tous les *datamarts*, la même unité de mesure, le même contexte dimensionnel » [Desnos, 2005].

### *d. Traitement analytique en ligne OLAP*

Le terme *On-line Analytical Processing* (OLAP), introduit par Edgar Frank Codd, désigne « une catégorie d'applications et de technologies permettant de collecter, stocker, traiter et restituer des données multidimensionnelles à des fins d'analyses » [Lupin, 2007].

À l'exemple de celles édictées pour le modèle relationnel, il a énoncé 12 règles régissant l'OLAP[6] [Codd *et al.*, 1993] :

1. Transparence : les services offerts par l'OLAP doivent être à la disposition de l'analyste, sans qu'il doive se préoccuper de l'emplacement du serveur, ni de l'impact de ces manipulations sur le système hôte. Toute complexité inutile doit être éludée ;
2. Accessibilité : les données doivent toutes être accessibles, sans ambiguïté. De plus, le système doit connaître la provenance et les transformations exercées sur elles ;
3. Manipulation intuitive des données : la navigation doit pouvoir s'effectuer naturellement via des interfaces ergonomiques. Les fonctionnalités sont directement accessibles ;
4. Souplesse d'affichage et flexibilité : le serveur doit permettre souplesse pour l'édition et réutilisation des rapports générés. Chaque requête doit être représentable sur une page, le nombre de dimensions ne doit pas importer ;
5. Multidimensionnalité : il s'agit de la nature même de l'OLAP. L'analyse est plus naturelle sur plusieurs dimensions que sur une seule, notamment grâce aux opérateurs pivot ou forage (décrits dans la section suivante) ;
6. Client-serveur : cette architecture permet la réalisation des suivantes. Elle doit aussi faciliter son accessibilité aux clients ;
7. Multi-utilisateur : l'accès et les recherches simultanés de la base doivent être possibles ;
8. Performances de production de rapport constantes : les performances sont indépendantes du nombre de dimensions, ce nombre et le niveau d'agrégation doivent pouvoir être modifiés

[6] Il faut cependant préciser que cette définition est sujette à caution à cause d'un conflit d'intérêt dû à la position de Codd lors de son écriture. Au moment de son écriture, Codd était, en effet, employé par la société qui allait devenir Hyperion Solutions, éditeur d'Essbase, depuis racheté par Oracle Corporation.

(et surtout augmentés) sans impact significatif sur les performances de production de rapports ;

9. Gestion dynamique des matrices creuses : le serveur supporte la représentation d'informations manquantes. Une faible densité de données dans un entrepôt peut rapidement impliquer de nombreux problèmes de performances dans les calculs ou une consommation excessivement inutile de place ;
10. Croisement des dimensions : le système permet d'effectuer des opérations entre et dans les dimensions. Ainsi les niveaux d'une même dimension se hiérarchisent, et les niveaux de dimensions différentes se croisent, dont le résultat est le produit cartésien ;
11. Dimensionnalité générique : toutes les dimensions sont équivalentes sur les plans structurels et des capacités opérationnelles. De plus, elles sont indépendantes ;
12. Analyse « sans limite » : le nombre de dimensions et de niveaux d'agrégation ne doit pas être restreint, afin de permettre des analyses complexes.

Codd considère que l'OLAP est un outil de médiation entre les logiciels éditant des rapports et les différentes sources auxquelles les utilisateurs finaux peuvent avoir un accès indirect, grâce à l'utilisation systématique de vues multidimensionnelles. Il s'agit d'une technologie pensée pour être manipulable rapidement, afin de changer de perspectives fréquemment et d'obtenir les résultats spontanément.

### *i. Opérateurs OLAP*

L'OLAP propose des opérateurs pour offrir une analyse la moins contraignante et la plus efficace possible. Au nombre de six, ces mécanismes servent à naviguer dans les dimensions, les hiérarchies et les niveaux [Tranchant, 2011].

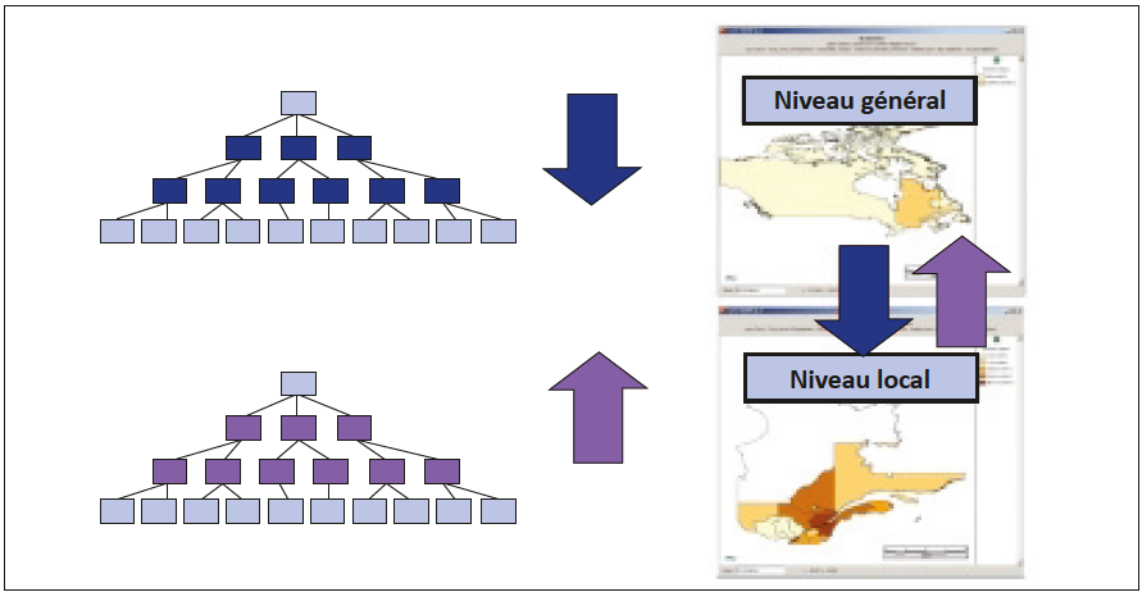



**Figure II-15 : Opérateurs de forage (en haut, en bleu) et de remontée (en bas, en violet) (Source : [Le Rubrus, 2009])**

Les opérateurs de forage (*drill-down*) et de remontée (*roll-up*) sont les outils les plus naturels pour la navigation dans la hiérarchie d'une dimension. Le premier permet de spécialiser une requête en descendant dans les niveaux, on peut ainsi passer du niveau continent à celui des pays, pour une dimension spatiale, ou de l'année au trimestre pour le temporel (figure II-15, partie supérieure). Le second offre une synthétisation des informations au niveau immédiatement supérieur : on pourrait passer du niveau départemental à régional spatialement parlant, ou des mois aux trimestres, temporellement (figure II-15, partie inférieure).

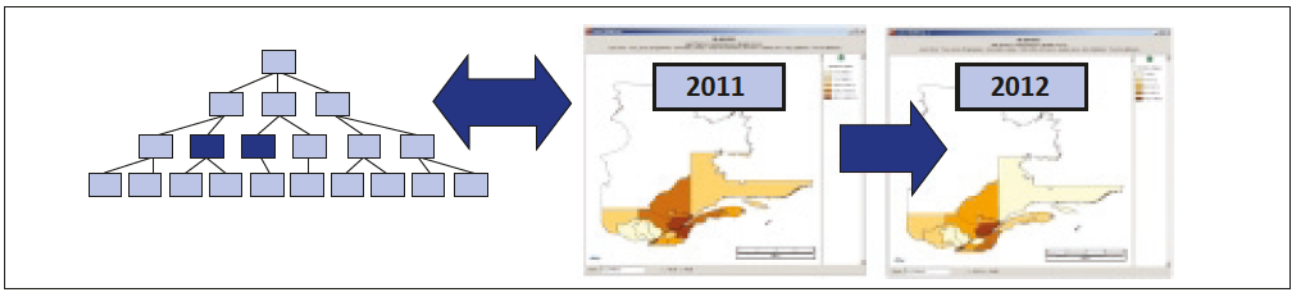



Figure II-16 : Opérateur de forage latéral (Source : [Le Rubrus, 2009])

Le forage latéral (*drill-across*) autorise une exploration horizontale d'une dimension. Tout en restant au même niveau, on passe d'un membre à un autre. La figure II-16 illustre un forage latéral dans la dimension temporelle, au niveau « année » sautant du membre « 2011 » à « 2012 ».

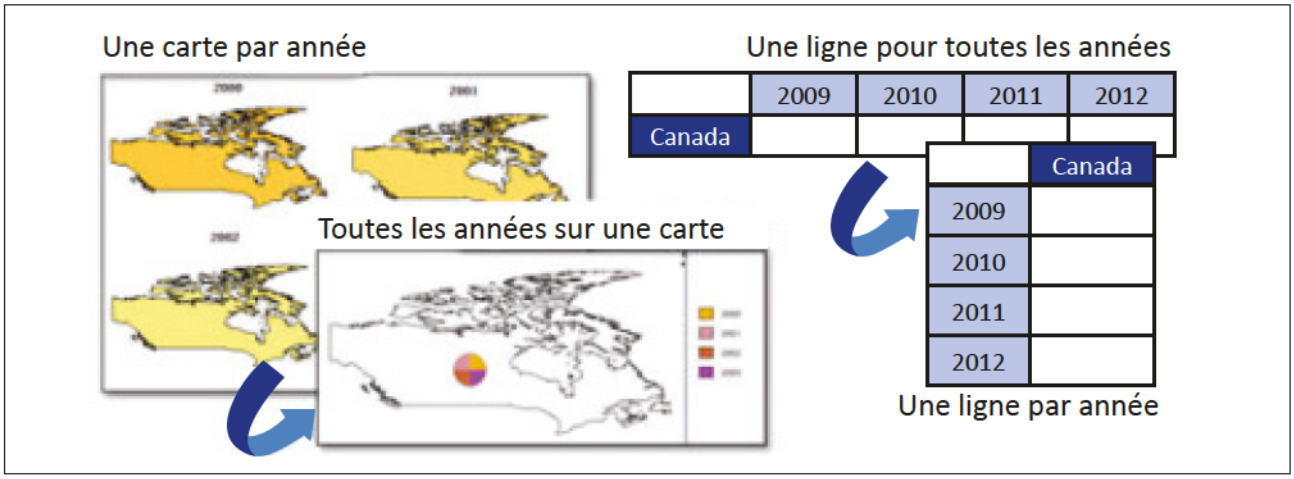



Figure II-17 : Opérateur pivot (Source : [Le Rubrus, 2009])

Pivoter (la communauté anglo-saxonne accepte deux termes : *swap* et *rotate*) : permet d'interchanger les deux axes principaux. Les éléments composant l'axe « ligne » seront placés en « colonne » et inversement. Il s'agit d'apporter une autre perspective aisément (figure II-17).

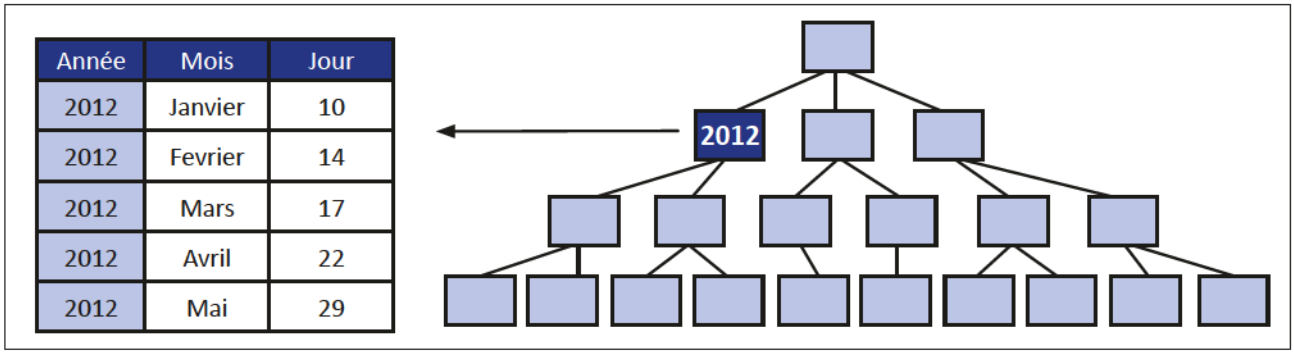



Figure II-18 : Opérateur de perçage

L'opérateur de perçage (*drill-through*) donne accès aux données sous-jacentes à celles stockées dans le cube : pour une cellule donnée, il est alors possible d'obtenir les informations qui la composent. Si cette opération est facilement réalisable à un niveau élevé de la hiérarchie (figure II-18), il devient normalement impossible lorsqu'on se trouve à la granularité maximale. Cependant, ceci est généralement autorisé par le modèle OLAP Hybride. Dans ce cas, cette fonction est tributaire de la structure de l'entrepôt, et surtout de la façon dont les données sources sont traitées.

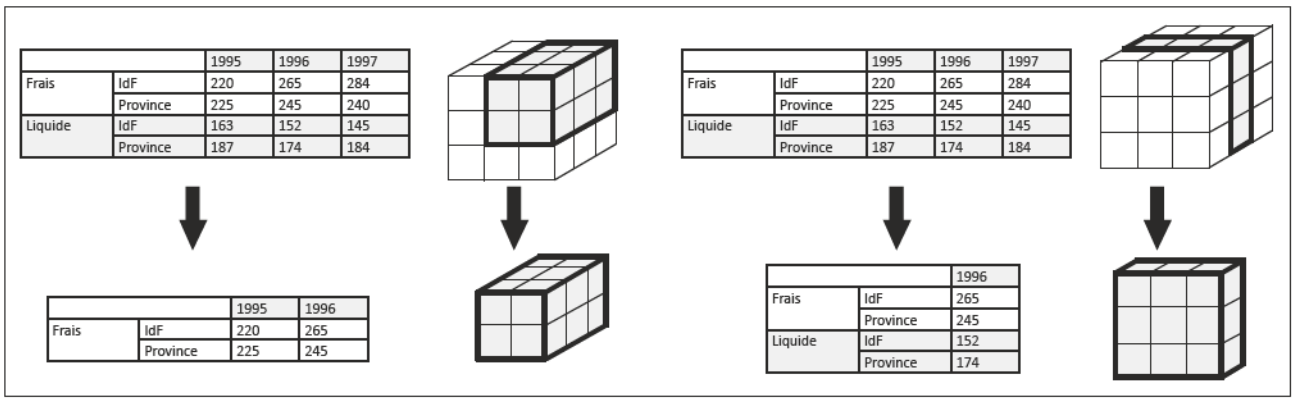



**Figure II-19 : Opérateurs de taillage (*slicing* à gauche et *scoping* à droite) (Source : [Donsez, 2006])**

Tailler (*slicing* et *scoping*) : véritables filtres de données, ces opérateurs autorisent l'extraction d'une partie de l'hypercube afin d'en produire un plus petit et ne retenir que les données désirées. Cette sélection permet, par l'exemple illustré par la figure II-19, à l'utilisateur de ne garder que les produits de type « Liquide » grâce au *slicing*, ou de ne consulter que les ventes de 1996, via le *scoping*.

### *ii. SOLAP : l'OLAP appliqué aux données spatiales*

Issu du mariage du monde décisionnel et des systèmes d'informations géographiques, l'OLAP spatial est l'outil idoine pour l'interrogation de cubes contenant des données géographiquement référencées.

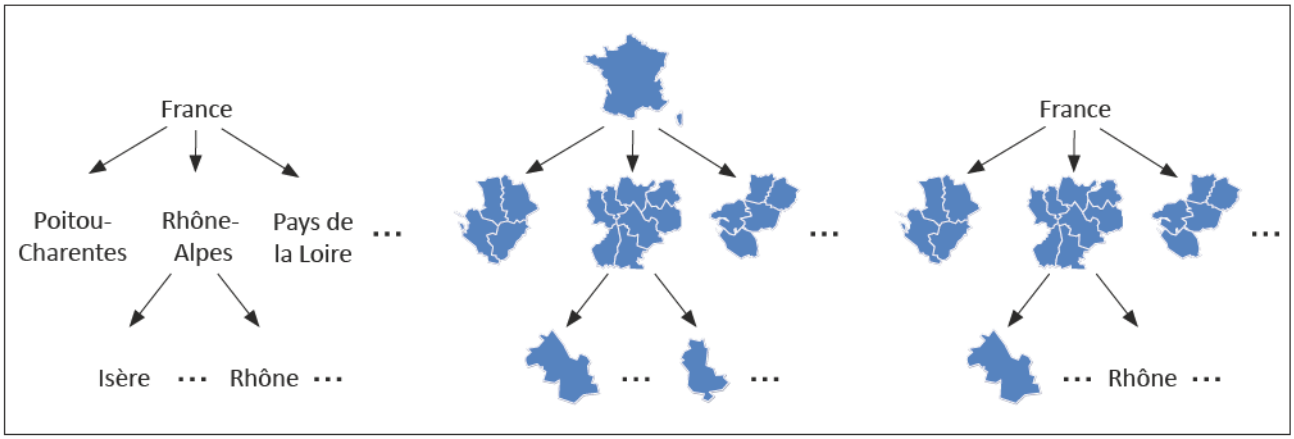



**Figure II-20 : Représentation des données spatiales, de gauche à droite, respectivement descriptive, géométrique et mixte (Source : [Tranchant, 2011])**

Le principal atout de cette solution est l'exploitation des informations géométriques, tandis que les outils OLAP classiques se contentent d'informations descriptives (voir figure II-20). Les géométries ont la particularité de pouvoir être manipulées de façon plus complète : à permettre, par exemple, d'opérer des intersections entre deux objets ou de filtrer par des critères de distance, opérations impossibles dans un moteur OLAP traditionnel. Une description plus détaillée des propriétés des données spatiales, ainsi que des fonctions attendues est proposée dans la section « Particularité de l'information spatiale » (page 41).

Le besoin de gérer ce type de données en exploitant le maximum de ces capacités provient du fait que, selon [Bimonte *et al.*, 2005], 80% des données contenues dans un entrepôt de données possèdent une composante géographique : il convenait donc de mettre en place les outils les plus

adaptés à cette spécificité. La cartographie est la façon la plus naturelle d'exploiter ces informations : elle permet de situer facilement les unités spatiales entre elles, tout en montrant des indicateurs parlant, grâce à la sémiologie*.

SOLAP exploite à la fois les capacités de l'analyse multidimensionnelle offertes par OLAP et les aptitudes des SIG à gérer les propriétés des données géoréférencées. À la fois concept et technologie, SOLAP est sensé représenter le juste équilibre entre les solutions dites « OLAP-dominantes » dans lesquelles les fonctions géographiques sont restreintes, et celles considérées comme « SIG-dominantes », où, au contraire, les opérateurs OLAP ne sont qu'embryonnaires, voire n'existent pas [LGS Group, 2000].

À l'origine de cet acronyme, Yvan Bédard a enrichi la définition d'OLAP en positionnant les logiciels SOLAP comme « des logiciels de navigation rapide et facile dans des bases de données spatiales à plusieurs niveaux de granularité d'information, plusieurs thèmes, plusieurs époques et plusieurs modes de visualisation synchronisés ou non : cartes, tableaux et diagrammes » [Bédard *et al.*, 2005].

Les outils SOLAP existants ont été l'objet d'une comparaison lors de notre travail d'étude et de synthèse technique de 2011 [Tranchant, 2011]. Nous reviendrons sur les conclusions de ce document dans la partie « Démarche et proposition », chapitre « Décisions ».

### *iii. XML for Analysis, une méthode d'accès*

Pour permettre l'accès aux systèmes analytiques depuis un outil client tiers, Hypérion, Microsoft, SAS et SAP, regroupés au sein du *XMLA Council*, ont co-développé *XML for Analysis* (généralement désigné par le sigle XMLA). Il s'agit d'une interface de programmation (*Application Programming Interface* ou API) intégrée aux logiciels OLAP, permettant l'interrogation à distance des cubes que le serveur contient. Standard *de facto*, il s'appuie sur les protocoles HTTP (*HyperText Transfert Protocol*) et SOAP (*Simple Object Access Protocol*) ainsi que sur le langage XML (*eXtensible Markup Language*) pour le transfert des messages en provenance et à destination du serveur.

Le langage de requête utilisé pour questionner la base multidimensionnelle est MDX (*MultiDimensional eXpressions*).

XMLA définit un éventail de messages XML utilisés pour permettre l'interaction entre le fournisseur de données analytiques et l'application consommatrice, messages divisés en deux catégories : « *discover* » et « *execute* » [Microsoft & Hyperion, 2002].

#### Messages « *Discover* »

Les messages « *discover* » sont utilisés pour découvrir les services et les métadonnées du serveur. Dans cette catégorie, on trouve une multitude de services, dont quelques-uns des plus importants sont listés dans le tableau II-3.

Grâce à ces messages, le consommateur des ressources du fournisseur de données peut avoir accès à la description de tous les schémas et sources d'informations qu'il contient. Un exemple de requête est présentée dans l'annexe A.

| Nom du service | Description |
|---|---|
| *DISCOVER_DATASOURCES* | Fournit la liste des sources de données accessibles par le serveur. |
| *DISCOVER_PROPERTIES* | Retourne une liste d'informations et de valeurs sur les propriétés standards ou spécifiques supportées par le fournisseur de données. |
| *DISCOVER_SCHEMA_ROWSETS* | Donne les noms, descriptions, et types des valeurs des méta-informations supportées par le serveur (standards ou spécifiques au fournisseur). |
| *DISCOVER_ENUMERATORS* | Produit la liste des caractéristiques spécifiques au fournisseur. |
| *DISCOVER_KEYWORDS* | Communique une énumération des mots-clés alloués au fournisseur de données. |
| *DISCOVER_LITERALS* | Retourne la liste des informations sur les littéraux supportés par le fournisseur. |

**Tableau II-3 : Liste des messages « *discover* » les plus rencontrés (Source : [Microsoft & Hyperion, 2002])**

**Messages « *Execute* »**

Les méthodes « *execute* », quant à elles, sont employées pour envoyer des requêtes impliquant des données : la récupération (et l'écriture) des informations contenues dans le cube. Ce sont celles qui acheminent les requêtes MDX jusqu'au serveur, en les adressant à la source ciblée (les serveurs pouvant accéder à plusieurs sources).

L'annexe B expose un échange d'un message « *execute* » du consommateur au fournisseur, et la réponse.

### *iv. MultiDimensional eXpressions*

Nous avons déjà très brièvement évoqué le langage de requête dédié à l'exploration des structures multidimensionnelles. MDX, sigle de *MultiDimensional eXpressions*, est donc le moyen offert pour la manipulation des éléments du cube.

Ce langage est à la fois très riche et complexe : il offre de nombreuses possibilités par le biais de fonctions spécifiques, ce qui en facilite l'utilisation technique, tant que les requêtes restent simples. Il n'en est pas de même lorsque les difficultés commence dès lors que l'on croise des dimensions différentes, et ce, sur plusieurs niveaux, par exemple.

Décrire exhaustivement MDX pourrait faire l'objet d'un ouvrage à part entière. Sans verser dans cet extrême, nous allons exposer les grands principes et notions qui entrent en jeu dans ce langage.

**Principes**

Une requête MDX est organisée autour d'axes, qui définissent les perspectives d'études. Ces axes sont désignés par leur nom (*columns*, *rows, pages, sections* et *chapters*) ou par leur position (« 0 » correspond aux colonnes, « 1 » aux lignes, etc.). Généralement, seuls les deux premiers sont utilisés. On affecte à ces axes des dimensions, des hiérarchies ou des niveaux, par l'utilisation de membres, de *n*-uplets ou de *sets*.

Un membre est une instance d'un niveau : « Europe » est un membre du niveau « Continent » de la dimension spatiale.

Un *n*-uplet représente une intersection unique des dimensions en jeu. Pour illustrer un *n*-uplet, prenons l'exemple du croisement de trois dimensions : nous cherchons à déterminer les ventes effectuées par l'un de nos commerciaux (dimension « salarié », niveau « commercial », membre « Alain Martin »), pour le premier trimestre de 2012 (dimension « temporel », membre « trimestre 1 » du niveau « 2012 ») pour l'agence de Grenoble (dimension « agence », membre « Grenoble »). Le *n*-uplet en question sera exprimé ainsi (figure II-21) :

```
(
   [Salarié].[Commercial].[Alain Martin],
   [Temporel].[2012].[Trimestre 1],
   [Agence].[Grenoble]
)
```

**Figure II-21 : Exemple de *n*-uplet à trois dimensions**

Un *set* est un ensemble ordonné de zéro à *n*-uplets de la même dimension [iCCube, 2012].

À noter qu'il est interdit d'utiliser une même dimension (ou n'importe lequel de ses éléments fils) sur plus d'un axe.

### Sélections

Les éléments que l'on veut voir entrer en jeu dans une requête sont désignés soit par leur nom simple, soit par leur nom unique, soit via une fonction. Prenons l'exemple de la dimension temporelle, comme illustrée à la figure II-2 (page 13). Pour avoir accès au membre « 2012 », on peut le désigner :

- Par son nom : `[2012]`. Ceci n'est possible que grâce au fait qu'il soit unique, il serait donc impossible de le faire pour le membre `[Trimestre 1]` ;

- Par son nom unique : `[Temporel].[2012]`. Ce nom est construit sur l'architecture pour l'atteindre, ici, le nom de la dimension, puis le membre. Il existe une alternative à la construction de ce nom unique en ayant recours aux noms de niveaux. Dans ce cas, le nom unique est `[Temporel].[Année].[2012]`. Si on veut récupérer le mois de novembre de l'année 2012, il faut écrire `[Temporel].[2012].[Novembre]`;

- Via l'utilisation d'une fonction : `[Temporel].lastChild` (la dernière année étant 2012). Nous allons détailler ce dernier point.

Le recours aux fonctions est nécessaire pour parcourir les éléments composant les axes, ou pour leur appliquer des filtres.

```
SELECT
   ([Spatial].[Europe].children) ON COLUMNS,(A)
   [Mesures].[PIB] ON ROWS (B)
FROM
   [monCube]
WHERE
   [Temporel].[2011] (C)
```

**Figure II-22 : Exemple de requête MDX mettant en œuvre une fonction de niveau**

La figure II-22 propose un exemple simple de requête MDX. Pour illustrer la finalité des fonction de sélection :

- (A) En colonne : la fonction « *children* » permet de retourner tous les enfants d'un membre, donc, ici les enfants du membre « Europe » de la dimension spatiale ;
- (B) En ligne : le membre « PIB » de la dimension contenant les mesures ;
- (C) En filtre (*slicer*) : le membre « 2011 » de la dimension temporelle.

Celle-ci permet de répondre à la question multidimensionnelle suivante : « afficher les données du PIB pour tous les pays de l'Europe, pour l'année 2011 ».

**Fonctions et opérateurs**

Il existe de nombreuses fonctions appliquées aux dimensions, hiérarchies et membres, toutes ayant trait aux relations de généalogies. En voici quelques exemples :

- *members* : permet récupérer les membres d'une dimension ou d'un niveau ;
- *parent* : retourne l'élément hiérarchiquement supérieur ;
- *firstChild* : donne le premier enfant de l'élément ;
- *lastSibling* : renvoie le dernier enfant du membre parent.

Les *sets* ont aussi leurs propres opérateurs. Ils permettent de filtrer (*filter*), de récupérer un sous-ensemble selon des critères (*subset*), de trier (*order*), d'exercer les actions ensemblistes telles que l'intersection (*intersect*), l'union (*union*), la différence (*except*), le produit cartésien (*crossjoin*), et quelques opérations spécifiques du monde décisionnel, comme l'imbrication hiérarchique des niveaux (*hierarchize*) ou la récupération des ancêtres d'un membre (*ancestor*) [MSDN MDX, 2012].

Comme les fonctions appliquées aux dimensions et niveaux, cette liste d'opérateurs est loin d'être exhaustive. Elle reflète cependant une grande partie de ceux qui sont usuellement utilisés.

L'exemple de la figure II-23 montre l'utilisation des opérateurs de hiérarchisation, d'union et de produits cartésiens sur les *sets*.

```
SELECT
  [Mesures].children ON COLUMNS,
  Hierarchize
  (
     Union
     (
        CrossJoin
        (
            [Spatial].[Continent].members,
            [Temporel].[Année].members
        ),
        CrossJoin
        (
            [Spatial].[Pays].members,
            [Temporel].[Année].members
        )
     )
  ) ON ROWS
FROM
   [monCube]
```

**Figure II-23 : Exemple de requête MDX mettant en œuvre un opérateur d'ensemble**

En décomposant la requête de la figure II-23, on constate que l'axe des lignes est complexe. Il faut lire la commande dans le sens du plus spécialisé au plus général, en l'occurrence, les deux *CrossJoin*, puis l'*Union* et enfin le *Hierarchize*, ce qui donne :

- Le produit cartésien du niveau « Pays » de la dimension spatiale avec le niveau « Année » de la dimension temporelle ;
- Le produit cartésien du niveau « Continent » de la dimension spatiale avec le niveau « Année » de la dimension temporelle ;
- L'union des deux produits ;
- La présentation hiérarchique du résultat.

Dans ce cas, la présentation tabulaire des données donnerait un résultat proche de celui du tableau II-4 ci-dessous (si l'on considère que l'on a des données uniquement sur 2010 et 2011 dans la dimension temporelle, sur la Chine, l'Inde, le Japon, l'Allemagne, la France et le Royaume-Uni sur la dimension spatiale, et les mesures du PIB et de la population) :

| Contient | Pays | Année | PIB | Population |
|---|---|---|---|---|
| **Asie** | | **2010** | 12 920 286 | 2 630 699 772 |
| **Asie** | | **2011** | 14 677 235 | 2 685 561 437 |
| | **Chine** | **2010** | 5 745 133 | 1 330 141 295 |
| | **Chine** | **2011** | 6 988 470 | 1 347 718 015 |
| | **Inde** | **2010** | 1 430 020 | 1 173 108 018 |
| | **Inde** | **2011** | 1 833 382 | 1 210 193 422 |
| | **Japon** | **2010** | 5 745 133 | 127 450 459 |
| | **Japon** | **2011** | 5 855 383 | 127 650 000 |
| **Europe** | | **2010** | 8 146 990 | 209 237 806 |
| **Europe** | | **2011** | 8 917 866 | 209 879 543 |
| | **Allemagne** | **2010** | 3 305 898 | 81 952 000 |
| | **Allemagne** | **2011** | 3 628 623 | 81 831 000 |
| | **France** | **2010** | 2 582 527 | 65 001 000 |
| | **France** | **2011** | 2 808 265 | 65 350 181 |
| | **Royaume-Uni** | **2010** | 2 258 565 | 62 284 806 |
| | **Royaume-Uni** | **2011** | 2 480 978 | 62 698 362 |

**Tableau II-4 : Exemple de résultat à la requête de la figure II-23**

Sur ce tableau II-4, en en-tête de lignes, on retrouve les deux niveaux de la dimension spatiale, croisés au niveau « Année » de la dimension temporelle. Sur les lignes où les seuls noms des continents apparaissent, les sommes des lignes des membres inférieurs sont totalisées.

### *e. Outils de visualisation*

Partie émergée du système décisionnel, les outils de visualisation sont l'objet d'une attention particulière, et ce, à plusieurs niveaux. D'abord ergonomiques, ils doivent être intuitifs et ne doivent pas nécessiter de longues périodes d'apprentissage pour maîtriser l'essentiel des commandes. Ensuite modulaires, les utilisateurs éprouveront une sensation de confort s'ils ont la possibilité de représenter les données sous plusieurs formes, selon leur desiderata. Enfin, ils représentent la « vitrine » du système : s'ils ne sont pas attractifs, ils ne donneront ni confiance de prime abord, ni l'envie de rester afin d'en exploiter les capacités.

Compte tenu de l'importance de cet élément de la chaîne décisionnelle dans notre sujet, il est décrit plus amplement dans le chapitre B « Visualisation de l'information » (page 34).

### *f. Métadonnées*

Les métadonnées, ou « données sur les données », sont omniprésentes dans les systèmes décisionnels. On considère qu'elles sont subdivisées en deux catégories. D'une part « structurelles », elles permettent de donner des informations sur le contenu de la structure dimensionnelle : il peut s'agir du système source d'origine, des transformations qui y sont appliquées, des redondances constatées avec d'autres systèmes, des règles de calculs auxquelles les données ont été soumises, date à laquelle la donnée a été ajoutée à l'entrepôt, sans que cette liste soit exhaustive. On parle aussi de « métaschéma ».

D'autre part, qualifiant « l'accessibilité », elles assurent l'interopérabilité entre les différentes ressources. En effet, par l'utilisation de mots-clés, par la construction d'un référentiel partageant les types de données, de supports, les fournisseurs, etc. Elles permettent d'établir des correspondances entre les systèmes et leur contenu.

Ces métadonnées sont accessibles via l'interface de manipulation des données.

À l'échelle européenne, la directive INSPIRE préconise une organisation répartie. D'un côté, les producteurs publient sur leur site (ou sur celui d'un partenaire) leurs métadonnées, et, de l'autre, les référencient dans des annuaires. Pour le cas des données spatiales, il s'agit de celui de l'institut géographique national (IGN), nommé « géocatalogue », sur le site GéoPortail[7].

Les métadonnées appliquées au domaine géographique sont soumises à plusieurs normes internationales, notamment les ISO19115, ISO19115-2 et ISO19119.

La première définit un cadre général, largement « modulable et extensible », la rendant adaptable. Elle concerne les informations d'identification, la description du contenu, le système de coordonnées (aussi appelé « référentiel », dont le terme anglais est *Spatial Reference System*, connu sous l'acronyme SRS), les informations de géolocalisation, des indicateurs de qualité, des informations de généalogie (sources des données, transformations appliquées), les modalités d'affichage (légendes), de diffusion et de maintenance [Eden-IGN, 2012].

La seconde étend la première en y apportant des précisions sur les parts généalogiques et qualitatives. Elle propose aussi un paquetage UML (*Unified Modeling Language*) complet permettant la définition des métadonnées d'acquisition [ISO, 2009].

Enfin, la troisième concerne les protocoles d'identification des services.

Après avoir passé en revue les différents constituant d'une chaîne géodécisionnelle, le chapitre suivant revient plus en détails sur l'étape des outils de visualisation.

[7] GéoPortail est le site national de recherche et de visualisation des données des administrations : http://www.geoportail.fr/

## B. Visualisation de l'information

La visualisation de l'information est un enjeu colossal dans tout système d'information (géo) décisionnel.

En directe relation avec l'utilisateur final, l'interface homme-machine (IHM) doit faire preuve de praticité, modularité et de fonctionnalité pour retenir son attention. Néanmoins, même si l'ergonomie de l'application reste un important sujet, la présentation des résultats des opérations que l'on y effectue le domine très largement.

Ce chapitre s'organise autour de deux aspects de la présentation des données. Le premier concerne les données tabulaires et graphiques issues de systèmes décisionnels. Pour illustrer nos propos, quelques outils sont passés en revue. Nous en extrayons les bons côtés pour nous en inspirer dans notre projet tout en tentant de résoudre les difficultés rencontrées. Le second aspect se concentre sur la singularité de l'information spatiale et de ces modes de représentation. Cette section en contient une brève définition puis conduit l'analyse des présentations cartographique au travers de logiciels existants, où, une nouvelle fois, nous dégagerons avantages et inconvénients.

### 1. Visualisation de données provenant de systèmes décisionnels

Les données informatiques, au sens large du terme, peuvent revêtir plusieurs formes : fichiers, chaînes de caractères, entiers, décimaux, formules, images, sons, etc. Les données contenues dans les faits des bases multidimensionnelles sont essentiellement numériques[8]. Quels sont les moyens pour les représenter et les interpréter ? Comment s'y prennent les logiciels du domaine des systèmes décisionnels ?

#### a. Moyens

**Tableau croisé multidimensionnel**

L'un des meilleurs moyens pour présenter un grand nombre d'enregistrements est, avec l'arbre, le tableau. Synthétique, il ne laisse aucune place à l'approximation : les informations sont à l'intérieur d'une cellule, elle-même indexée par un ou plusieurs en-têtes.

Les données extraites des bases de données relationnelles sont affichées sous forme de tableaux à une entrée : seules les colonnes possèdent une référence, correspondant aux attributs de la table. En ce qui concerne les systèmes décisionnels, le nombre de références correspond à celui des dimensions impliquées dans la requête. Par exemple, pour deux dimensions (réparties sur les deux axes principaux), le tableau possèdera deux en-têtes : un sur les colonnes, l'autre sur les lignes. Les problèmes de représentations commencent à partir de trois dimensions. En effet, il n'est pas aisé d'afficher un tableau en trois dimensions (même si les technologies d'affichage tridimensionnel ont été largement démocratisées ces dernières années). Et qu'en est-il des requêtes à cinq, six, dix dimensions ?

Pour émuler ce fonctionnement, les dimensions sont « empilées » sur chacun des deux axes disponibles, et forment des tableaux croisés multidimensionnels. La figure II-24 ci-contre montre un tableau croisé, extrait du tableur de Microsoft, Excel.

[8] Voir II.A.2.a Modélisation multidimensionnelle, page 12.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 4 | Étiquettes de lignes | Active Population 2006 thousands inhab | Unemployment Population 2006 thousands inhab | Area ha | Active Population 2001 thousands inhab |
| 23 | ⊞Mediterranean Sea | 328,80000 | 14,86200 | 962500,00000 | 311,86200 |
| 24 | ⊞**CZ Czech Republic** | **5174,36600** | **370,50700** | **7888600,00000** | **5097,45400** |
| 25 | ⊞Mediterranean Sea | 1368,71200 | 101,34900 | 2176800,00000 | 1337,24900 |
| 26 | ⊞North Eastern Atlantic Ocean | 3805,65400 | 269,15800 | 5711800,00000 | 3760,20500 |
| 27 | ⊞**DE Germany** | **41506,52800** | **4264,92800** | **36368600,00000** | **39067,92000** |
| 28 | ⊞Mediterranean Sea | 4844,82200 | 284,64300 | 5618900,00000 | 4692,43600 |
| 29 | ⊞North Eastern Atlantic Ocean | 36661,70600 | 3980,28500 | 30749700,00000 | 34375,48400 |
| 30 | ⊞**DK Denmark** | **2794,34600** | **108,79200** | **4717500,00000** | **2789,23600** |
| 31 | ⊞North Eastern Atlantic Ocean | 2794,34600 | 108,79200 | 4717500,00000 | 2789,23600 |
| 32 | ⊞**EE Estonia** | **685,23700** | **40,24200** | **4708900,00000** | **649,72100** |
| 33 | ⊞North Eastern Atlantic Ocean | 685,23700 | 40,24200 | 4708900,00000 | 649,72100 |
| 34 | ⊞**ES Spain** | **19513,11800** | **1659,43700** | **50963900,00000** | **17723,52400** |
| 35 | ⊞Mediterranean Sea | 8629,75900 | 652,88800 | 18829800,00000 | 7908,63700 |
| 36 | ⊞North Eastern Atlantic Ocean | 10883,35900 | 1006,54900 | 32134100,00000 | 9814,88700 |
| 37 | ⊞**FI Finland** | **2546,47400** | **198,14100** | **34663700,00000** | **2546,26600** |
| 38 | ⊞Artic Ocean | 3,96300 | 0,48500 | 2582200,00000 | 4,03700 |
| 39 | ⊞North Eastern Atlantic Ocean | 2542,51100 | 197,65600 | 32081500,00000 | 2542,22900 |
| 40 | ⊞**FR France** | **26094,70200** | **2297,07800** | **55451300,00000** | **25370,51500** |
| 41 | ⊞Mediterranean Sea | 5932,80100 | 560,95000 | 13046800,00000 | 5570,00600 |
| 42 | ⊞North Eastern Atlantic Ocean | 20161,90100 | 1736,12800 | 42404500,00000 | 19800,50900 |
| 43 | ⊞**GR Greece** | **4636,40400** | **411,27200** | **13929800,00000** | **4428,31000** |
| 44 | ⊞Mediterranean Sea | 4636,40400 | 411,27200 | 13929800,00000 | 4428,31000 |
| 45 | ⊞**HR Croatia** | **2,27300** | **0,13700** | **5902100,00000** | **2,26000** |
| 46 | ⊞Mediterranean Sea | 2,27300 | 0,13700 | 5902100,00000 | 2,26000 |
| 47 | ⊞**HU Hungary** | **4295,99600** | **320,35900** | **9314700,00000** | **4107,71200** |
| 48 | ⊞Mediterranean Sea | 4295,99600 | 320,35900 | 9314700,00000 | 4107,71200 |
| 49 | ⊞**IE Ireland** | **1835,73800** | **79,38000** | **7293100,00000** | **1734,18300** |
| 50 | ⊞North Eastern Atlantic Ocean | 1835,73800 | 79,38000 | 7293100,00000 | 1734,18300 |
| 51 | ⊞**IM Isle of Man** | **0,00000** | **0,00000** | **65500,00000** | **0,00000** |
| 52 | ⊞North Eastern Atlantic Ocean | 0,00000 | 0,00000 | 65500,00000 | 0,00000 |
| 53 | ⊞**IS Iceland** | **0,00000** | **0,00000** | **10617400,00000** | **0,00000** |
| 54 | ⊞North Eastern Atlantic Ocean | 0,00000 | 0,00000 | 10617400,00000 | 0,00000 |
| 55 | ⊞**IT Italy** | **23314,32800** | **1583,01700** | **30516700,00000** | **23295,88700** |
| 56 | ⊞Mediterranean Sea | 23314,32800 | 1583,01700 | 30516300,00000 | 23295,88700 |
| 57 | ⊞North Eastern Atlantic Ocean | 0,00000 | 0,00000 | 400,00000 | 0,00000 |

**Figure II-24 : Exemple, sous Microsoft Excel, de tableau croisé à trois dimensions : NUTS et Bassins versants en lignes, mesures en colonnes (Source : [Milego,2012])**

Les différents niveaux de la hiérarchie spatiale sont référencés en lignes. Si nous prenons l'exemple de la France, nous pouvons constater que deux membres de la dimension « bassins versants » sont accessibles. En dépliant la ligne « FR France », nous atteindrons le second niveau de la dimension NUTS, qui sera automatiquement subdivisé par le premier niveau des bassins versants. Ce manque de clarté est inhérent au célèbre tableur de Microsoft, mais il ne s'agit pas ici d'un véritable outil décisionnel.

### Diagramme

Dans les systèmes décisionnels (entre autres), les diagrammes sont un moyen de synthétiser les données quantitatives référencées dans le tableau, pour en faciliter l'analyse. Ils offrent une représentation visuelle des informations pour en permettre une interprétation plus aisée et plus rapide mais cependant simplifiée.

Il existe un très grand nombre de types de diagrammes [Visual Literacy, 2011], parmi lesquels on peut citer les plus usuels. Tout d'abord le diagramme circulaire, plus communément nommé « camembert », permet de résumer un ensemble de données nominales* ou les différentes parties d'un ensemble. La surface de chaque segment représente la proportion de chacun par rapport à leur somme totale.

Le diagramme à barres (ou à bandes) sert à représenter des données nominales ou numériques*, groupées par intervalles de classes. La valeur représentée est montrée par la longueur de ses barres : plus elle est longue, plus la valeur est élevée. Il faut le différencier de l'histogramme pour deux aspects : premièrement, ce dernier est généralement utilisé pour résumer des données discrètes, secondement, la fréquence est mesurée par la surface de ses colonnes.

Enfin, le diagramme linéaire est constitué de points, reliés entre eux, dans un repère cartésien. Il révèle clairement les tendances des données, et permet de montrer plusieurs séries à la fois, à des fins de comparaisons [Statistique Canada, 2010].

### *a. Outils*

Afin de déterminer les possibilités offertes par les systèmes décisionnels du marché pour représenter les données, cette section propose un comparatif des solutions *open source* actuelles d'analyse. Elle s'attardera aussi sur l'ergonomie et la maniabilité de ces applications.

L'offre étant pléthorique, les logiciels ont d'abord été sélectionnés parmi ceux retenus par l'équipe de la société Smile, dans son livre blanc, paru en avril 2012 [Smile, 2012]. Il s'agit de la sixième version de cette étude, mise à jour annuellement. Nous avons complété cette première sélection par les outils que nous avons rencontré au gré de nos tests et qui semblaient significatifs, tant au niveau des fonctionnalités que de l'expérience utilisateur.

Nous focalisons sur les applications *open source* parce qu'elles sont désormais considérées comme matures. Elles sont de plus librement téléchargeables sur Internet : ceci permet donc de les essayer sans contrainte ni limitation.

#### JPivot

Parmi tous les outils pour l'édition de rapports, JPivot est sans contexte la référence. Il doit cette position notamment par le fait qu'il soit l'un des projets les plus anciens du domaine, mais aussi parce qu'il est encore très largement diffusé dans de nombreuses suites logicielles actuelles, telles que Pentaho Community Edition, JasperSoft et SpagoBI.

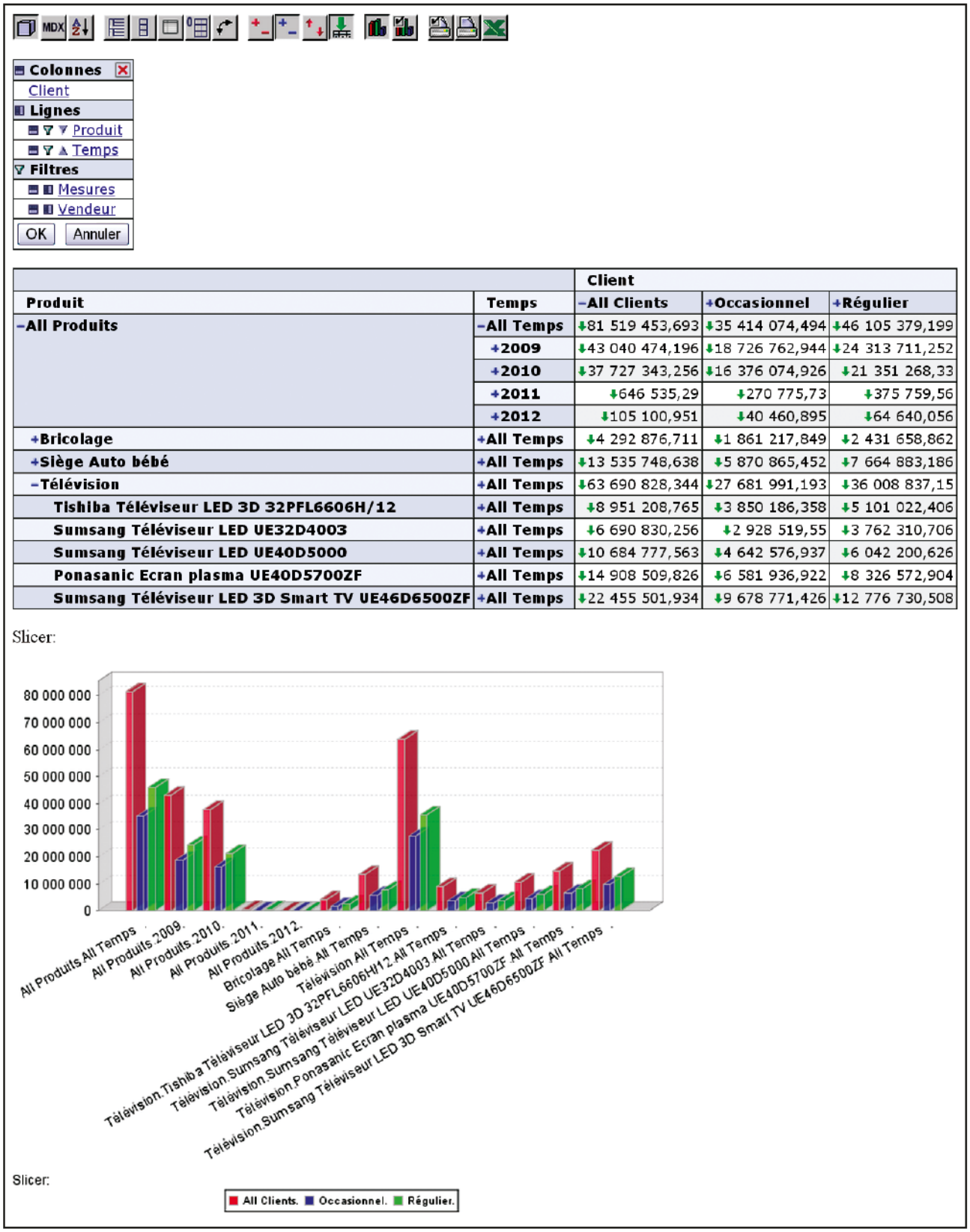
Il s'agit d'une interface Web légère, dont l'esthétique est sans fioriture et rappelle les prémices d'Internet, comme l'illustre ci-contre la figure II-25.

JPivot permet l'affichage de tableaux croisés multidimensionnels et la représentation sous forme de diagrammes de ces données. À ce sujet, il propose jusqu'à 11 graphiques différents, plus quelques déclinaisons en trois dimensions. Il autorise, en outre, l'export des données sous forme de tableur Excel ou de document PDF. Les opérations OLAP classiques sont possibles sur le tableau, via des icônes sous forme de flèches : vers le bas (forage), vers le haut (remontée), une flèche double (pivot). Le perçage (*drill trough*) est réalisable, aussi bien via le tableau que par le diagramme, donnant accès aux données du cube constituant la cellule.

L'affectation des dimensions dans les axes est réalisable par le biais de choix, exprimés par des clics sur des icônes. Quant à la sélection des membres, elle se fait par l'ouverture successive des branches

de l'arbre que constituent les dimensions, niveaux et membres. Toute manipulation doit être validée par le bouton « ok » pour provoquer la mise à jour de l'interface. De ces deux manipulations sortent une impression de lourdeur dans le processus de sélection.

Pour finir, le bouton de label « MDX » donne accès à l'éditeur de requête, en affichant, par défaut, la dernière exécutée par le logiciel. Sa modification est répercutée sur les éléments plus « graphiques » de l'interface.




| | | Client | | |
|---|---|---|---|---|
| **Produit** | **Temps** | **All Clients** | **Occasionnel** | **Régulier** |
| **All Produits** | **All Temps** | 81 519 453,693 | 35 414 074,494 | 46 105 379,199 |
| | **2009** | 43 040 474,196 | 18 726 762,944 | 24 313 711,252 |
| | **2010** | 37 727 343,256 | 16 376 074,926 | 21 351 268,33 |
| | **2011** | 646 535,29 | 270 775,73 | 375 759,56 |
| | **2012** | 105 100,951 | 40 460,895 | 64 640,056 |
| **Bricolage** | **All Temps** | 4 292 876,711 | 1 861 217,849 | 2 431 658,862 |
| **Siège Auto bébé** | **All Temps** | 13 535 748,638 | 5 870 865,452 | 7 664 883,186 |
| **Télévision** | **All Temps** | 63 690 828,344 | 27 681 991,193 | 36 008 837,15 |
| **Tishiba Téléviseur LED 3D 32PFL6606H/12** | **All Temps** | 8 951 208,765 | 3 850 186,358 | 5 101 022,406 |
| **Sumsang Téléviseur LED UE32D4003** | **All Temps** | 6 690 830,256 | 2 928 519,55 | 3 762 310,706 |
| **Sumsang Téléviseur LED UE40D5000** | **All Temps** | 10 684 777,563 | 4 642 576,937 | 6 042 200,626 |
| **Ponasanic Ecran plasma UE40D5700ZF** | **All Temps** | 14 908 509,826 | 6 581 936,922 | 8 326 572,904 |
| **Sumsang Téléviseur LED 3D Smart TV UE46D6500ZF** | **All Temps** | 22 455 501,934 | 9 678 771,426 | 12 776 730,508 |

Figure II-25 : Interface de JPivot

## Saiku

Saiku est, selon son éditeur Analytical-Labs, une « suite open source modulaire offrant un [analyseur] OLAP léger, facilement intégrable, extensible et configurable » [Analytical-Labs, 2012]. Il se présente en successeur de JPivot par la similarité de ses fonctionnalités. Il offre une représentation sous forme de tableau croisé multidimensionnel et graphique, grâce à cinq sortes de diagramme (à barres verticales, à barres verticales superposées, linéaire, circulaire et, plus surprenant, en « grille de cœur »).

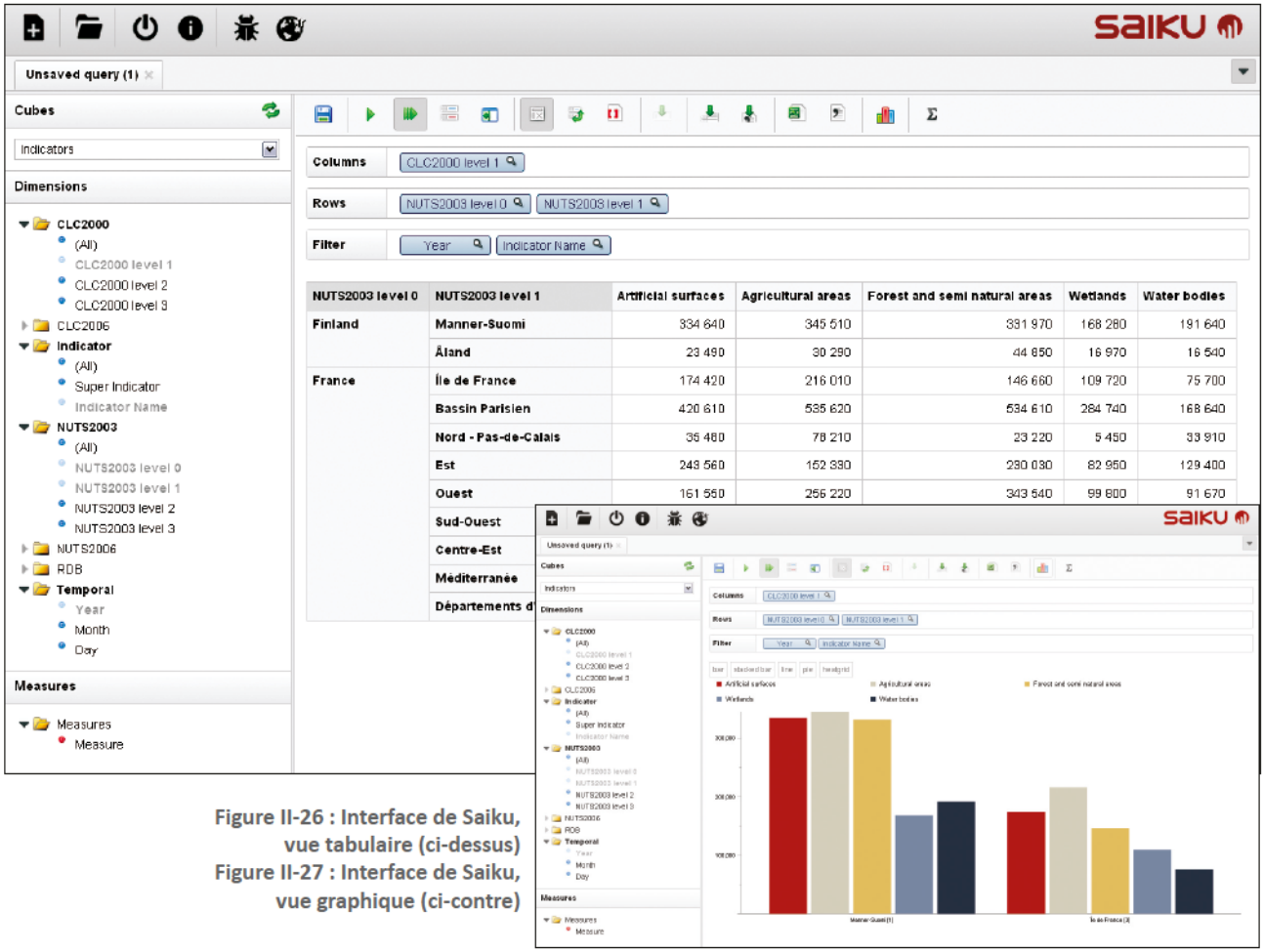



**Figure II-26 : Interface de Saiku, vue tabulaire (ci-dessus)**
**Figure II-27 : Interface de Saiku, vue graphique (ci-contre)**

Les figures II-26 et II-27 montrent les résultats d'une même requête, sous forme tabulaire et sous la forme d'un diagramme à barres. Il faut noter que l'affichage simultané des deux représentations n'est pas possible, il est nécessaire, dans ce cas, d'alterner les vues en appuyant sur un bouton, pour saisir les données dans leur ensemble.

L'export des données sous forme de fichiers Excel ou CSV est réalisable via le menu central. Le pivot est faisable, ainsi que le perçage, sur les données du tableau. Les niveaux des dimensions, placés à gauche, sont affectés aux axes grâce au glisser-déposer jusqu'à la zone idoine, ce qui permet une sélection aisée et intuitive. Quant aux membres, ils peuvent être choisis en double-cliquant sur le niveau dont ils font partie. Chaque changement sur la gestion des axes provoque l'envoi automatique de la requête au serveur OLAP, ce comportement est débrayable via un bouton.

Un système d'onglets permet la sauvegarde de plusieurs requêtes et leur restauration à des fins de comparaison ultérieure.

Sur la version que nous avons testée (2.3), l'accès à la requête MDX est possible, mais sans possibilité de modification.

## JRubik

JRubik est, comme JPivot et Saiku, un client OLAP. Il s'agit cependant d'une application Java autonome utilisant la technologie Swing : il faut l'installer pour pouvoir l'utiliser. Développé à partir des composants de JPivot, il en permet les mêmes capacités, tout en les étendant. En effet, la gestion par onglets des multiples vues permet une navigation plus fluide au travers des fonctionnalités, donnant une place à chacun des éléments (figure II-28). À gauche, un quart de l'écran permet la navigation dans les dimensions, niveaux et membres, ainsi que leur affectation dans les différents axes, par glisser-déposer ; tandis que la droite se focalise sur les résultats des requêtes par tableaux croisés, diagrammes et cartes. Les graphiques proposés sont nombreux (circulaire, linéaire, à barres verticales et horizontales, empilées ou non, affichées en deux ou trois dimensions).

Cette partie abrite également un éditeur MDX, autorisant l'édition de la requête précédemment exécutée. Cependant, les autres éléments de l'interface ne sont pas impactés par les modifications qui y sont apportées.

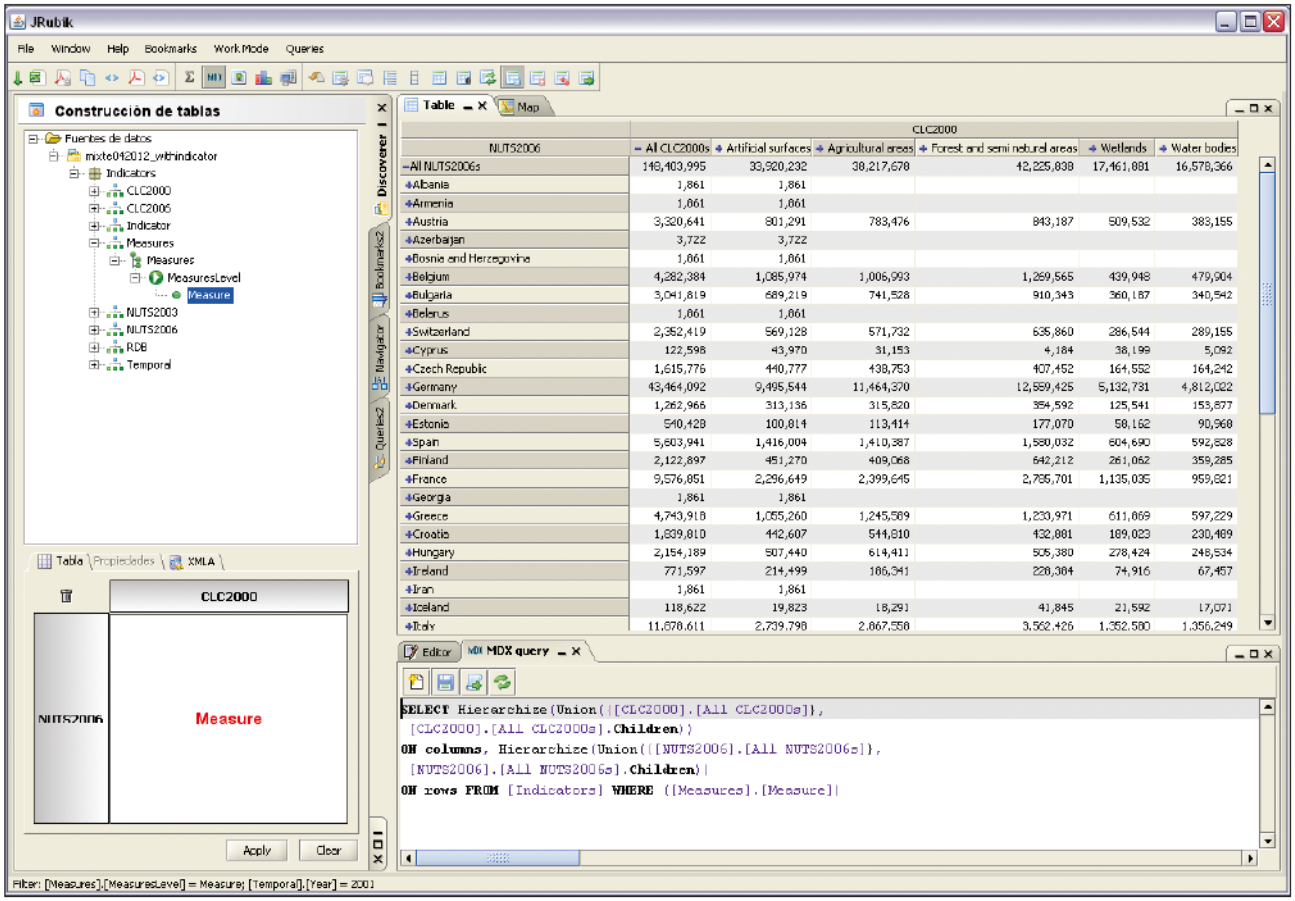



Figure II-28 : Interface de JRubik

Les opérateurs OLAP de forage, de remontée et de pivot sont présents, actionnables sur le tableau via les boutons situés au-dessus. Le perçage est lui aussi possible, après avoir choisi une cellule, un nouvel onglet s'ouvre sur la partie droite et affiche le résultat.

Le projet JRubik semble être au point mort depuis janvier 2009, aucun envoi n'a été effectué sur le dépôt de code et la version dont nous nous sommes servis pour ce test (0.9.7) a été aussi générée à la même date. L'unité de l'application n'est pas respectée : certaines parties de l'interface sont en anglais, d'autres en espagnol, quelques commandes ont un fonctionnement erratique, rendant le programme instable (graphique, carte). C'est également pour cette raison que JRubik n'a pas été conservé en tant qu'outil cartographique, bien qu'il en soit capable, selon la documentation.

## icCube

icCube, édité par MISConsulting, se place plus près du cube que les autres solutions. Cette application se comporte à la fois en tant que client, pouvant consommer les ressources d'un serveur OLAP, opérer des requêtes et représenter les résultats ; mais aussi en tant que serveur, en exploitant lui-même d'autres sources, comme des fichiers Excel, des services Web ou des bases de données, puis en les mettant à disposition via XMLA. Pour créer ces nouvelles sources, l'outil possède une interface dédiée, bien plus performante que celle proposée par Pentaho, pour produire le fichier décrivant le cube[9].

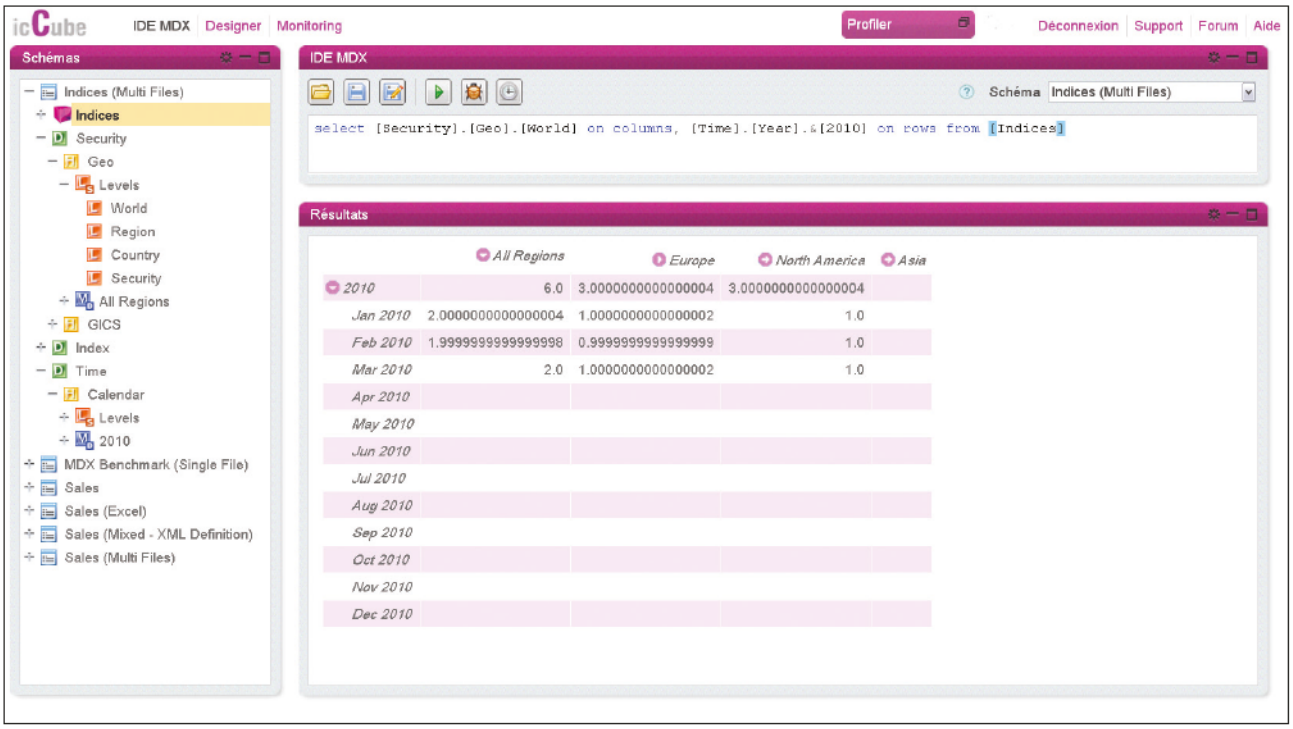



Figure II-29 : Interface d'icCube

icCube possède une interface aux couleurs acidulées (figure II-29). Les cubes, dimensions, hiérarchies et niveaux se trouvent à gauche, chaque catégorie possède sa propre image la figurant. La sélection des niveaux est possible grâce au glisser-déposer, jusqu'au bloc situé au-dessus des résultats. Il s'agit de la zone de saisie MDX : il n'y a pas d'assistant pour les axes, l'utilisateur doit rédiger lui-même la requête, l'aide apportée par le glissé-déposé n'apportant que le nom unique de l'élément sélectionné. Pour exécuter la requête, il doit finir par appuyer sur le bouton dédié. En revanche, l'outil possède un *profiler* de requête, ce qui permet de donner à l'analyste des retours sur les points améliorables de sa demande, ainsi qu'un débogueur pour l'aider à la résolution des problèmes.

Le forage et la remontée sont réalisables par clic sur le tableau croisé multidimensionnel. Par contre, le pivot n'est pas disponible, ni le perçage. De plus, aucun outil de diagramme n'est proposé.

Un autre outil, icCube Web Reporting, en licence commerciale, vient compléter ces manques. Il s'agit cependant plus d'un logiciel de *reporting* que d'analyse : sur la version de test en ligne[10], il est impossible de choisir une dimension, ou tout autre élément OLAP usuel. Il est par contre possible de modifier la disposition des éléments affichés, d'en enlever ou d'en rajouter, etc. afin d'obtenir des sorties standardisées lors de rapports automatisés.

[9] Pentaho met à disposition « Workbench » pour décrire la base ROLAP et ainsi permettre au moteur OLAP de Mondrian d'interagir avec la multidimensionnalité de la base. Cet outil est évoqué dans la Partie IV.B.2 (page 77).

[10] Démonstration en ligne : http://demo3.iccube.com/icCube/doc/tutorial/ic3report-tutorial.html?ic3demo

### Conclusion

Pour conclure cette section, il est frappant de constater que les interfaces, bien que différentes, possèdent de nombreuses similitudes. Celles-ci sont, en partie, dues à la nature même du type de logiciel dont nous faisons la comparaison : il est nécessaire de trouver la liste des cubes, puis de leurs dimensions et niveaux respectifs, et enfin, d'afficher les résultats, si ce n'est sous forme graphique, au moins sous la classique forme de tableau croisé multidimensionnel.

Si du point de vue ergonomique certaines conventions semblent établies, les choix des cubes et niveaux se trouvent à gauche, les résultats à droite ; une grande latitude est observée sur les principes de fonctionnement : ajout des éléments par boutons ou par glissé-déposé, exécution automatique ou manuelle des requêtes. Ces comportements semblent tous valables, même si certains ajoutent en ergonomie.

En exceptant le cas particulier d'icCube, tous les logiciels proposent l'aide à la génération de requêtes par la gestion des axes, ce qui apparaît comme naturel si l'on se remémore le principe de facilité d'utilisation d'OLAP : aucune compétence informatique ne doit être nécessaire pour exploiter le cube. La possibilité d'agir au niveau de la requête MDX est toutefois généralement offerte, même si cela peut poser des problèmes vis-à-vis de l'interface de génération automatique : en effet, un changement dans cette instruction devrait, en principe, mettre à jour les éléments sélectionnés sur les axes. Il s'agit cependant d'une opération complexe à réaliser.

Après avoir parcouru les caractéristiques des outils décisionnels classiques, la section suivante se concentre sur les particularités de l'informatique et des logiciels du domaine.

## *2. Particularités de l'information géographique*

La manipulation de l'information spatiale a toujours été singulière et, en général, est resté le parent pauvre des données. Alors que l'on estime que jusqu'à 80 % des données contenues dans un entrepôt de données ont une composante spatiale [Bimonte *et al.*, 2005], celles-ci ont longtemps été reléguées à un rôle figuratif, en complément des informations opérationnelles.

Cependant, depuis quelques années, la tendance s'inverse et on donne une priorité sans cesse croissante à la gestion des données géographiquement localisées. Avec l'explosion du nombre d'appareils munis d'un système de localisation (tel que le GPS, *Global Positioning System*), ou dont la position est repérable par triangulation, les applications exploitent de plus en plus et de mieux en mieux cette dimension. De nouveaux outils ont été créés pour gérer la spécificité de ce type d'information, ainsi que pour la représenter. Ce sont ces notions que cette section rappelle.

### *a. Définition et rappels*

L'information géographique associe à un objet une thématique et sa localisation sur la surface terrestre à un moment donné, ou vis-à-vis d'autres objets. On dit alors qu'elle est géoréférencée [Leobet & Merrien, 2011] :

- Soit par rapport à un système de coordonnées : coordonnées géographiques (longitude et latitude) ou coordonnées planes (mesurables par une règle sur une carte). La donnée doit alors préciser son système de référence spatiale* pour pouvoir être repérée et représentée : on obtient une carte résultant de la projection de la surface terrestre sur une surface plane ;
- Soit par rapport à des objets eux-mêmes géoréférencés : les relations topologiques sont alors utilisées pour positionner les différentes parties.

La localisation reprend la position géographique et la forme de l'objet en question.

Les données géographiques sont souvent regroupées sous forme de collections. Il faut effectivement une série de données géographiques pour constituer une carte. Les données sont de trois types :

- Les référentiels géographiques sont utiles pour le fond de carte et la présentation des autres données ;
- Les objets géographiques constituent les unités spatiales sujettes de l'étude, superposées aux référentiels ;
- Les données à proprement parler sont rattachées aux objets et nommées attributs de cet objet. Elles peuvent représenter les stocks (nombre d'habitants, répartition de types d'emplois, etc.), ou de donner des précisions sur l'objet (son nom par exemple).

La norme *Simple Features Specification* (SFS) de l'*Open GeoSpatial Consortium* (OGC) a proposé un modèle permettant la classification et l'héritage des objets de représentation. Ce modèle a évolué ensuite, au travers des normes ISO19107[11] et ISO19123[12], comme l'illustre la figure II-30 [Plumejeaud, 2011].

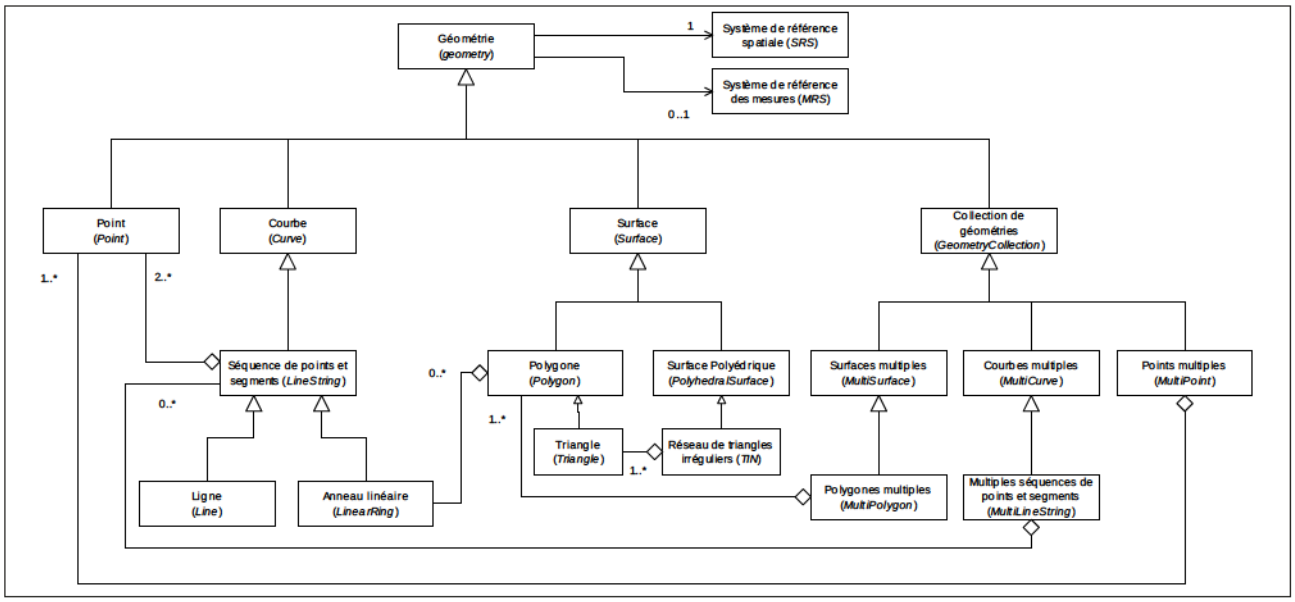



**Figure II-30 : Représentation UML des données géographiques en mode vecteur selon l'ISO19107 (Source : [Plumejeaud, 2011])**

Ce modèle est (censé être) utilisé par les bases de données spatialisées pour décrire les objets géographiques qu'elle stocke (format de représentation), les fonctions qui définissent les relations topologiques ainsi que les transformations, et fournir un référentiel pour les métadonnées [CNES 2007]. Ces dernières sont les informations décrivant les séries de données afin d'en faciliter l'inventaire, l'utilisation et la recherche. Elles sont constituées de mots-clés, de dates et thèmes auxquels elles se rattachent, d'indicateurs de qualité, etc.

### *b. Carte*

Fernand Joly définit la carte comme « une représentation géométrique plane simplifiée et conventionnelle de tout ou partie de la surface terrestre, et cela dans un rapport de similitude convenable qu'on appelle échelle » [Joly, 1976]. Il s'agit du média de visualisation par excellence des données spatiales. Elle représente sur un espace géographique défini, les phénomènes et indicateurs de ladite zone d'étude. Une carte est toujours subjective dans le sens où elle existe en tant que support de communication : le but est de faire passer un message. Celui-ci doit être concis afin d'être aisément compris par son public.

[11] Standard ISO 19107 *Geographic information – Spatial Schema*.

[12] Standard ISO 19123 *Geographic information – Schema for coverage geometry and functions*.

Les informations affichées sur les cartes suivent des conventions. Celles-ci sont les sujets de la science des signes, la sémiologie. Cette discipline, très vaste, possède de nombreuses ramifications et celles qui nous occupent, concernent les mondes graphique et visuel.

La production cartographique se trouve aussi dans le domaine statistique, et la jonction entre ces deux sciences est parfois difficile à appréhender. Il s'agit d'une opération délicate, même si assistée par l'informatique, nécessitant l'intervention humaine [Béguin & Pumain, 2003], c'est pourquoi les logiciels proposent généralement des options de personnalisation pour permettre à l'utilisateur de choisir le nombre de classes à afficher, les dégradés de couleurs ou les symboles utilisés pour représenter les informations.

Les cartographes disposent de différentes méthodes pour créer des cartes mais cinq techniques sont particulièrement usuelles [Wikipedia, 2012].

### Symboles de taille proportionnelle

Une carte à symboles proportionnels met en jeu des formes ayant des tailles différentes, directement liées à la quantité représentée. Elles sont positionnées sur la zone qu'elles décrivent, souvent au centre ou à un point stratégique, comme l'emplacement de la capitale pour un pays. Même si l'utilisation des cercles est la plus répandue, d'autres formes peuvent être utilisées, comme des carrés ou encore des pictogrammes. Pour l'affichage de plusieurs informations, on a alors plutôt recours à des diagrammes circulaires, à barres ou à crêtes de coq. La figure II-31 illustre ces différentes représentations.

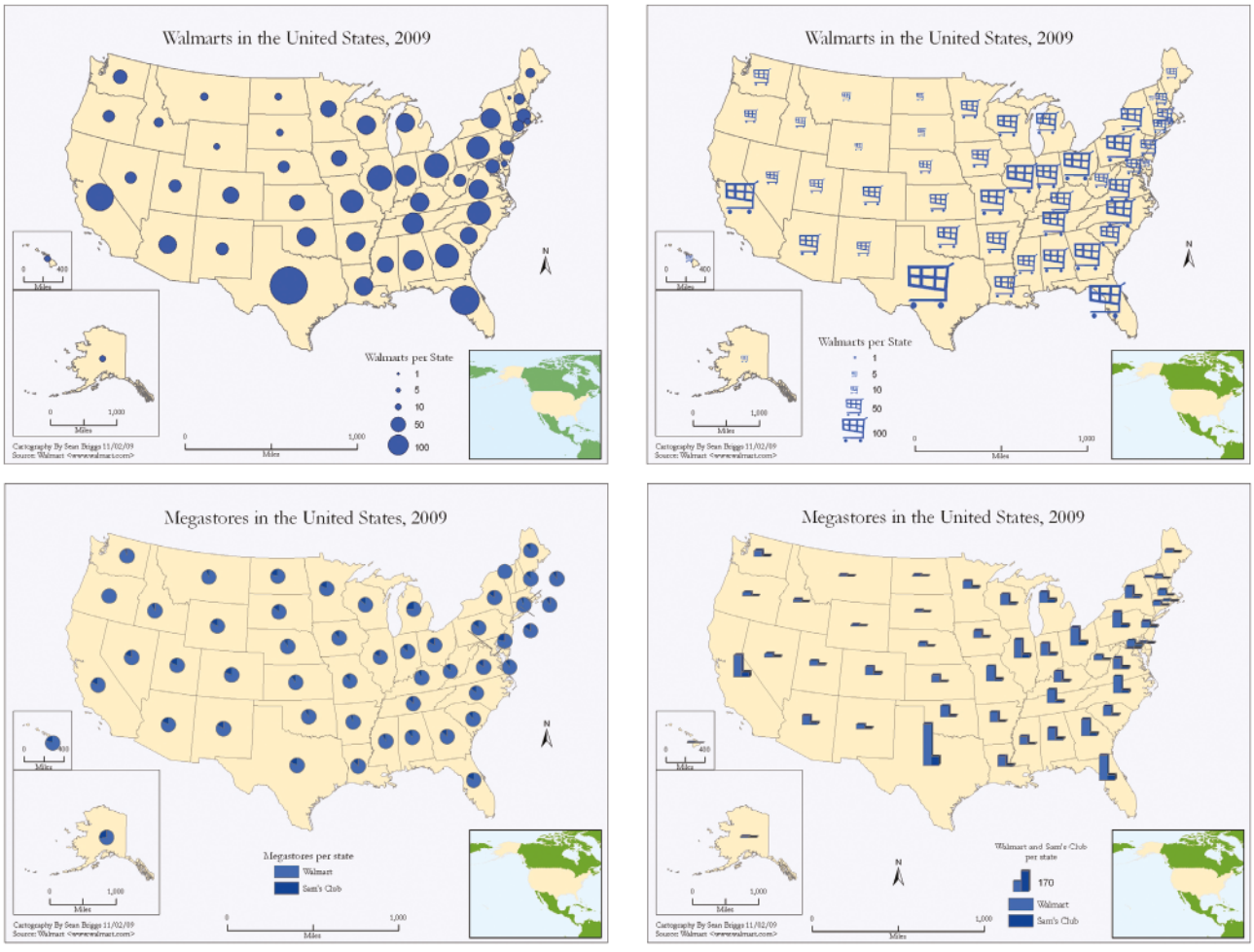



Figure II-31 : Cartes à symboles : cercles (haut gauche), pictogrammes (haut droite), diagrammes circulaires (bas gauche) et à barres verticales (bas droite) (Source : [Briggs, 2009])

## Choroplèthe

Les cartes choroplèthes présentent les données sous forme de régions colorées. Il en résulte des aplats de couleurs nuancées (souvent un dégradé) ou différentes selon les valeurs. Les données représentées sont généralement le résultat d'une division (ratio) : la densité de population, le taux de pollution d'une rivière, etc.

Concernant l'utilisation des couleurs, celle-ci est dépendante des données. Plusieurs cas de figures existent, mais nous allons nous concentrer sur les deux dominantes.

Le recours à une seule teinte est le plus courant : elle peut être déclinée du plus foncé au plus clair pour respectivement indiquer de la valeur la plus grande à la plus petite.

L'usage de plage de couleurs bipolaires, par exemple du bleu au rouge, en passant par le blanc, montre le changement de signe des valeurs : négatives en bleu, positives en rouge, nulles en blanc. Cependant, dans certains cas, la couleur la plus « favorable » de l'indice pourra être représentée par le vert, et la moins favorable par le rouge, notamment dans le cas de l'espérance de vie.

La figure II-32 illustre les utilisations « simple teinte » à gauche et « bipolaire inversée » à droite.

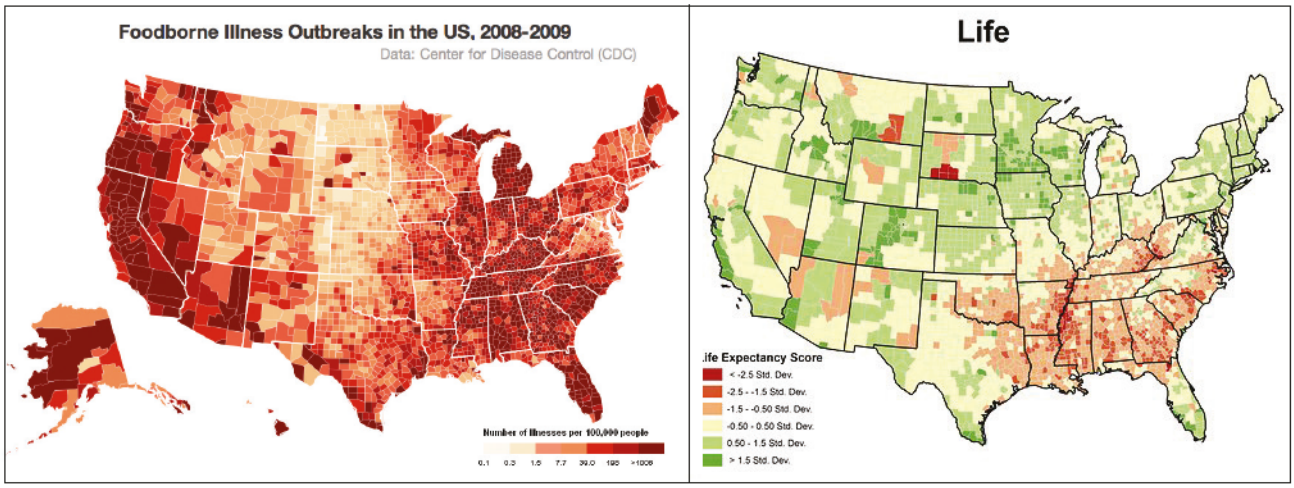



**Figure II-32 : Cartes choroplèthes : plage simple teinte (à gauche) et plage bipolaire inversée (à droite) (Sources : [Shana, 2012] & [Johnston, 2011])**

Les valeurs calculées des ratios sont distribuées en classes d'équivalence. Ceci permet de répartir les unités spatiales constituant l'aire d'étude (le contexte) dans chacune de ces classes, en lui attribuant par la même occasion une couleur. On trouve essentiellement deux types de distributions :

- Distribution des ratios par quantiles : les classes sont bornées par des points pris à des intervalles réguliers d'une fonction de répartition de telle sorte qu'elles possèdent toutes le même effectif d'unités. Cette fonction est basée sur une variable qui peut être le ratio, le numérateur ou le dénominateur ;
- Distribution des déviations : les valeurs d'une déviation absolue constituent un intervalle. Celui-ci est divisé en sous-intervalles. La progression pour passer d'un intervalle à un autre peut être arithmétique ou géométrique [Poulenard, 2011].

### Isoplèthe

Cette représentation est basée sur des contours de zones, décrivant des phénomènes continus, comme par exemple les courbes barométriques en météorologie, ou l'élévation des terrains en topographie.

### À points

La carte à points fait correspondre un phénomène à un point. Ce dernier peut en représenter une à *n* occurrences. Elle est utilisée pour dégager les tendances spatiales dudit phénomène. La carte à point la plus connue est celle du Docteur Snow, créée en 1854, référençant les cas de choléra à Londres[13]. Elle a permis de déterminer les sources d'eau contaminées.

### Dasymétrique

À mi-chemin des cartes isoplèthes et choroplèthes, les cartes dasymétriques permettent de représenter des phénomènes continus par des zones colorées avec plus d'exactitude (les valeurs exceptionnelles apparaissent mieux) que les choroplèthes. C'est le type de représentation par excellence utilisé pour décrire les types de climat (océanique, continental, etc.) ou la densité de population sur un territoire.

Les deux catégories les plus usuelles sont les cartes à « symboles proportionnels » et celles dites « choroplèthes ».

#### *c. Outils*

Dans le but de nous familiariser avec les outils de production cartographique, nous avons procédé à l'essai de deux logiciels. Tout d'abord, HyperAtlas, dans sa dernière version, outil développé par l'équipe STEAMER, dont nous avons dressé les grands traits dans le contexte de ce mémoire. Ensuite, Map4Decision, dont nous avions étudié les capacités dans notre rapport TEST [Tranchant, 2011]. Durant cette étude, nous focaliserons sur le rendu de cartes et les options offertes pour leur personnalisation.

### HyperAtlas

Bien qu'il s'agisse d'un logiciel qui ne fasse pas partie du domaine de l'informatique décisionnelle parmi ceux gardés pour cette étude, il est intéressant de voir ce dont cet outil dédié à l'analyse territoriale multi-scalaire est capable concernant la génération de cartes. Cette étude est même cruciale car HyperAtlas étant un projet de l'équipe STEAMER, il est certainement possible de réutiliser une partie des composants cartographiques.

L'ATM est une méthode de recherche proposée par Liliane Lizzi et Claude Grasland. Celle-ci repose sur l'hypothèse selon laquelle « l'étude d'une unité territoriale n'a de sens que comparée à ses voisines et mise en perspective à différentes échelles géographiques, en prenant en considération à la fois les caractéristiques des territoires voisins et les différentes entités territoriales de plus ou moins grandes dimensions auxquelles elle appartient » [Lizzi & Grasland, 2004]. HyperAtlas est donc entièrement tourné vers cette analyse et propose des outils spécifiques. De nombreuses cartes sont éditables, c'est sur ce point que nous nous concentrerons. Mais bien d'autres représentations sont possibles, notamment sous forme graphique, pour démontrer les inégalités du territoire : la courbe de Lorenz et l'autocorrélation spatiale n'en sont que deux exemples [Le Rubrus, 2011].

[13] Carte à points du Docteur Snow : http://serc.carleton.edu/eyesinthesky2/week7/intro_tabular.html

La carte à symboles de tailles proportionnelles représente l'indicateur choisi sous l'aspect de disques (figure II-33 à gauche). Le ratio des deux indicateurs choisis est exprimé par le biais d'une carte choroplèthe (figure II-33 à droite).

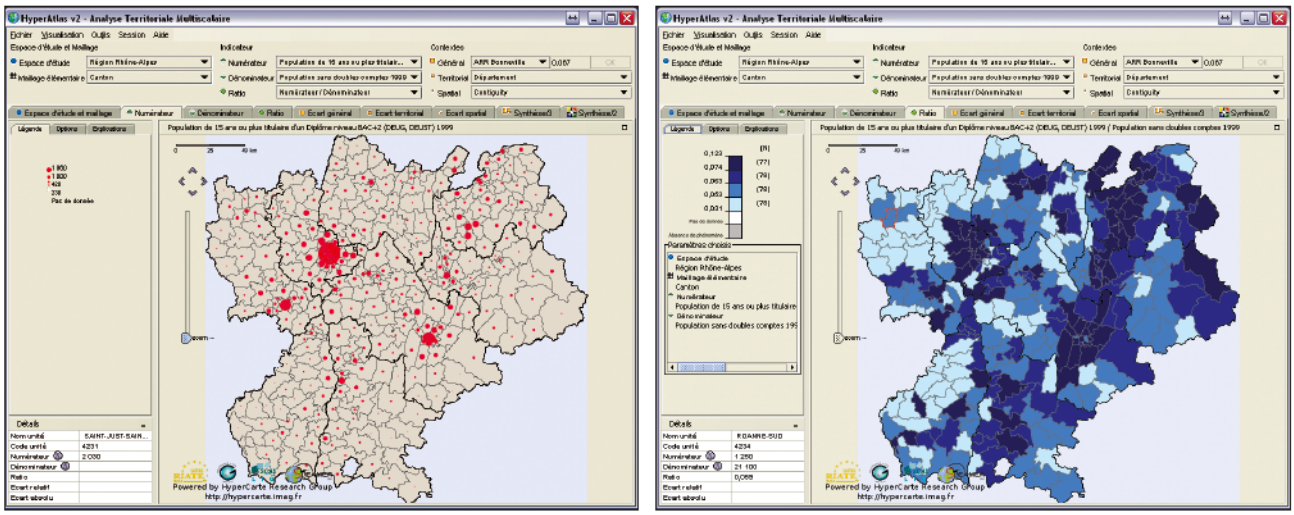


**Figure II-33 : Cartes à symboles de tailles proportionnelles (à gauche) et choroplèthe à plage simple teinte (à droite) dans HyperAtlas**

En tant qu'ATM, HyperAtlas a implanté d'autres calculs, eux aussi rendus sous forme de cartes de ce type. Il s'agit des écarts (général, territorial et spatial) et leurs synthèses (les trois écarts à la fois ou deux à deux). Ces derniers sont illustrés par la figure II-34.

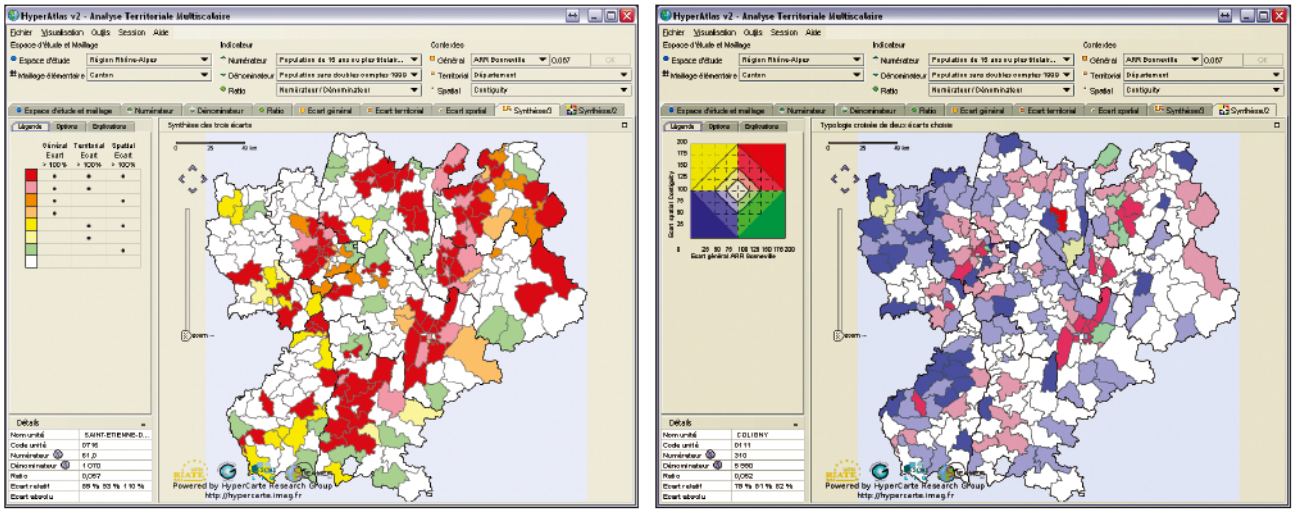


**Figure II-34 : Cartes de synthèse des trois écarts (à gauche), deux à deux (à droite). Il s'agit de cartes choroplèthes à plusieurs teintes dans HyperAtlas**

HyperAtlas est désormais très fourni avec neuf types de calculs rendus par le biais de représentations cartographiques. Les légendes sont complètes, personnalisables tant au niveau des couleurs que du nombre de classes utilisées (pour les cartes choroplèthes) et des explications sont proposées afin de parfaire l'interprétation.

Il n'intègre cependant qu'un seul type de symboles (le disque) pour la figuration d'indicateurs simples. Enfin, il est impossible d'afficher plusieurs indicateurs à la fois : les diagrammes circulaires ou à barres ne sont par exemple pas disponibles.

Pour compléter l'étude de cet outil, nous l'avons évoqué lors de la présentation du contexte de ce mémoire, HyperAtlas puise ces données dans des fichiers propriétaires non normalisés. Dès lors, il est impossible pour l'analyste d'utiliser librement les sources qu'il désire sans passer par l'outil d'administration ad hoc, HyperAdmin.

## Map4Decision

Édité par la société Intelli³, Map4Decision est une extension du logiciel JMap, de K2 Geospatial. Il permet d'ajouter une utilisation décisionnelle (en mode ROLAP et MOLAP) au serveur cartographique sur lequel il repose. C'est donc sur le duo que notre étude porte, sachant que le rendu final est pris en charge différemment selon les cas. Map4Decision est le fruit de la collaboration de l'Université Laval de Québec, de Sovar[14] et d'Intelli³.

Si la réputation de JMap n'est plus à faire, Map4Decision est une solution relativement récente. Elle apporte la possibilité de configurer un serveur OLAP en tant que source des données, conjointement à l'accès aux géométries. En effet, les données géoréférencées du cube possèdent un identifiant qui permet de faire la correspondance avec les fichiers ou tables d'informations géométriques.

La figure II-35 ci-dessous montre l'interface du logiciel. Les outils de manipulation des dimensions et mesures sont répertoriés à gauche, comme il est d'usage, ce dont nous avons pu constater lors de nos précédents essais. La figure met en avant la représentation de la population de Toronto, sous forme de carte choroplèthe. Les intervalles de couleurs indiquent donc un stock (et non pas un ratio). Les outils OLAP de forage, remontée et forage latéral, activables par le biais d'icônes en haut à gauche, sont applicables directement sur la carte. Ainsi en sélectionnant le forage et en cliquant sur l'une des régions de couleurs, un zoom sera opéré jusqu'à ce que la zone en question occupe la majorité de l'espace de visualisation disponible. Cette unité sera ensuite décomposée en sous-régions qui la constituent.

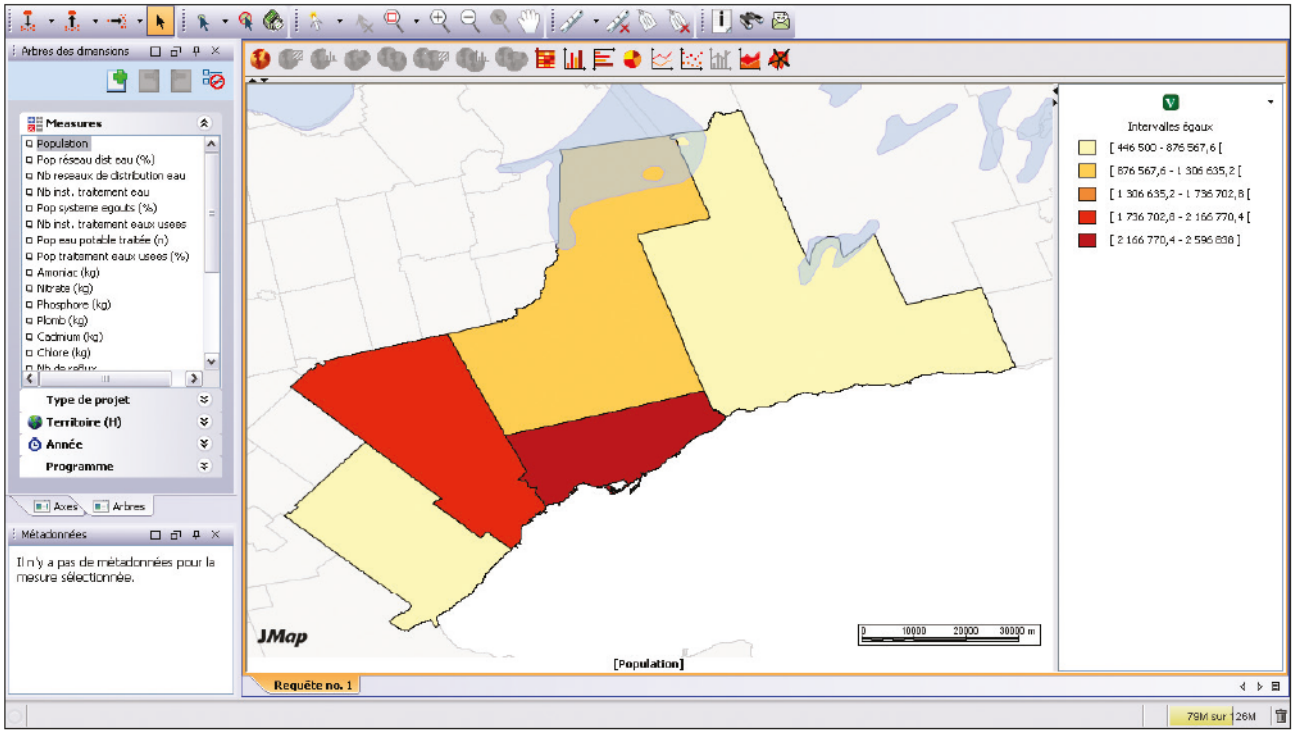



Figure II-35 : Carte choroplèthe dans Map4Decision

[14] Sovar est une société ayant pour mission de concrétiser par des applications commerciales les découvertes des chercheurs de l'Université Laval.

Map4Decision étend les possibilités de JMap, en offrant notamment d'autres représentations : des diagrammes circulaires ou à barres peuvent exprimer les mesures demandées, ce qu'illustre la partie supérieure de la figure II-36. Un mode d'affichage « multicarte » est également disponible : chacune des mesures est alors dépeinte sur sa propre carte (partie inférieure de la figure II-36). Pour finir, comme nous l'avions noté dans notre rapport TEST [Tranchant, 2011], Map4Decision est capable, grâce à l'utilisation simultanée de deux dimensions spatiales, de symboliser les flux par un diagramme origine-destination.

En laissant le pointeur sur une région, une fenêtre apparaît, donnant accès aux données exposées pour la zone en question (figure II-36 en haut).

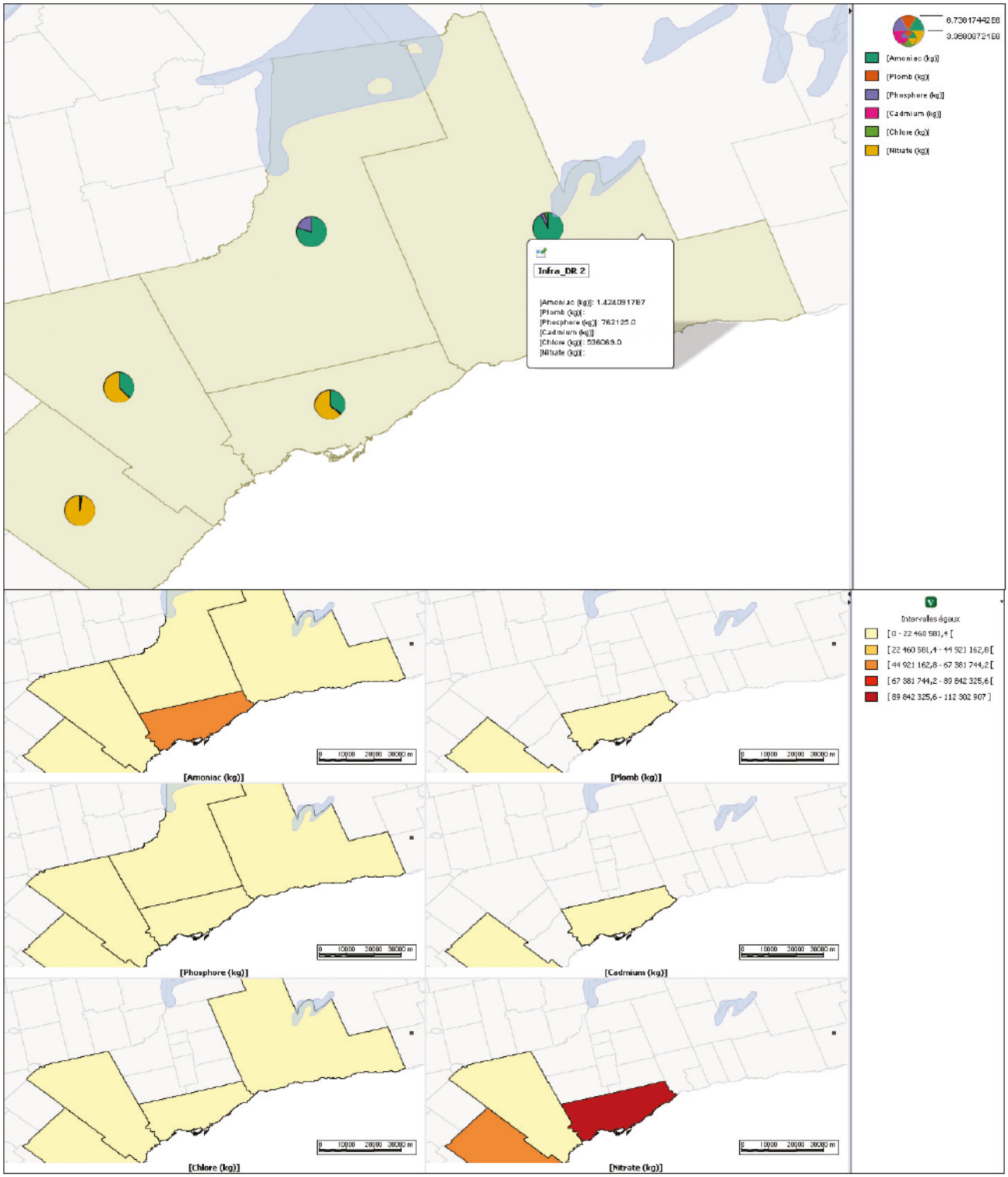



**Figure II-36 : Représentation de polluants par diagrammes circulaires (partie supérieure) et par multicartes (partie inférieure) dans Map4Decision**

Le recours à la multicarte est le plus incontestable de ce logiciel. Cette option permet une visualisation claire et simultanée de plusieurs mesures. Map4Decision est aussi le seul outil à proposer la représentation des flux ; ceci n'est cependant possible que pour une certaine configuration à la fois de l'entrepôt et des requêtes (dimensions spatiales doublées : la première désignant l'origine, la seconde, la destination). La possibilité d'éditer les principaux types de cartes est très intéressante bien qu'elle ne soit pas complètement laissée à la charge de l'utilisateur : selon les demandes, certains rendus seront activables tandis que d'autres ne le seront plus. L'utilisation de cartes choroplèthes pour l'affichage de stocks par le biais d'intervalles est assez déroutante, d'autant qu'il s'agit ici du comportement par défaut. Pour le modifier, paramétrer les vues et configurer la sémiologie du logiciel, il faut faire appel au module d'administration. À l'instar de nombreux logiciels, la mise en place de Map4Decision nécessite une personnalisation et un réglage fin, tous deux effectués par un administrateur. Ce dernier est aidé d'une documentation fournie ; l'analyste, quant à lui, peut aussi se reposer sur un guide utilisateur très complet.

### *d. Conclusion*

La comparaison de ces deux outils, bien que très différents dans leur but et aspect général, fait ressortir un point commun essentiel : la priorité est toujours donnée à la carte. Elle est systématiquement mise en avant et occupe tout l'espace disponible. Cet état de fait permet une lecture plus évidente de la carte ainsi qu'une manipulation plus aisée. Tous les outils sont placés autour de la zone de visualisation et peuvent y être appliqués : c'est le cas du zoom pour HyperAtlas, celui-ci est complété des opérateurs OLAP pour Map4Decision. Ceci a pour contrepartie d'occulter (au moins partiellement) une représentation plus conventionnelle des données, sous forme de tableau par exemple. Ce dernier est néanmoins accessible par le biais d'un onglet dans Map4Decision.

En ce concentrant essentiellement sur leurs capacités de production cartographique, nous réduisions drastiquement le champ des divergences possibles. Elles sont tout de même intéressantes à étudier car elles nous permettent d'envisager des alternatives pour notre projet.

Tout d'abord, les changements d'indicateurs ne se répercutent pas de la même façon sur la carte. Une fois la modification effectuée, il faut « l'appliquer » : un appui sur un bouton est parfois indispensable pour mettre à jour la représentation.

Ensuite, l'utilisation de la carte choroplèthe est totalement différente dans les logiciels : d'un côté, elle n'est possible que dans le cas de ratios ; de l'autre, elle est systématiquement appliquée, même sur les indicateurs simples, par le biais d'intervalles. Il n'existe effectivement pas de standardisation dans ce domaine [Palsky, 1996].

Enfin, les objectifs des représentations sont clairement définis dans HyperAtlas, notamment pour ce qui est des cartes de synthèses. Il en résulte des rendus clairement tournés vers un message : la légende y est spécifique, les couleurs coïncident parfaitement aux croisements des indicateurs. La contrepartie est un verrouillage de ce type de rapport à ce seul but : l'utilisation doit être guidée si l'on veut garder cohérence et exactitude.

Nous pouvons considérer le fait de pouvoir effectuer un grossissement de la zone étudiée comme l'un des indispensables, les outils de zoom sont des incontournables. Si l'utilisateur fait le choix d'une unité spatiale, le système doit aussi savoir la représenter centrée et à l'échelle la plus adéquate pour permettre une analyse optimale.

Offrir plusieurs types de représentation semble un minimum pour la construction de la carte, ainsi que quelques éléments de personnalisation, tels que le nombre de classes ou le style de symboles.

## C. Synthèse

Cet état de l'art a d'abord récapitulé les principes de l'informatique décisionnelle par la description de ses possibilités. Nous avons pu constater que celle-ci reposait sur une architecture spécifique : les concepts impliqués ont été décrits ainsi que leurs conséquences techniques. Une attention particulière a été apportée à l'entrepôt de données ainsi qu'à l'OLAP et à sa déclinaison spatiale.

L'entrepôt sera bâti grâce à un système de gestion de base de données relationnel. Le but est de faciliter la maintenance corrective et évolutive de la structure en s'affranchissant des barrières techniques. De plus, le SQL est un langage largement connu des informaticiens. Pour finir, il n'existe pas encore, à l'heure où nous écrivons ce mémoire, de SGBD multidimensionnel acceptant des attributs spatiaux. En conclusion, l'approche ROLAP s'impose.

Il nous faut donc ensuite opter pour le modèle qui correspondra le mieux à notre problématique. Nous devons faire cohabiter des données spatiales et attributaires, tout en orientant nos structures vers des performances maximales. Nous avons vu que le modèle étoile est le plus efficace par un recours limité aux jointures. Cependant si les volumineuses informations décrivant les formes devaient être stockées en base, il serait préférable de ne pas les doublonner afin de ne pas la surcharger. Le modèle mixte semble alors le meilleur compromis.

Adossé à cet entrepôt, un serveur SOLAP permettra l'accès aux informations, sous forme multidimensionnelle, par le biais de *XML for Analysis* et du langage MDX.

Enfin, en tant que dernière maille de la chaîne, un outil devra donner la possibilité à l'utilisateur d'interroger l'entrepôt et d'obtenir un rendu des résultats sous plusieurs formes.

Cette partie a ensuite proposé un récapitulatif des principales manières de représenter ces résultats. Outre le tableau croisé, référence des systèmes décisionnels, l'étude de logiciels du domaine a montré qu'il existe quelques graphiques plus communément rencontrés, des « standards ». C'est le cas des diagrammes à barres ou linéaires, par exemple.

Pour finir, nous nous sommes penchés sur la problématique des données géoréférencées. Cette particularité leur permet d'être replacées spatialement, les unes par rapport aux autres. Il s'agit d'un moyen supplémentaire pour les représenter, ainsi que pour les comparer : il est plus rapide de saisir le message d'une carte mettant en jeu des disques de tailles différentes selon leur valeur que de lire un tableau avec pourtant les mêmes données. Nous avons donc présenté les grandes familles de cartes et, au travers de solutions qui nous servent de référence, découvert les moyens de les mettre en œuvre. L'expérimentation de ces outils nous a permis d'apprendre les représentations les plus répandues, les avantages et inconvénients de chacune, les points d'améliorations potentiels.

# Partie III. Démarche et proposition

**Le sujet énoncé originellement par l'équipe STEAMER, du laboratoire d'informatique de Grenoble, met l'accent sur l'exploitation du concept et de la technologie du traitement spatial et analytique en ligne à des fins de visualisations, notamment, la génération cartographique, pierre angulaire des systèmes géographiques.**

Inscrite dans le sens de la directive INSPIRE, notre proposition tient compte des contraintes liées à l'interopérabilité et la diffusion des données, en mettant à disposition les données du système, que ce soit sous forme de cartes ou de tableaux.

Cette partie est composée de quatre chapitres. Tout d'abord, un récapitulatif des besoins est établi afin de cerner les tenants et le périmètre de notre projet. Ensuite, notre proposition est exposée, tirée de ces besoins. Suivent l'architecture retenue pour y répondre puis la démarche que nous adoptons pour mener à bien notre mission. Le chapitre suivant est consacré aux décisions préalables à tout développement qu'il a fallu entériner. Enfin, la feuille de route esquisse nos différents axes de travail, constituant la trame de nos réalisations.

## *A. Besoins*

Le projet a pour vocation de tester l'utilisation des données spatiales dans le cadre de l'informatique décisionnelle. En effet, selon Gartner, l'analyse de données est la priorité des décideurs informatiques pour 2012[15] et ce, à des fins de croissance économique et de meilleure prise de décisions. Pour ce faire, l'informatique décisionnelle est l'une des voies plébiscitées car résolument tournée vers cet objectif. L'équipe STEAMER, sur la base des résultats énoncés par le professeur Yvan Bédard, de l'Université Laval de Québec, a décidé du développement d'un prototype afin d'explorer et de tester les possibilités offertes par ces technologies, tout en les appliquant aux problématiques auxquelles le laboratoire est chargé de répondre, notamment par le développement d'outils de visualisation pour aider à l'organisation et l'aménagement d'un territoire.

Le besoin exprimé est donc l'exploitation du cube de données afin de profiter des opérateurs offerts par le traitement analytique en ligne, comme le forage ou le pivot. Placé dans le cadre de la géomatique, nous pouvons bénéficier des capacités spécifiques du *Spatial OLAP*, exposées dans la section éponyme dans notre état de l'art (page 27). L'application de type « intergiciel » (*middleware*) dont il est question, doit faire le lien entre l'hypercube et un logiciel client de représentation des données. À l'instar de la version actuelle d'HyperAtlas, leur restitution doit être permise sous forme de cartes. Pour permettre ces fonctionnalités, cet « outil de visualisation » permettra l'affichage de la représentation générée et l'envoi des demandes, sous forme de requêtes.

[15] Enquête menée par Gartner auprès de 2000 CIO dans 45 pays.
Source : http://www.lemondeinformatique.fr/actualites/lire-le-gartner-pointe-les-priorites-it-des-entreprises-en-2012-47460.html

Le but final de ce projet et de l'ensemble de ces éléments est d'abord d'expérimenter les capacités de l'informatique décisionnelle en général et du SOLAP en particulier. Ensuite, il s'agit de déterminer ce qu'elles peuvent apporter à l'équipe STEAMER. Certes, les technologies utilisées ici sont éloignées des principes actuellement en usage sur HyperAtlas, mais certains pans sont peut-être réutilisables. Il sera donc question de ces éventuels apports et différences certaines vis-à-vis d'HyperAtlas.

## B. Proposition

À la connaissance de cet énoncé, nous avons étudié puis proposé une solution englobant les contraintes exprimées.

### 1. Vue globale

Ce projet couvre la partie aval de l'environnement décisionnel, de l'entrepôt où sont stockées les données selon un archétype prédéfini, à leur visualisation cartographique et tabulaire. En émergent trois grandes tâches, à l'intersection de plusieurs thèmes et environnements.

Tout d'abord, la chaîne décisionnelle, qui est composée de plusieurs éléments, décrits dans la section « informatique géodécisionnelle », en particulier, le serveur SOLAP et l'entrepôt de données. De part l'énoncé des besoins, la priorité est donnée à cette partie qui est elle-même composée de plusieurs étapes.

Ensuite, la rediffusion des informations, sous forme de services Web, doit être effectuée de manière la plus souple et logique possible, dans un contexte où il n'existe pas de standard et ce, dans le but de répondre aux contraintes de la directive INSPIRE. Entre le serveur SOLAP et ces services Web, toute une application doit être bâtie, afin d'un côté, de générer les requêtes nécessaires au questionnement du cube, grâce au langage MDX, de l'autre côté, comprendre et interpréter les demandes émanant des services exposés, puis y répondre de manière appropriée.

Enfin, un client de visualisation doit lui aussi être construit pour profiter des mécanismes et services offerts par le serveur et permettre à l'utilisateur final d'interroger l'entrepôt et d'en visualiser le contenu, sous plusieurs représentations.

Les choix des différentes technologies, stratégies et développements inhérents à chacun des aspects de notre proposition sont partiellement dépendants. Ils prennent aussi en compte une contrainte forte exprimée au commencement de notre projet : n'utiliser que des logiciels libres.

### 2. Architecture

La structure générale d'un système informatique doit être l'une des premières décisions à prendre durant la conception d'un projet. De ce choix découlent les lignes directrices du projet, les cloisonnements inhérents aux différentes pièces constituant l'application. L'architecture d'un SI est une perspective dépendante du point de vue adopté et de ce que l'on souhaite mettre en exergue. Nous évoquerons ici deux perspectives afin de définir l'urbanisme de notre environnement.

#### a. Architecture métier

D'un point de vue métier, notre projet doit mettre à disposition une interface pour exploiter les données d'un ou plusieurs entrepôts de données, afficher le résultat des requêtes et une représentation de ces informations. La figure III-1 (ci-contre) illustre cette vue métier en mettant en relief l'aspect multi-clients et centralisation des données de notre projet. En découle la nécessité de

gérer la concurrence entre les utilisateurs et de maximiser les capacités de réutilisation des parties logiques, par exemple, par l'utilisation d'un cache.

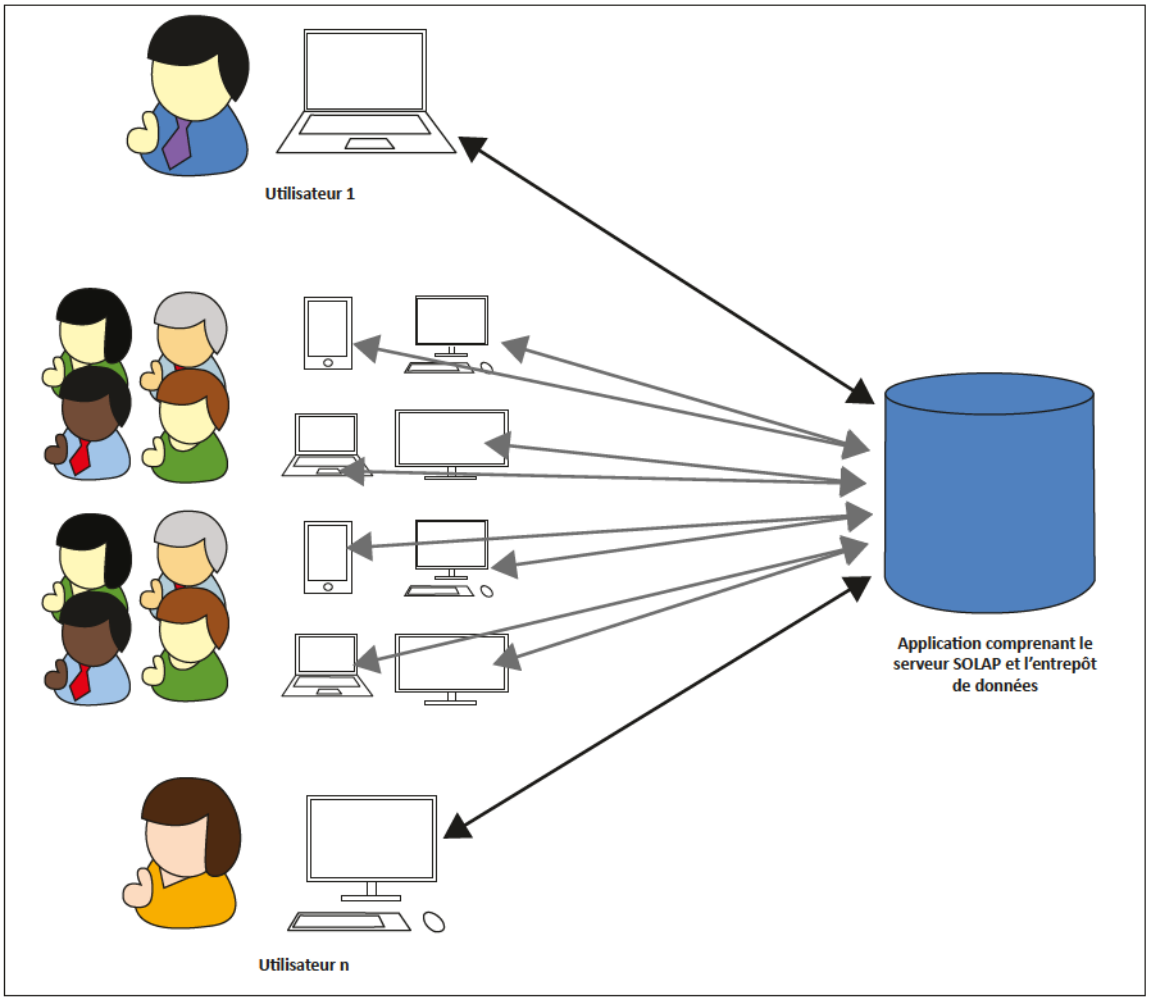



Figure III-1 : Vue métier de l'application

### *b. Architecture logicielle*

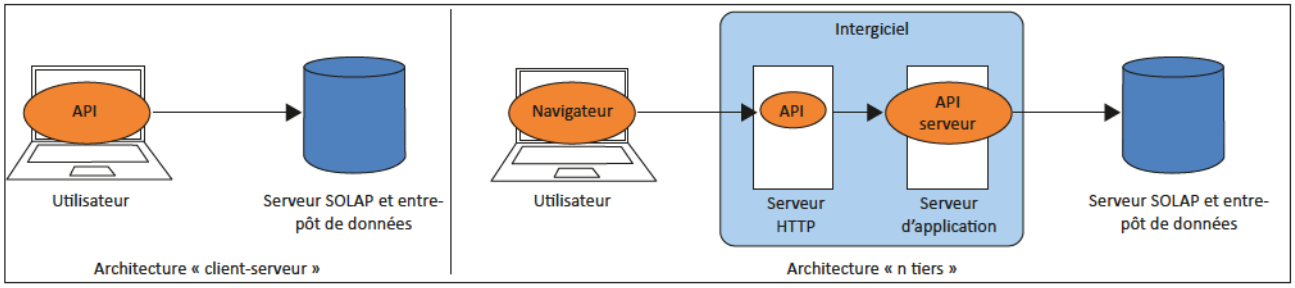



Figure III-2 : Deux architectures logicielles possibles : « client serveur » (à gauche) et « *n* tiers » (à droite)

L'architecture logicielle macroscopique, quant à elle, permet de dégager deux options.

Dans la première (nommée « client-serveur », à gauche sur la figure III-2), le client embarque tout le métier, se connectant à un serveur SOLAP par ces propres moyens, l'interroge, interprète le résultat, en génère éventuellement la carte (selon les données) et affiche le tout. Il est aisé de constater qu'il s'agit ici d'un client autonome. En effet, tous les processus de traitements des ressources communiquées par l'OLAP sont supportés par le client.

La seconde solution envisage un client léger, qui se connecte à une couche logicielle intermédiaire

de type intergiciel. Le client ne fait alors que du rendu auprès de l'utilisateur, et envoie les requêtes générées par ses manipulations. Nous sommes alors en présence d'une architecture *n* tiers. Cette approche permet de définir clairement les rôles des différents éléments : le serveur OLAP est en charge de la couche d'accès aux données (gestion de la persistance), l'intergiciel du traitement métier (traitement et transformation éventuelle des données). Le client, quant à lui, permet la présentation et l'interaction entre l'utilisateur et le *middleware*.

Nous avons choisi d'opter pour la deuxième alternative pour plusieurs raisons. Tout d'abord, il est inutile, dans cette configuration, d'installer quoique ce soit sur le poste de l'utilisateur final : il n'a besoin que d'un navigateur Web afin d'accéder au serveur en ligne. Ensuite, la partie métier n'est traitée que par un serveur dédié. Il n'y a pas de dispersion ni de mélange dans les couches applicatives. Cette segmentation permet aussi la définition de contrats entre les strates de l'environnement et limite leur couplage. Enfin, la mutualisation des efforts déployés pour l'interrogation de l'entrepôt et de la génération de cartes atteint son maximum. Ici encore, la mise en place d'un système de caches permet la réutilisation de données extraites de l'entrepôt et les dernières cartes produites. De plus, dans le cadre de la directive INSPIRE, nous nous devons de partager ces données et les traitements qui s'y rattachent, un service Web supplémentaire peut être sereinement ajouté afin d'accéder à cette exigence. Pour répondre à cette contrainte, l'élément « intergiciel » de la figure III-2 (à droite) sera réellement scindée en deux parties. La première, référencée « API » sera notre « client HyperAtlas³ » et répondra aux besoins d'interface de l'utilisateur tandis que la seconde, « API serveur », hébergera les services nécessaires au fonctionnement du client et les mettra à disposition du public : notre « serveur HyperAtlas³ » sera ainsi conforme à la directive INSPIRE.

## 3. Démarche

Le stage CNAM impose, par les conditions dans lesquelles il est exécuté, un certain nombre de contraintes. Pour une durée totale de neuf mois, une unique personne peut contribuer au projet. Le célèbre triangle de variables « Coût – Qualité – Délai » est alors réduit à une inconnue : le délai ne pouvant être prolongé, l'ajout de développeurs rendu impossible, bloquant par la même occasion le coût, seule la qualité peut varier. Cependant, comme l'avait indiqué Laurent Poulenard, il n'est pas question ici de dégrader la qualité du code, des tests ou de la documention, ni d'éluder des étapes des processus de vie du logiciel, il s'agit d'ajuster le nombre de fonctionnalités offertes par le programme [Poulenard, 2011].

Afin d'éviter l'effet de tunnel, aléa majeur des grands projets, et le risque de ne pas livrer une application viable à temps, un état des lieux des tâches à effectuer, leur ordonnancement et un maquetage ont été effectués en amont, ainsi que des points de suivi régulier avec notre tuteur.

### a. Découpage en tâches

La liste des tâches nécessaires à l'obtention du produit dans un état satisfaisant a été établie (tableau III-1) après un processus de découpage en unités. Elle contient tout d'abord une description succincte de ladite tâche, l'objectif auquel elle doit répondre. Ensuite, les étapes prévues et livrables escomptés, que nous considèrerons comme nos jalons, ainsi que le but auquel elle répond. La dernière colonne renvoie à la section à laquelle elle est développée dans ce mémoire.

Même si elles sont nécessairement ordonnées, ces tâches ne doivent pas toujours être entièrement complétées pour passer à la suivante. Dans le cas contraire, nous ne pourrions pas avoir la possibilité de présenter une démonstration à chaque fin d'itération. Dans le cas où elles n'ont pas été ordonnancées par une contrainte purement technique (comme définir le serveur d'application avant le développement à proprement parler), elles ont été classées selon la priorité exprimée par notre tuteur.

| # | Description | | |
|---|---|---|---|
| | **Étapes et livrables** | **But** | **Étape de réalisation** |
| 1 | Choix d'un système de gestion de bases de données spatiales pour l'hébergement de l'entrepôt | | |
| | - Comparaison des SGBD ;<br>- Installation du logiciel et de son extension spatiale. | Stockage physique des données | Décisions préalables (page 59). |
| 2 | Choix d'un serveur *Spatial OLAP* et de ses méthodes d'accès | | |
| | - Etude des outils SOLAP ;<br>- Détermination de la méthode de récupération des objets géographiques ;<br>- Installation et paramétrage de l'outil retenu. | Interrogation de l'entrepôt | Décisions préalables (page 62) et Chaîne géodécisionnelle (page 77) |
| 3 | Détermination du langage et du serveur d'application | | |
| | - Décision sur le langage à utiliser ;<br>- Installation du serveur d'application en charge d'héberger le logiciel. | Accès à l'applicatif | Décisions préalables (page63) |
| 4 | Modélisation du cube | | |
| | - Etude des expérimentations effectuées par d'autres laboratoires ;<br>- Réflexion sur les perspectives à manipuler ;<br>- Génération du modèle ;<br>- Création d'un jeu de données de test. | Stockage logique des données | Chaîne géodécisionnelle (page 71) |
| 5 | Découpage du serveur en modules | | |
| | - Liste des modules nécessaires à la construction de l'application. | Fonctionnement global de l'application | Serveur HyperAtlas³ (page 79) |
| 6 | Choix du service Web et de ces méthodes | | |
| | - Comparatif des technologies relatives aux services Web ;<br>- Interfaces et WSDL. | Accès aux services | Serveur HyperAtlas³ (page 81) |
| 7 | Développement d'un module de connexion XMLA | | |
| | - Etat de l'art sur la norme XMLA ;<br>- Etude des outils existants ;<br>- Développement du module XMLA. | Connexion et interrogation du serveur SOLAP | Serveur HyperAtlas³ (page 84) |
| 8 | Moteur MDX pour l'interrogation du serveur SOLAP | | |
| | - Etat de l'art sur le langage MDX ;<br>- Module de génération de requêtes MDX. | Gestion de requêtes MDX | Serveur HyperAtlas³ (page 88) |
| 9 | Générateur cartographique | | |
| | - Localisation et extraction des sources à partir d'HyperAtlas ;<br>- Gestion par Maven de la fonctionnalité ;<br>- Adaptation à notre contexte ;<br>- Ajout de nouvelles représentations. | Génération de cartes | Serveur HyperAtlas³ (page 96) |
| 10 | Finalisation du serveur | | |
| | - Création d'un module central et de sous-modules transversaux contenant les objets communs ;<br>- Serveur fonctionnel. | Assemblage des modules | Serveur HyperAtlas³ (page 86) |
| 11 | Interface utilisateur pour l'accès via un client léger | | |
| | - Connexion au serveur ;<br>- Interface simple pour l'affichage du tableau croisé multidimensionnel ;<br>- Client (version simple). | Manipulation par l'utilisateur du cube | Client HyperAtlas³ (page 104) |

| # | Description | | |
|---|---|---|---|
| | **Étapes et livrables** | **But** | **Étape de réalisation** |
| 12 | Aide pour l'utilisateur | | |
| | - Rédaction d'une aide en ligne. | Assistance utilisateur | Client HyperAtlas³ (page 116) |
| 13 | Amélioration du client vis-à-vis des opérateurs OLAP | | |
| | - Pivot, forage, remontée ;<br>- Contrôle de navigation, contraintes MDX ;<br>- Client amélioré OLAP. | Navigation sécurisée dans les dimensions | Client HyperAtlas³ (page 110) |
| 14 | Evolution du client vis-à-vis des opérateurs SOLAP | | |
| | - Affichage de la carte et de la légende ;<br>- Ajout des options de modification de la représentation de la carte ;<br>- Ajout des options de grossissement ;<br>- Client géodécisionnel. | Exploitation de la dimension spatiale | Client HyperAtlas³ (page 114) |
| 15 | Amélioration de l'interface du client | | |
| | - Options d'export pour le tableau ;<br>- Export de la carte. | Complément des options existantes | Client HyperAtlas3 |

**Tableau III-1 : Liste des tâches déterminées pour mener à bien le projet**

### *b. Méthodologie*

Une fois la décomposition de l'application effectuée, nous avons pu débuter les premières actions visant à la livraison du produit final.

Nous avons adopté une démarche agile s'inspirant de plusieurs méthodes. Bien que celle-ci soit généralement plus adaptée à une équipe[16], quelques bonnes pratiques sont toujours réutilisables. Chaque méthode ayant son vocabulaire, généralement exprimé en termes anglo-saxons, nous allons tenter d'expliquer et de traduire (lorsque cela est possible) ces concepts, sur chacune d'elle.

L'étape initiale, empruntée à Scrum[17], consiste à dresser une liste des grandes fonctionnalités attendues (des *epics*), redécoupées en scénarios utilisateurs (les *stories*, listées dans le *product backlog*) auxquelles nous avons ajouté les processus techniques. Ces scénarios sont eux-mêmes subdivisés en tâches élémentaires.

À chaque début de cycle, nous avons ensuite sélectionné, parmi cette liste, certaines des tâches afin de constituer notre objectif d'itération (nommée *sprint*, liste elle-même nommée *sprint backlog*).

L'itération, d'une durée de deux semaines, est composée d'une séquence de cinq étapes, empruntées à la méthode de développement rapide d'applications (dont le sigle anglais RAD désigne *Rapid Application Development*) :

- Préparation, définition des objectifs ;
- Cadrage et balisage du périmètre, recherches relatives aux notions en jeu;
- Conception, modélisation et spécifications, dessin de maquettes ;
- Construction du prototype par son développement (code métier et tests unitaires) et de sa documentation (JavaDoc, site d'explication) ;
- Déploiement de la version finalisée de l'application, en l'état, afin de vérifier la bonne prise en compte des objectifs exprimés, dans le cadre défini.

[16] Les méthodes agiles sont basées sur des valeurs, dont l'une d'elle est l'équipe et les individus qui la composent (cf. le manifeste agile http://www.agilemanifesto.org/iso/fr/).

[17] Site officiel de la méthode Scrum : http://www.scrum.org/Scrum-Guides.

Notre objectif majeur était d'obtenir le plus rapidement possible un résultat visuellement significatif pour l'utilisateur final. Dans cette optique, il nous a fallu joindre les deux extrémités : de l'entrepôt de données (et du serveur SOLAP) à la visualisation, en produisant, dans un premier temps, le minimum nécessaire à la communication entre les différentes parties, comme illustré par le trait orange sur la figure III-3 ci-dessous. L'approche descendante (*top-down*) nous a permis, à partir d'une vue d'ensemble et d'un nombre restreint de composants, de les décomposer et de les affiner au fur et à mesure des itérations.

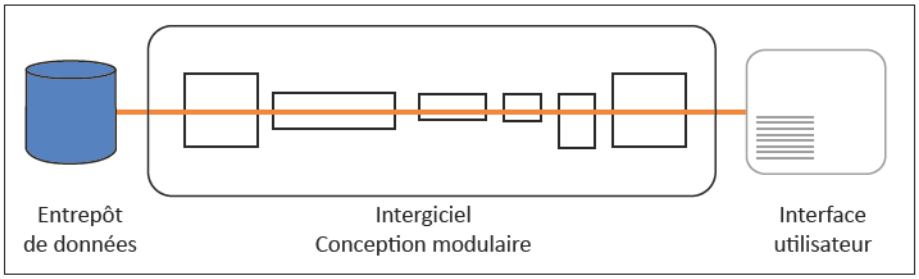



**Figure III-3 : Vue logicielle après les premières itérations**

Couplée à une conception modulaire, la démarche itérative se prête parfaitement à l'enrichissement progressif des différents composants (en violet, sur la figure III-4), offrant de nouvelles fonctionnalités, aux modules immédiatement alentours, et, *in fine*, à l'interface finale.

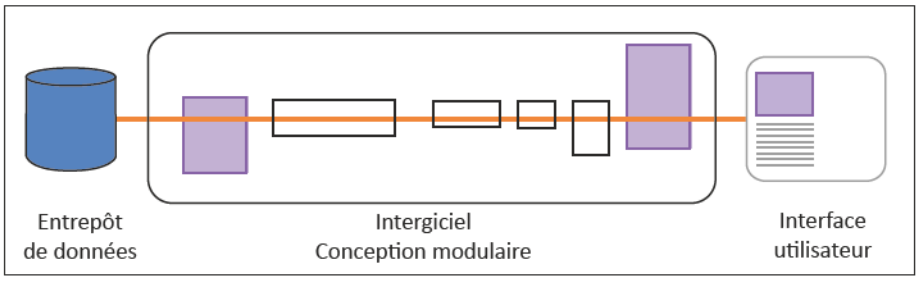



**Figure III-4 : Vue logicielle : modules enrichis**

Enfin, les modules peuvent ensuite être subdivisés (ou regroupés), afin de les spécialiser (généraliser) sur une fonction précise, tout en les rendant les plus abstraites possibles (cas représenté en vert par la figure III-5).

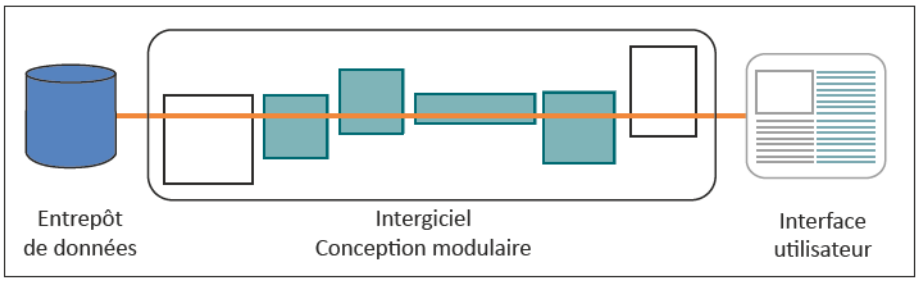


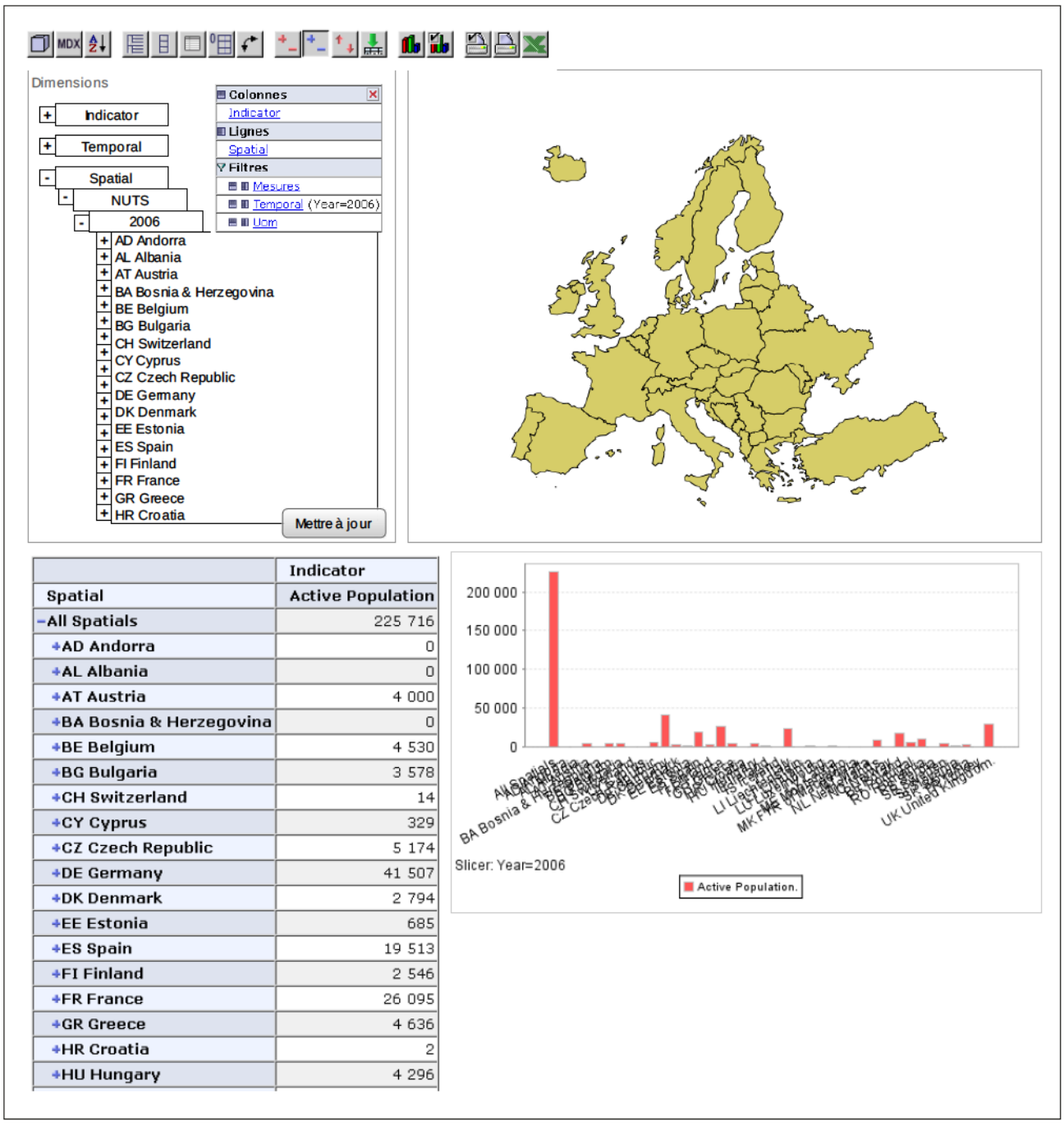

**Figure III-5 : Vue logicielle : modules spécialisés**

À l'instar de la démarche agile, la programmation modulaire est l'une des bases de la construction de notre projet. En effet, le logiciel est conçu comme l'assemblage de différents modules, chacun ayant une collection de fonctions proches (unité du module) et dont le périmètre est précisément défini. Cette décomposition donne une meilleure compréhension du fonctionnement global du programme et une facilité dans le développement : il est plus pratique de manipuler plusieurs petits concepts qu'un seul plus vaste. La modularité permet un couplage faible entre les modules et impose entre eux des interfaces explicitement déclarées. En outre, une grande abstraction s'impose pour plus de

réutilisabilité des divers modules. Si cette approche est un standard de fait, qui plus est, dans les langages de paradigme objet comme Java, elle est exploitée à son paroxysme dans notre application grâce, notamment, à l'utilisation généralisée de Maven (se référer à la section éponyme IV.A.1) et par la réutilisation des composants de génération cartographique d'HyperAtlas.

### *c. Maquettes et prototypes*

Dans le but de concrétiser les attentes de chacune des parties, nous avons opté pour la modélisation par maquettes de l'interface du client. Les premières, composées d'éléments d'autres applications, nous ont permis de cerner plus précisément quelles allaient être les contraintes techniques et demandes des utilisateurs en termes d'exploitation des données (figure III-6).




| | Indicator |
|---|---|
| Spatial | Active Population |
| All Spatials | 225 716 |
| AD Andorra | 0 |
| AL Albania | 0 |
| AT Austria | 4 000 |
| BA Bosnia & Herzegovina | 0 |
| BE Belgium | 4 530 |
| BG Bulgaria | 3 578 |
| CH Switzerland | 14 |
| CY Cyprus | 329 |
| CZ Czech Republic | 5 174 |
| DE Germany | 41 507 |
| DK Denmark | 2 794 |
| EE Estonia | 685 |
| ES Spain | 19 513 |
| FI Finland | 2 546 |
| FR France | 26 095 |
| GR Greece | 4 636 |
| HR Croatia | 2 |
| HU Hungary | 4 296 |

Figure III-6 : Première maquette de l'interface manipulable par le client

Visuellement, nous pouvons isoler de cette esquisse quatre parties :

- En haut à gauche, les outils de manipulation des dimensions ;
- En haut à droite, la représentation cartographique ;
- En base à gauche, la reproduction tabulaire des données sélectionnées ;
- En base à droite, un graphique illustrant les informations récupérées.

Il n'y a en fait que deux ensembles : d'un côté, les outils d'interrogation et de génération de requêtes, et de l'autre, la représentation de leur réponse, sous plusieurs formes.

Les méthodes agiles prônent la mise à disposition régulière de prototypes pouvant servir de support de démonstration, en vue d'obtenir des retours des « clients ». Dès lors, les commentaires obtenus sont intégrés à une itération suivante, selon l'importance qui leur est donnée. Ainsi, les premiers prototypes manipulables ont fait émerger des problèmes inhérents à la manipulation des dimensions ainsi qu'à leur représentation tabulaire qu'il était difficile de prévoir durant les périodes de spécifications. Heureusement, ces difficultés ont pu être détectées tôt, donc leur résolution en a été plus aisée.

Pour accélérer encore le processus, nous avons profité des retours d'expérience des utilisateurs notamment lors des réunions de l'équipe.

## C. Décisions préalables

Après avoir analysé les besoins, dessiné grossièrement les contours de notre proposition et dressé notre démarche, nous rentrons désormais dans une phase de décisions, qui serviront de base dans nos réalisations.

Arrêter des choix à cette étape peut être contraignant mais sans eux, il est impossible de progresser. Pour se permettre une souplesse maximale, nos partis pris doivent être assez génériques : ce sont des solutions éprouvées ou remplaçables. Dans ce dernier cas, le coût de transformation se doit d'être le plus réduit possible.

### 1. Gestion des données

Au cœur de tout environnement informatique, les données constituent la matière première et l'intérêt même de ce système. Pour un système décisionnel, l'entrepôt est l'élément critique de la chaîne. Il est aussi le premier à mettre en place. Nous avons conclu notre état de l'art par l'adoption de l'approche OLAP relationnel et du modèle mixte. Il reste cependant quelques questions à éclaircir.

#### a. Stockage des données spatiales

Le premier sujet à clarifier est de savoir comment les informations spatiales sont stockées. Nous avons étudié la particularité de ce type de données et des opérateurs qui sont nécessaires pour les extraire sous une forme exploitable par la suite. Il existe deux moyens principaux de gérer leur stockage.

Le premier est aussi le plus répandu : il s'agit d'un stockage en base de données relationnelles, dans un SGBD qui aurait été préalablement modifié afin d'accepter les données spatiales. La géométrie est enregistrée sur une ligne d'une entité et couplée à d'autres informations dont, à minima, un identifiant.

Les systèmes de gestion de bases de données relationnels les plus utilisés ont désormais presque tous une extension spatiale, plus ou moins aboutie, qui implante les recommandations édictées par l'OGC dans son document « *OpenGIS Simple Features Implementation for SQL*[18] » :

- MySQL[19] est un SGBD libre de droit édité par l'Oracle Corporation. Son module spatial ne met en place qu'une partie des instructions de l'OGC, les manques que cela occasionne éliminent d'emblée MySQL pour une utilisation poussée des capacités spatiales ;
- PostGreSQL[20], SGDB libre de droit, et son extension PostGIS[21] sont édités par le PostGreSQL Global Development Group. Les capacités spatiales offertes par PostGIS honorent les directives de l'OGC. Cette solution est très largement adoptée au LIG, ainsi que dans de nombreuses entités travaillant sur des données géoréférencées, ce qui en fait l'outil du domaine qui fait autorité ;
- Oracle Database[22], le plus célèbre des SGBD propriétaires, intègre, grâce à son extension « Oracle Spatial » des capacités spatiales complètes. Il est édité par l'Oracle Corporation. Bien qu'implantant les recommandations de l'OGC entièrement, le prix d'acquisition de la solution (et de ces éventuelles options) reste rédhibitoire et contraire à nos contraintes ;
- SQL Server[23], édité par Microsoft, est un SGDB propriétaire aussi. Les capacités spatiales que ce logiciel offre se limitent au stockage des géométries.

La seconde option réside dans la possibilité d'accéder à des fichiers spécifiques, référencés. Ces fichiers doivent être lus à partir de l'application, les indexes chargés en mémoire pour retrouver les géométries et les informations liées le plus rapidement. Il existe de nombreux formats dédiés à l'hébergement de données spatiales vectorisées, dont en voici une infime partie :

- MIF/MID : format d'échange du logiciel MapInfo[24]. Le MIF contient la géométrie et la structure de la table, tandis que le MID garde les informations attributaires. Ce sont tous deux des fichiers ASCII (*American Standard Code for Information Interchange*) ;
- *Shapefile*[25] : littéralement, le « fichier de formes », est en fait un lot de trois fichiers, contenant les géométries (SHP), les données attributaires (DBF) et un index de la géométrie (SHX). Ce format est ouvert et très largement utilisé dans le monde des SIG ;
- *Geography Markup Language* (GML) : il s'agit d'un dérivé du XML, dédié à la manipulation de données géographiques, édité par l'OGC, et donc, standard OpenGIS. Il permet de décrire les géométries, les objets s'y rapportant, le système de projection, les métadonnées, etc. ;
- HyperAtlas : format propriétaire généré par HyperAdmin (`.hyp`), il contient les géométries, les stocks de données, ainsi que plusieurs autres informations, notamment une matrice de distance.

Cette approche scindée en deux parties a l'avantage incontestable de permettre la mise à disposition rapide des données spatiales, sans avoir à les retravailler. Le corollaire réside dans le fait que l'application que nous devons bâtir devra, dans ce cas, savoir manipuler ce genre de fichiers.

Nous avons fait le choix de la solution système de gestion de bases de données relationnel qui aurait été préalablement « spatialement activé » afin d'unifier nos accès à ces ressources. Cette décision semble logique quant à notre utilisation de ces données dans le monde du décisionnel où l'on prépare les informations (via les outils d'extraction, de transformation et de chargement)

[18] Ce document de référence 99-049 est disponible à l'adresse suivante : http://portal.opengeospatial.org/files/?artifact_id=829

[19] MySQL : http://www.mysql.fr/

[20] PostGreSQL : http://www.postgresql.org/

[21] PostGIS : http://postgis.refractions.net/

[22] Oracle Database : http://www.oracle.com/us/products/database/overview/index.html

[23] SQL Server : http://www.microsoft.com/sqlserver/

[24] MapInfo (Pitney Bowe Software) : http://www.pbinsight.com/welcome/mapinfo/

[25] Initialement développé par ESRI : http://www.esrifrance.fr/index.aspx

avant de pouvoir les exploiter. Ici, celles-ci seront chargées dans des tables-entités dans le SGBD PostGreSQL et son extension PostGIS. Cette solution s'est imposée aux regards de plusieurs points :

- Le recours exclusif aux logiciels libres (ici, en licence *open source GNU General Public Licence*) ;
- La bonne connaissance de ce SGBD par les membres de l'équipe STEAMER ;
- La prise en compte complète des recommandations de l'OGC pour ce qui est des données et fonctionnalités spatiales. Il est, de plus, conforme aux normes ISO19107[26] et ISO19123[27].

### *b. Stockage des données du cube*

Une fois la question du stockage des données spatiales résolue, celle concernant le media des informations attributaires est plus simple. Bien entendu, nous pourrions dédier un schéma au cube, voire exploiter un autre SGBD, éventuellement nativement multidimensionnel. Mais nous pouvons aussi héberger ces deux sources dans la même instance du système de gestion de base de données. C'est cette dernière solution qui est retenue afin de faciliter la maintenance ainsi que pour la raison exposée précédemment.

### *c. Accès aux données*

Nous sommes donc en présence de deux sources d'informations stockant respectivement les données de géométries et les informations du cube. Le but étant, au final, d'obtenir et d'utiliser à la fois les géométries et les données du cube dans les mêmes résultats. Grâce au choix de ROLAP, deux grandes tendances se dégagent pour arriver à nos fins.

Le stockage des géométries peut être établi dans les tables de dimensions. La base de données étant « spatialisée », les données relatives aux contours des zones peuvent être stockées dans n'importe quelle table et la plus naturelle semble celle qui héberge les autres informations. La figure III-7 illustre ce fonctionnement : (1) l'application demande les informations d'une dimension au moteur SOLAP, (2) celui-ci cherche dans la dimension concernée les informations exigées, et (3) la retourne, complète, pour des traitements ultérieurs.

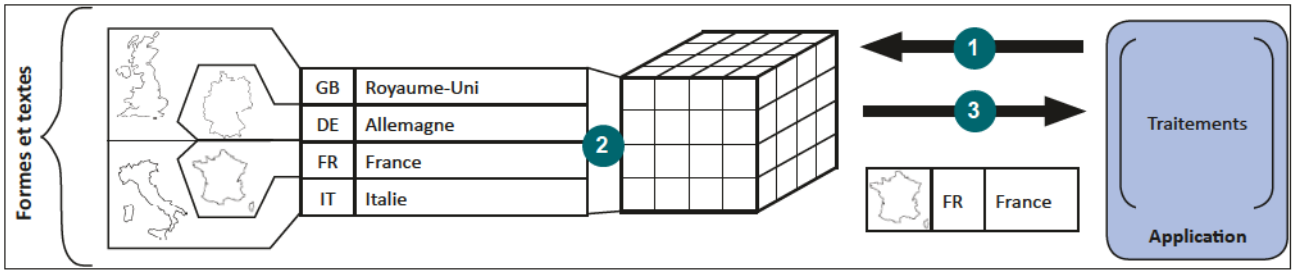



Figure III-7 : Accès aux données spatiales « à la volée »

Cette méthode a l'avantage de réduire le nombre d'opérations à effectuer pour acquérir toutes les informations nécessaires à l'application. De plus, côté ETL, il n'y a qu'une table à renseigner pour que la donnée spatiale soit complète. Elle est cependant moins universelle car elle nécessite que le moteur OLAP soit un *Spatial OLAP*.

[26] ISO19107:2003 : http://www.iso.org/iso/catalogue_detail.htm?csnumber=26012

[27] ISO19123:2005 : http://www.iso.org/iso/catalogue_detail.htm?csnumber=40121

L'autre approche consiste en un mécanisme de correspondance, côté application. Le processus se déroule en cinq étapes, comme le montre la figure III-8. D'abord, (1) l'application émet une requête auprès du cube, (2) celui-ci cherche puis (3) retourne les informations relatives dans la dimension spatiale, et parmi elles, un identifiant. Cet identifiant sert ensuite à l'application pour (4) interroger une table contenant les géométries, dans une base de données relationnelle spatialisée. (5) Le SGBD interrogé retourne l'information demandée. Le tout est (6) fusionné et transféré pour la suite des traitements, au reste de l'application.

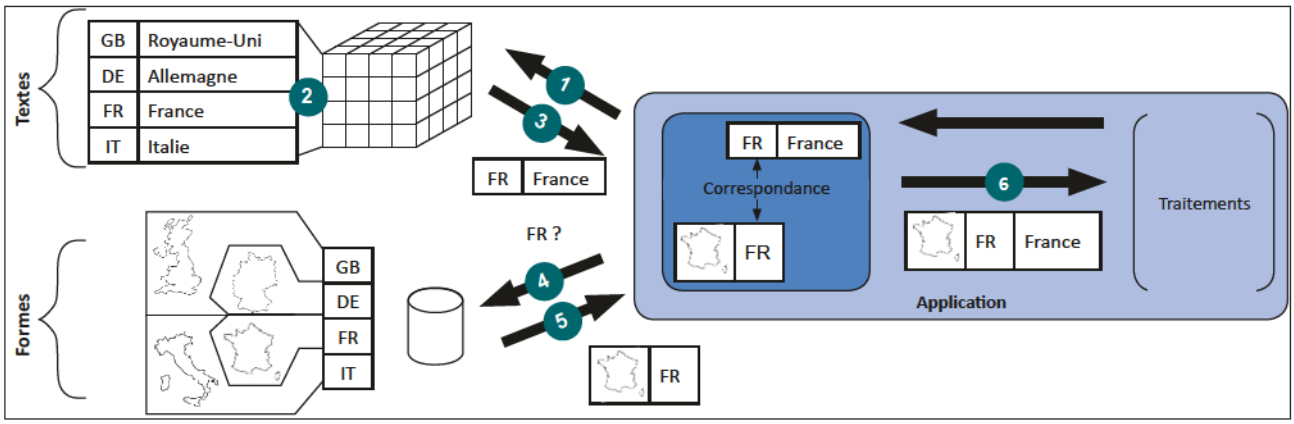



Figure III-8 : Accès aux données spatiales par une connexion annexe

Cette solution offre une flexibilité plus grande que la première : l'information sous forme textuelle est stockée dans un cube qu'un simple moteur OLAP peut extraire, tandis que les géométries sont disponibles dans une base distincte, et réutilisable pour constituer d'autres couples. L'alternative du stockage des géométries dans des fichiers imposerait ce fonctionnement. Elle est plus complexe à mettre en place notamment au niveau paramétrage : il faut définir deux connexions (une dédiée à l'OLAP et l'autre à l'OLTP, *OnLine Transaction Processing*, via JDBC, *Java Database Connectivity*, pour les géométries) pour une seule exploitation spatiale. De plus, elle nécessite une synchronisation à la fois au moment du chargement (l'ETL devra mettre à jour deux emplacements distincts), ainsi qu'à l'interrogation, comme illustré précédemment.

Notre projet étant aussi dédié à explorer et évaluer les capacités spatiales du SOLAP, nous avons opté pour la première approche, afin de tirer profit des moyens offerts pour l'interrogation des données de géométries. Nous pourrons, grâce à ce choix, récupérer, en une requête, mesures (sujet de l'étude) et information géographique pour une représentation cartographique plus performante.

### *2. Serveur SOLAP*

Une fois les bases de l'entrepôt établies, il nous faut penser au serveur SOLAP. Compte tenu des exigences initiales, notamment vis-à-vis de l'utilisation d'outils *open source*, le choix est restreint. À tel point que la seule possibilité se résume à GeoMondrian, édité par Spatialytics[28]. Il est en effet l'unique outil OLAP *open source* supportant l'aspect spatial.

L'étude des outils *Spatial OLAP*, menée dans notre rapport de l'épreuve TEST [Tranchant, 2011], posait le constat que l'application ne semblait pas maintenue assidûment et les documentations manquaient sur son utilisation. Si la seconde partie est toujours vraie, notre expérience sur l'utilisation du logiciel a pu démentir la première. Nous avons été en contact avec l'équipe de développement et Thierry Badard à plusieurs reprises pour leur faire part de nos doléances et nous avons à chaque fois obtenu une réponse concrète pour les résoudre (indication de paramétrage ou version corrigée de GeoMondrian).

[28] Spatialytics : http://www.spatialytics.com/

GeoMondrian nécessite au minimum un serveur de servlets pour fonctionner car il est développé en Java.

### *3. Java et serveur d'application*

Java est un langage orienté objet, né en 1995. L'éventail des possibilités qu'il offre est très complet : application lourde, site Internet, intranet, intergiciel, applet, programme de terminaux *Android*, etc. De plus, grâce à l'utilisation de machines virtuelles, une application Java est multi plate-forme.

Selon les besoins exprimés et les contraintes énoncées, Java possède à la fois le fonctionnement et les bibliothèques nécessaires à l'élaboration des différents mécanismes qui nous seront indispensables, telles qu'un connecteur OLAP, des outils de rendu pour client léger, sans que cette liste soit exhaustive. De plus, les informaticiens de l'équipe STEAMER utilisent essentiellement ce langage pour leurs développements, exception faite des projets dont la technologie en impose un autre, comme les terminaux mobiles Apple par exemple (*Objective C*).

Notre application, nous l'avons vu dans le chapitre « Proposition », expose certaines de ses méthodes en tant services Web. Elle doit donc être accessible par le biais d'un réseau et être hébergée par un serveur d'applications Java.

Il en existe un grand nombre, parmi lesquels nous pouvons citer les plus connus : JBoss AS[29] (RedHat), WebSphere AS[30] (IBM), JOnAS[31] (consortium OW2), GlassFish[32] ou WebLogic[33] (tous deux édités par Oracle Corporation). De cette liste sont exclus Tomcat[34] et Jetty[35] (respectivement édités par les fondations Apache et Eclipse) car il ne s'agit « que » de serveurs HTTP et conteneurs de servlets. Il faut toutefois préciser que les serveurs d'applications sont aussi capables d'agir en tant que simples conteneurs de servlets. C'est pourquoi certains embarquent l'un d'eux : c'est le cas de JBoss AS qui englobe Tomcat en son sein.

Notre application utilisera des *Enterprise JavaBeans* (EJB) (voir IV.C.2.a.ii, page 82), ce qui explique la nécessité d'employer un serveur d'application. Eux seuls ont la capacité d'interpréter cette technologie. En se conformant à la contrainte de ne recourir qu'à des logiciels libres, WebSphere et WebLogic sont dores-et-déjà éliminés. Nous avons aussi besoin de faire appel à la toute dernière spécification des EJB, la version 3.1, pour sa gestion des appels asynchrones de méthodes, excluant de fait, JOnAS.

Ne restent plus que GlassFish et JBoss AS. Il a été très difficile de les départager, tant leurs caractéristiques sont proches. Ils sont tous les deux certifiés compatibles avec *Java Enterprise Edition* 6 (JEE) et supportent les dernières moutures de servlet, de *Java Server Page* (JSP) et de *Java Server Faces* (JSF). Les bancs d'essais de performances exposés sur Internet sont contradictoires : une fois l'un est plus rapide, une autre fois il s'agit de l'autre. Toute opinion objective est alors rendue impossible.

---

[29] RedHat JBoss Application Server : http://www.jboss.org/jbossas

[30] IBM WebSphere Application Server : http://www-01.ibm.com/software/webservers/appserv/was/

[31] OW2 JOnAS : http://jonas.ow2.org/xwiki/bin/view/Main/WebHome

[32] GlashFish : http://glassfish.java.net/fr/

[33] Oracle WebLogic : http://www.oracle.com/technetwork/middleware/weblogic/overview/index.html

[34] Apache Tomcat : http://tomcat.apache.org/

[35] Eclipse Jetty : http://jetty.codehaus.org/jetty/

Nous avons finalement retenu la dernière version disponible de JBoss au début du projet (7.1.1.Final). JBoss est une application qui nous a semblé plus mûre : elle possède un historique plus long ainsi qu'une communauté plus grande que GlassFish. Elle fait de plus partie du consortium *Web Services Interoperability* (WS-I), organisme chargé de promouvoir l'interopérabilité entre plateformes de services web.

Les projets JBoss AS et GlassFish sont très actifs et si le choix d'un serveur d'application devait être fait à l'heure de la rédaction de ce mémoire, il est possible que le résultat ne serait pas identique. L'adoption d'une nouvelle fonctionnalité ou d'une dernière spécification peut faire pencher la balance en faveur de celui qui l'a implanté, selon l'impact qu'elle pourrait avoir sur le projet.

### *4. Fonctionnement global*

Nous avons déterminer l'architecture afin de répondre au mieux à notre sujet et avons choisi quelques-unes des briques nécessaires à la mise en place de notre structure. La figure III-9 illustre le fonctionnement global des différents éléments constitutifs de notre projet.

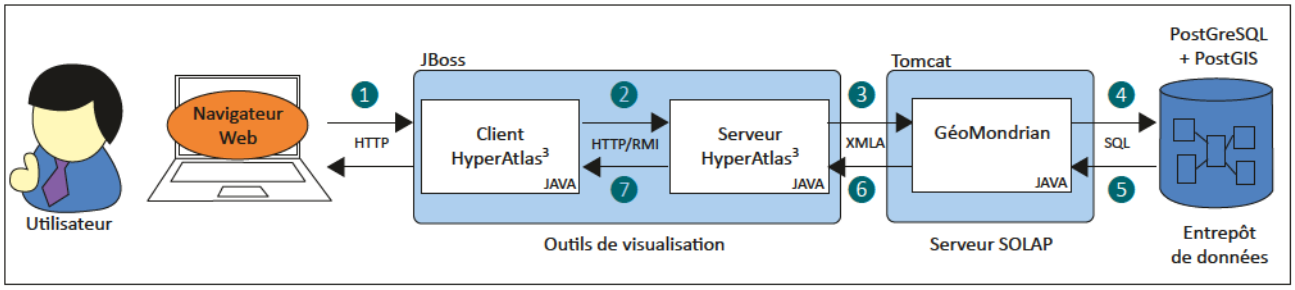



Figure III-9 : Proposition de fonctionnement des briques logicielles

L'utilisateur utilise un navigateur Web généralement déjà installé sur son poste de travail. Il se connecte au client HyperAtlas³ par le protocole HTTP (1). Ce client Java offre les commandes nécessaires à la manipulation du serveur HyperAtlas³ (lui aussi en Java), auquel il est connecté (2). Ce dernier donne accès aux mécanismes pour joindre des serveurs OLAP et SOLAP via XMLA (3). Dans notre architecture, hébergé au sein de Tomcat, GeoMondrian interroge (4) et récupère (5) les informations directement dans l'entrepôt de données stockées, pour notre cas, dans une instance de PostGreSQL spatialement activée grâce à PostGIS. GeoMondrian retransmet l'information au serveur (6), qui lui applique quelques transformations, notamment une génération cartographique, avant de transférer le tout au client (7). L'utilisateur obtient alors la réponse à la question exprimée initialement.

## *D. Feuille de route*

Après avoir analysé les besoins, dessiné les contours de notre proposition et dressé notre démarche, nous rentrons ici dans une analyse plus fine des grandes étapes de la construction du projet.

#### Outils supplémentaires

En plus de JBoss AS, d'autres logiciels nous seront nécessaires pour mener à bien notre projet. C'est le cas de Maven par exemple. Utile dans l'élaboration de nos paquetages, il l'a été encore plus dans la vérification de nos développements par l'application systématique des jeux de tests unitaires JUnit, par le contrôle de leur couverture du code et par la constitution des documentations tant techniques (JavaDoc) que fonctionnelles (site d'explication). Une fois le code stocké dans le dépôt

du gestionnaire de versions, Jenkins profite des puissants mécanismes de Maven pour effectuer l'intégration continue du projet et remonter les différents indicateurs.

### Entrepôt de données et serveur SOLAP

Cœur des systèmes décisionnels, l'entrepôt est le premier élément à mettre en place. Nous avons conclu notre état de l'art par l'adoption de l'approche OLAP relationnel et du modèle mixte. Lors de la phase de décisions, le système de gestion de bases de données a été choisi. PostGreSQL et sa cartouche spatiale constituent donc les fondations de notre entrepôt. GeoMondrian, quant à lui, est en charge de garantir un accès aux données contenues dans le SGBD. Les prochaines étapes consistent à définir les cubes et dimensions de façon à garder un équilibre entre rapidité d'accès et évolutivité de l'entrepôt tout en assurant de couvrir les besoins exprimés.

### Serveur HyperAtlas[3]

Une fois l'entrepôt et le serveur SOLAP mis en place, il nous faut nous atteler à la construction de l'intergiciel. Il doit répondre aux exigences de la directive INSPIRE, permettre une connexion à une source de données, interpréter les informations spatiales, générer une carte correspondant aux attentes de clients, fournir aux clients les renseignements demandés, sans que cette liste soit exhaustive. Par le nombre des objectifs qu'il doit remplir, il constitue le maillon le plus important de ce projet, véritable clé de voute de notre chaîne géodécisionnelle. Il doit être à la fois performant et stable, il est le garant de la crédibilité du système dans son ensemble et de la confiance de l'analyste.

### Client HyperAtlas[3]

En tant que consommateur des services offerts par le serveur, il exploite toutes les capacités de celui-ci afin de répondre aux sollicitations de l'utilisateur. Le client se doit d'être ergonomique à l'usage et agréable aux yeux de ceux qui vont le manipuler. Il doit être accessible aisément et ne pas nécessiter de connaissances particulières pour permettre son utilisation. Pour finir, il doit mettre à disposition tous les outils que l'on retrouve traditionnellement sur de telles applications, tableaux, diagrammes, cartes, ainsi que les options pour affiner leur représentation.

# Partie IV. Réalisation

**Après avoir introduit les besoins, notre démarche et le découpage du projet en grands sous-ensemble, dans la partie précédente, celle-ci s'attèle à décrire, en quatre chapitres, leur réalisation concrète. Le premier chapitre explique notre cadre de travail en passant en revue, notamment, notre gestionnaire de projet. Le second chapitre expose la mise en place de notre chaîne géodécisionnelle et rapporte les étapes de la construction de notre cube. Le troisième détaille les composants constituant le serveur HyperAtlas[3]. Le dernier chapitre montre le client HyperAtlas[3], interface entre l'analyste et le serveur.**

## *A. Outils et méthodes de développement*

Afin de permettre la meilleure réutilisation possible des composants de notre projet, et dans le souhait de fournir des livrables les plus homogènes possibles vis-à-vis de l'existant à l'équipe STEAMER, des outils et des méthodes spécifiques de développement ont été instaurés. Certains, méconnus de quelques membres de l'équipe, ont fait l'objet d'une formation préalable. Sans pour autant décrire le classique environnement de développement Eclipse, très largement plébiscité, ou le serveur d'application Tomcat, tout aussi reconnu, nous avons choisi d'évoquer ici certaines des applications et démarches utilisées lors de notre stage.

### *1. Gestion de projets Maven*

Signifiant « accumulateur de savoir » en Yiddish, Maven est un outil de gestion de projets. Il permet l'automatisation des tâches basiques de la construction de projets Java, et notamment *Java Enterprise Edition*, l'application systématique des tests unitaires qui y sont rattachés. Il gère aussi la production de rapports sur la couverture de ces tests, les métriques, génère la documentation à partir des balises JavaDoc[35], met à disposition le paquetage ainsi produit aux membres de l'équipe sans que cette liste soit exhaustive. Outil gratuit sous licence Apache[36], il est de plus en plus largement adopté par les sociétés éditrices de logiciels fournissant des services informatiques de par sa simplicité de configuration, d'utilisation et l'application d'une convention de programmation aux projets qui y font appel.

Les cinq objectifs de Maven sont [Maven (A), 2012] :

- Simplifier le processus de compilation : en occultant en grande partie les mécanismes sous-jacents de la construction des paquetages ;
- Fournir un système homogène de compilation : grâce au paradigme *Project Object Model* (POM), Maven fournit un système uniforme pour la construction de projets. Une fois familiarisé avec un premier fichier POM, le développeur pourra aisément comprendre tous les autres en un temps minimal ;

[35] La JavaDoc est un outil d'extraction de commentaires spécifiquement dédiés à ce but, au sein des classes Java du paquetage, afin de générer la documentation de l'application.

[36] La licence Apache est une licence de logiciel libre disponible en ligne : http://maven.apache.org/license.html

- Documenter le projet : Maven fournit de nombreuses informations sous forme de rapports, lors de la génération du paquetage. Parmi ces indicateurs, nous pouvons retrouver la liste des dépendances, le résultat des tests, leur couverture du code, définir le taux d'activité du projet, etc. ;
- Fournir les bases d'un développement selon les « meilleures pratiques » : le cycle de vie de génération du paquetage applique systématiquement les tests, extrait la documentation et les rapports, les livrables annexes sont donc toujours à jour, et le code très régulièrement testé. L'arborescence des répertoires et les emplacements des fichiers sont aussi prédéfinis : les sources des tests sont séparés du code « métier », ils sont stockés dans un répertoire parallèle, les fichiers de configurations (resources) ont eux aussi leur emplacement dédié ;
- Faciliter l'évolution du projet : les dépendances et les greffons sont appelés en indiquant la version à utiliser, il est donc aisé de modifier cette version afin d'en utiliser les versions les plus à jour et ainsi garantir une plus grande sécurité dans l'exécution du programme. Maven prend en charge le téléchargement et le stockage des dépendances manquantes. Il n'y a plus qu'à lancer les tests à la suite.

Généralement, chaque développeur a tendance à construire son projet pour faire en sorte, bien sûr, de répondre à la problématique posée, mais aussi en tenant compte de ces habitudes de travail et de ce qu'il a appris durant sa vie professionnelle. Maven propose des conventions, afin que les bases soient communes à tous les membres de toutes les équipes. Les développeurs familiarisés avec ce paradigme sont alors plus à même de comprendre le travail des autres, sans avoir à s'immerger totalement dans le code produit.

Ainsi, à la création d'un nouveau projet (ou module), Maven crée une arborescence de répertoires et une pré-configuration par défaut qui répondront aux attentes de la majorité des applications. La figure IV-1 illustre une architecture type. Les répertoires de (1) concernent le code métier du projet. Ceux de (2) contiennent les fichiers nécessaires à la construction du site d'accompagnement. Les tests sont contenus dans (3). Enfin, (4) contient les livrables (paquetage final, documentation, site) ainsi que les classes compilées.

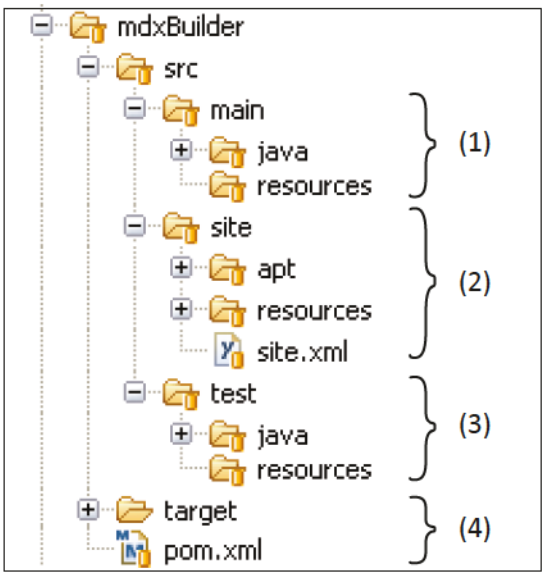



Figure IV-1 : Architecture type d'un projet généré par Maven, depuis l'IDE Eclipse

Mais Maven ne se limite pas à la création de répertoires : les greffons (*plugins*) appliquent un ensemble de conventions pour la compilation du code source, pour le conditionnement des éléments à distribuer, pour la production de sites web, etc. Les phases de construction des livrables suivent un cycle de vie séquentiel qu'il est possible de surcharger à n'importe quel objectif (*goal*) [Maven (B), 2012] :

- Valider (*validate*) : confirme que le projet est correct et que toutes les informations nécessaires sont présentes ;

- Compiler (*compile*) : compile le code source (code métier et code de tests) ;
- Tester (*test*) : exécute les tests embarqués en s'appuyant sur le cadriciel (*framework*) déclaré (le plus utilisé est JUnit, voir point suivant) ;
- Conditionner (*package*) : récupère le code compilé et le conditionne en une archive distribuable parmi : *Java Archive (JAR), Web Archive (WAR) et Enterprise Archive (EAR)* ;
- Tester en intégration (*integration-test*) : déploie le paquetage, si nécessaire, dans un environnement de tests d'intégration ;
- Vérifier (*verify*) : procède à des vérifications pour la validation du paquetage ;
- Installer (*install*) : enregistre l'archive dans le dépôt (*repository*) local, afin qu'elle puisse être réutilisée en tant que dépendance d'une autre application ;
- Déployer (*deploy*) : copie le paquetage final dans un dépôt distant, pour permettre aux autres membres de l'équipe de l'utiliser.

Ainsi, par exemple, l'appel de l'objectif *install* exécutera tous les objectifs qui lui sont antérieurs. En surchargeant certains objectifs, il est possible par exemple, de générer la documentation automatiquement, de la conditionner et de la stocker au même titre que l'archive « opérationnelle » : les utilisateurs de la dépendance (ou artéfact) auront alors la possibilité de la consulter aussi facilement qu'ils l'utilisent. D'autres objectifs, optionnels, non cités dans cette liste, permettent la gestion au quotidien du projet. Il est notamment possible de générer les rapports précités en indiquant l'objectif « *site* », dont vous pouvez trouver, en annexe C, un exemple tiré du développement de ce projet. Il est notamment question des métriques, de la liste des « à faire » (*TODO*), des résultats des tests, de leur couverture du code, de la vérification du format des sources, de la génération des diagrammes de classes UML, etc.

Dans les projets de grande envergure, il est préférable de découper le travail qui est attendu en tâches élémentaires : l'objectif de ces unités est plus clairement défini, donc le code facilement maintenable. Une application peut aussi répondre à plusieurs objectifs : fournir un service accessible via un serveur, et un client permettant de le consommer. Dans ce cas, les conditionnements de ces deux parties doivent être différents : tandis que le premier est un JAR, le second peut être un WAR déployant une interface Web. Maven gère ces projets modulaires et instaure des liens de parenté et d'héritage entre eux. En effet, on peut créer un projet « père », listant les dépendances communes à tous les modules, décrivant les paramètres de compilation, ajoutant des métadonnées transverses (adresse du dépôt de code, membres de l'équipe, etc.) et, bien sûr, référençant les « fils », qui eux, hériteront de ces configurations. Du point central, en une commande, il est possible de déclencher la séquence de compilation et de déploiement, sur tout ou partie des modules, tout en préservant les dépendances de chacun. Ainsi, si A et B sont des modules de C et que A est dépendant de B, alors B sera construit avant A.

Pour conclure, Maven permet la simplification de la gestion au quotidien des dépendances de l'application, favorise la programmation modulaire, encourage le développement de tests, augmente la maintenabilité du code et permet, grâce aux conventions qu'il propose, de se concentrer sur l'essentiel, sur le code, en s'occupant des tâches annexes de compilation, conditionnements, etc. ; tout ceci, en offrant un suivi plus aisé du projet via la remontée d'indicateurs.

### *2. Tests unitaires JUnit*

Nous avons évoqué l'objectif « *test* » du cycle de vie auquel les projets sont soumis grâce à Maven. Durant cette phase, les classes dédiées à la vérification fonctionnelle du code source sont exécutées. Grâce aux tests unitaires, on peut avoir l'assurance qu'une méthode répond aux spécifications, et ce, peu importe les modifications effectuées à la fois sur la méthode elle-même, ou autour d'elle :

on qualifie alors ces tests de « non-régression ». Certaines méthodes, comme TDD (*Test Driven Development*) ou XP (*eXtreme Programming*), prônent l'écriture du test antérieurement à celle du code métier.

Dans le cadre de Java, la bibliothèque JUnit est la plus communément adoptée. Il s'agit d'un cadriciel proposant un environnement pour l'écriture et l'exécution de tests unitaires. Il recommande la rédaction d'une classe de test simple pour vérifier une classe simple. JUnit étant une solution tellement utilisée qu'il est sans doute inutile de s'attarder plus longtemps sur sa description ici.

### *3. Intégration continue Jenkins*

Dès qu'il est question de tests unitaires, et de programmation modulaire, les outils d'intégration continue (ou CI pour *Continuous Integration*) apportent leur lot de fonctionnalités. Parmi elles, la compilation automatique et le test systématique du code produit, à chaque modification de fichier, par tout membre de l'équipe, afin de détecter au plus tôt les régressions et fautes. L'équipe PimLIG a mis en place Jenkins, l'un de ces instruments.

Pour rendre réalisable ces objectifs, l'intégration continue repose sur plusieurs principes :

- Le code source est partagé dans un dépôt unique (*repository*) au sein d'un système de gestion de version (comme Subversion[37] par exemple), et est accessible par l'outil. Toutes les dépendances nécessaires à la compilation doivent être accessibles soit dans le code versionné, soit par un autre moyen clairement défini ;
- La construction de l'application est entièrement automatisée : grâce à l'utilisation d'ANT[38] ou de Maven (d'autres moyens existent), le CI peut produire le code compilé, mais aussi l'empaqueter, ou générer les documentations, sites et rapports s'ils ont été définis ;
- Les tests sont déclenchés à chaque compilation ;
- Les membres de l'équipe enregistrent (*commit*) leurs modifications de code dans le dépôt commun au moins une fois par jour. Plus la fréquence est grande, plus il y aura de tests, et si problème il y a, il sera détecté plus tôt ;
- Chaque enregistrement dans le dépôt doit être compilable. On n'envoie pas de code qui ne compile pas ;
- La compilation doit être et rester rapide ;
- Les tests d'intégration doivent être effectués dans un environnement quasi-identique à celui de production ;
- Les dernières compilations doivent rester accessibles, afin de pouvoir récupérer les derniers rapports, documentations et exécutables fonctionnels ;
- Les rapports de compilation sont accessibles à tous : chacun peut constater l'origine des défaillances ;
- Le déploiement automatique de l'exécutable est possible.

L'intégration continue a l'avantage d'éviter l'éventuel chaos de la dernière minute : la version fonctionnelle la plus récente de l'application est toujours accessible, que ce soit pour des tests, une démonstration ou une livraison.

Jenkins embarque Maven en son sein, lui permettant l'exploitation des fichiers POM décrits dans les projets et une parfaite symbiose avec les outils développés pour l'application elle-même. Il est

---

[37] Abrégé SVN. Logiciel de gestion de versions, projet de la Fondation Apache, sous licence Apache/BSD. http://subversion.apache.org/

[38] Produit par la Fondation Apache (comme Maven), sous licence Apache. Il s'agit d'un logiciel d'automatisation des opérations répétitives. http://ant.apache.org/

alors inutile de coder des mécanismes supplémentaires qui serviront exclusivement au logiciel d'intégration continue.

## B. Chaîne géodécisionnelle

Comme nous l'avons évoqué dans notre état de l'art, l'entrepôt de données est le cœur du système décisionnel. Notre proposition a exposé les principes sur lesquels nous allions nous appuyer au niveau du stockage et de l'accès aux données. Ce chapitre détaille les étapes de la construction logique de l'entrepôt ainsi que la mise en place du serveur SOLAP permettant son exploitation.

### 1. Modélisation du cube

La modélisation de l'entrepôt est une étape cruciale dans l'élaboration d'un système décisionnel. Le choix des dimensions, surtout, définit toutes les opérations qui seront ensuite permises, ainsi que la navigation au travers les données de l'hypercube. C'est pourquoi, cette décision doit être réalisée le plus tôt possible dans le processus de spécification de l'entrepôt.

Si l'objectif de tester les capacités SOLAP était clairement défini dès le commencement des travaux, il en était tout autrement des perspectives selon lesquelles nous pourrions étudier le résultat de ces fonctionnalités.

#### a. Expérimentation

Dans un premier temps, nous avons adopté une méthode empirique afin d'expérimenter plusieurs structures et modèles, de nous familiariser avec la manipulation et le paramétrage des dimensions, et de découvrir les avantages et restrictions de chacun. Cette phase de sensibilisation nous a amenés à nous poser les bonnes questions celles auxquelles nous voulions que l'outil réponde.

Ainsi, notre première version s'inscrivait dans une stricte continuité avec HyperAtlas. Comme nous l'avons rappelé dans le contexte, ce logiciel se base essentiellement sur les divisions territoriales de type NUTS pour l'affichage des stocks. Basé sur le modèle flocon, l'hypercube créé offrait une dimension spatiale à quatre niveaux, reflétant ceux du découpage NUTS, et autorisant ainsi une navigation hiérarchique simple des unités spatiales. Finalement, cette modélisation était trop restrictive : en voulant garder un point de comparaison avec un outil connu, nous avions bridé les capacités décisionnelles notamment dans le fait de n'avoir qu'une dimension spatiale de type NUTS.

Une version ultérieure, à trois dimensions, a été construite sur une architecture hydride, à mi-chemin de l'étoile, par l'utilisation d'une seule table pour chacune des dimensions, et du flocon, par la présence d'une clé étrangère référençant la clé primaire et autorisant donc l'extension à l'infini du nombre de niveaux. Celles-ci introduisaient une récursivité dans l'exploration de la dimension. Le but de cette version « étoile floconnée » était de prendre en considération toutes les hiérarchies spatiales possibles, dont le nombre de niveaux peut varier d'un type à un autre. C'est notamment le cas entre le découpage de type NUTS, possédant quatre niveaux, et le type GRID qui n'est pas hiérarchique (un seul niveau). Malheureusement, il est impossible de croiser les membres d'une même dimension sur deux axes différents.

#### b. Travaux de l'UAB

Pendant nos tests, nous avons appris qu'un groupe de chercheurs espagnols de l'Université Autonome de Barcelone (UAB ou *Universitat Autònoma de Barcelona*) travaillait à la mise au point d'algorithmes pour la phase de chargement de l'entrepôt de données socio-économiques et environnementales.

Ce groupe est l'un des partenaires de l'équipe STEAMER dans le cadre du projet ESPON *Database M4D*.

Nous avons étudié la présentation de leurs résultats, afin de comprendre quels buts ils poursuivaient, les difficultés qu'ils ont rencontrées, et ce, dans le but d'apprendre les moyens qu'ils ont déployé et les principes qui sous-tendent leur application.

Roger Milego, lors de la *GISCO Working Party* de mars 2012, introduit les tenants et aboutissants de l'OLAP pour la combinaison de données socio-économiques et environnementales. Présentée comme la réponse au défi d'hétérogénéité d'échelle et d'objets géographiques, le tout horodaté, la solution qu'il propose repose sur une désagrégation des données socio-économiques brutes et une pondération, avant l'intégration des informations obtenues dans l'entrepôt [Milego, 2012].

Les sources utiles à l'équipe de Roger Milego sont variées :

- Les données socio-économiques : la population totale, active, au chômage, le produit intérieur brut (PIB) sont relevées par les États, et donc rattachées au découpage administratif NUTS ;
- Les informations sur les pollutions des eaux : elles sont référencées par bassins versants, aires dans lesquelles l'eau drainée a le même exutoire, et dont les périmètres sont constitués par les lignes de partage des eaux [Touchart, 2004] ;
- Les renseignements biophysiques : couverts par la classification *Corine Land Cover* (CLC), ils sont répartis sur une grille de 100 mètres sur 100 mètres. Ils permettent de cartographier le territoire de l'Union Européenne en affectant à chaque cellule de la grille l'une des 44 classes disponibles [Gouvernement, 2011] ;
- Des découpages géographiques différents : en plus des NUTS et des bassins versants, d'autres typologies sont envisageables. Il peut s'agir de la répartition par l'altitude (*elevation breakdown*, répartie en cinq catégories), par massifs montagneux (divisés en 17 zones), par bassins maritimes régionaux (cinq territoires), etc.

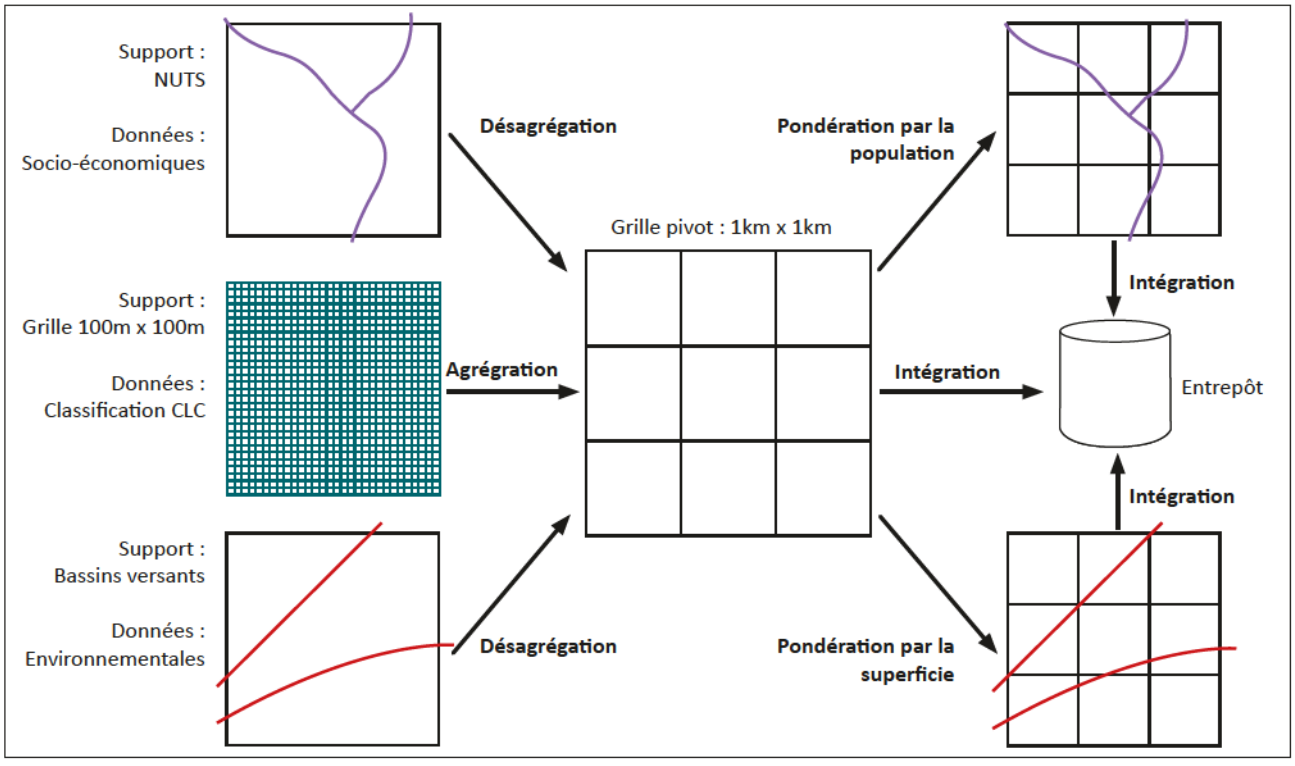



Figure IV-2 : Transfert des données socio-économiques et environnementales aux supports géographiques

Le problème majeur de ces données réside dans le fait qu'elles ne partagent pas d'autres référentiels que celui de toutes appartenir au continent européen. Sans calcul préalable, il est donc difficile, voire impossible de déterminer, par exemple, la population d'un bassin versant, ou inversement, le taux de pollution dans un département. L'équipe de recherche de l'UAB propose donc une méthode pour transférer toutes les données dans tous les découpages spatiaux. En limitant l'exemple au découpage NUTS, la classification CLC et les bassins versants, les opérations nécessaires sont illustrées par la figure IV-2 ci-contre.

Les sources sont donc toutes ramenées à la « grille-pivot » servant de référence à toutes les transformations. Cette grille est la grille de référence Européenne[39] (*European Reference Grid*). Elle est composée de zones carrées d'un kilomètre de côté et est aussi utilisée comme support pour des indicateurs d'occupation des sols, et environnementaux.

La première étape consiste à superposer les autres objets géométriques sur cette grille et à en faire l'intersection. On obtient, par exemple, une situation telle que la représente la figure IV-3.

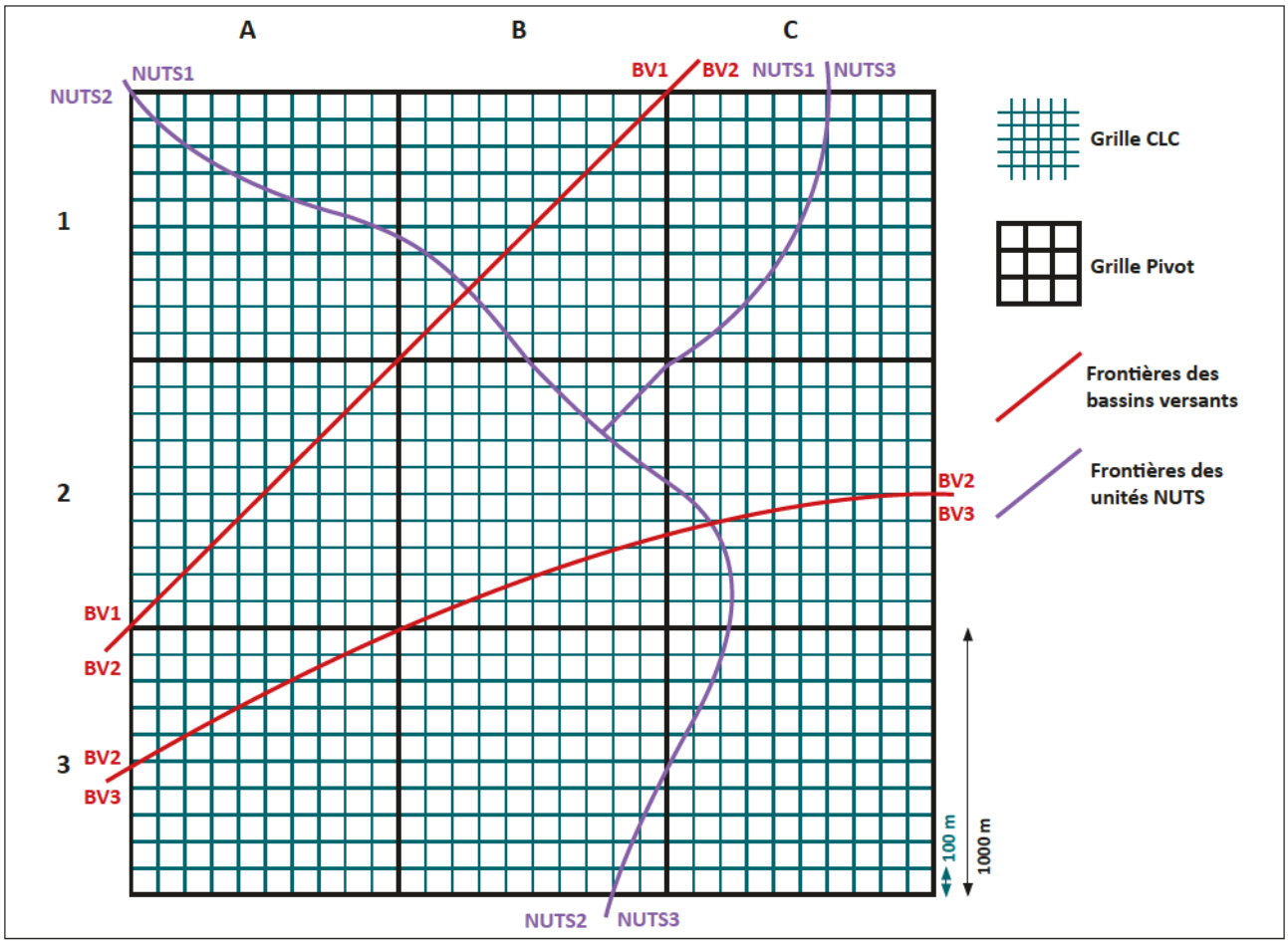



Figure IV-3 : Superposition et intersection des objets géographiques

Sur cette illustration, on peut voir les différentes unités spatiales entrant en jeu. Dans la case référencée B1 se trouvent :

- 100 cases de la grille CLC ;
- Deux divisions de bassins versants :
  - BV1 couvre 50% de B1 ;
  - BV2 couvre lui-aussi 50% de B1 ;
- Deux entités NUTS :
  - NUTS1 occupe 88% de la zone ;
  - NUTS2 occupe les 12% restants.

[39] L'EEA recommande l'utilisation de cette grille depuis le premier Atelier Européen sur les références, en 2003. http://eusoils.jrc.ec.europa.eu/projects/alpsis/Docs/ref_grid_sh_proc_draft.pdf

Pour définir les valeurs au niveau de la grille pivot, trois méthodes sont applicables pour les données socio-économiques et environnementales [Milego&Ramos, 2011].

Une agrégation préalable est opérée pour déterminer la classe CLC de chaque cellule. Un décompte par type des 100 cellules la constituant est effectué et le type le plus représenté est gardé.

### Zone maximum

Appliquée aux variables indénombrables, la valeur d'une cellule prend la valeur de l'unité qui la couvre le plus, comme l'illustre l'algorithme présenté par la figure IV-4.

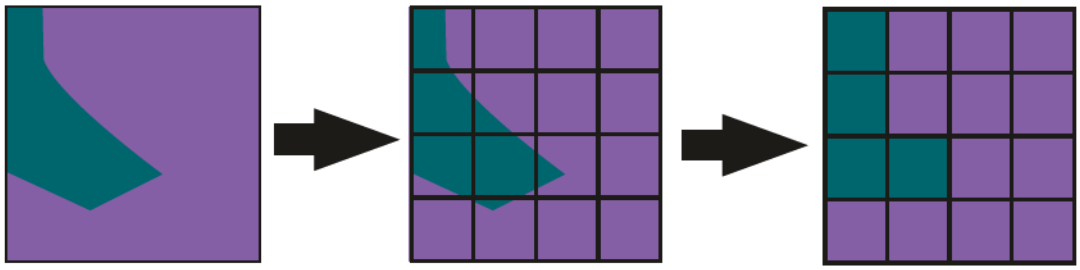


Figure IV-4 : Zone maximum (Source : [Milego&Ramos, 2011])

### Calcul proportionnel

En ce qui concerne les variables dénombrables, la cellule est calculée dépendamment de la valeur des unités et de la surface qu'elle occupe dans cette case (figure IV-5).

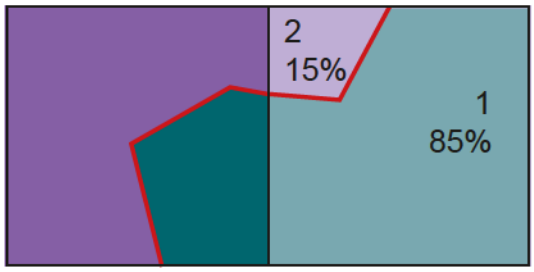



Figure IV-5 : Illustration du calcul proportionnel (Source : [Milego&Ramos, 2011])

La méthode de calcul appliquée est la suivante (figure IV-6) :

$$\text{Valeur de la cellule} = \Sigma( V_i * \text{PourcentSurface}_i )$$

Où :

$V_i$ = Valeur de l'unité i

$\text{PourcentSurface}_i$ = Pourcentage d'occupation de la surface de i dans la cellule

Figure IV-6 : Calcul proportionnel (Source : [Milego&Ramos, 2011])

Dans l'exemple illustré par la figure IV-5, la valeur de la cellule de droite sera le résultat de l'opération suivante : V1 * 0.85 + V2 * 0.15.

### Calcul proportionnel et pondéré

Pour améliorer la répartition territoriale d'un indicateur socio-économique comme le PIB, le calcul proportionnel peut-être pondéré par une variable extérieure, en l'occurrence, la population. En effet, le Centre Commun de Recherche (CCR) Européen propose un décompte démographique précis, établi sur la grille européenne de référence [Milego&Ramos, 2011]. Le PIB peut ainsi être distribué sur ce quadrillage grâce à la pondération par la population.

La méthode de calcul appliquée est la suivante (figure IV-7) :

$$\text{Valeur de la cellule} = W_c * \Sigma( V_i * \text{PourcentSurface}_i )$$

Où:

$V_i$ = Valeur de l'unité i

$\text{PourcentSurface}_i$ = Pourcentage d'occupation de la surface de i dans la cellule

$W_c$ = Poids assigné à la cellule

Figure IV-7 : Calcul proportionnel et pondéré (Source : [Milego&Ramos, 2011])

Finalement, après les calculs, les données sont stockées dans l'entrepôt, à la fois dans les dimensions et les faits, à la granularité la plus grande :

| **Dimensions** | | | **Mesures** | |
|---|---|---|---|---|
| **Code NUTS niveau 3** | **Corine Land Cover niveau 3** | **Bassin versant** | **Surface (hectares)** | **Chômage (habitants)** |
| FR714 Isère | 111 Tissu urbain continu | Rhône et littoral méditerranéen | 449 | 1 433 |

Tableau IV-1 : Données intégrées à l'entrepôt (Source : [Milego&Ramos, 2011])

Ce tableau montre l'exemple d'une ligne de la table de fait, dénormalisée pour une meilleure lecture, ainsi que les valeurs les plus fines que l'on puisse obtenir. Il y a d'autres perspectives et mesures étudiables. Les dimensions retenues par les chercheurs de l'UAB sont [Milego, 2012] :

- Les régions biogéographiques/climatiques ;
- Les *Corine Land Cover* en versions 1990, 2000 et 2006 ;
- Les changements de CLC entre 1990 et 2000, entre 2000 et 2006 et entre 1990 et 2006 ;
- Les *Land Cover Flows* entre 1990 et 2000, entre 2000 et 2006 et entre 1990 et 2006 ;
- Les types dominants de couverture terrestre (*Dominant Land Cover Types*) ;
- Les zones urbaines larges (LUZ ou *Large Urban Zones*) ;
- La répartition par l'altitude (*elevation breakdown*) ;
- Les massifs montagneux ;
- Les découpages administratifs NUTS de 2003 et 2006 ;
- Les bassins maritimes régionaux (*River Sea Basins*) ;
- Les bassins versants (*River Basin Districts*).

Quant aux mesures considérées, elles sont listées ci-après :

- La population de 2001 ;
- La population active en 2001 et en 2006 ;
- Le PIB de 2001 et de 2006 ;
- Le chômage de 2001 et 2006 ;
- La superficie.

L'équipe conclut son analyse en précisant que toute modification dans l'entrepôt impliquera de devoir le vider entièrement, de procéder à tous les calculs une nouvelle fois, avec les nouvelles données, puis de les charger à nouveau dans les tables.

### *c. Notre hypercube*

À partir des résultats de nos essais successifs et des investigations du groupe de recherche de l'UAB, nous avons modélisé un hypercube qui répond à nos besoins :

- Croiser des dimensions spatiales de types hétérogènes ;
- Faire cohabiter les différentes versions de ces découpages ;
- Exploiter les données socio-économiques et environnementales.

Tenant compte de ces exigences, la figure IV-8 ci-dessous montre la structure que nous avons mise en place.

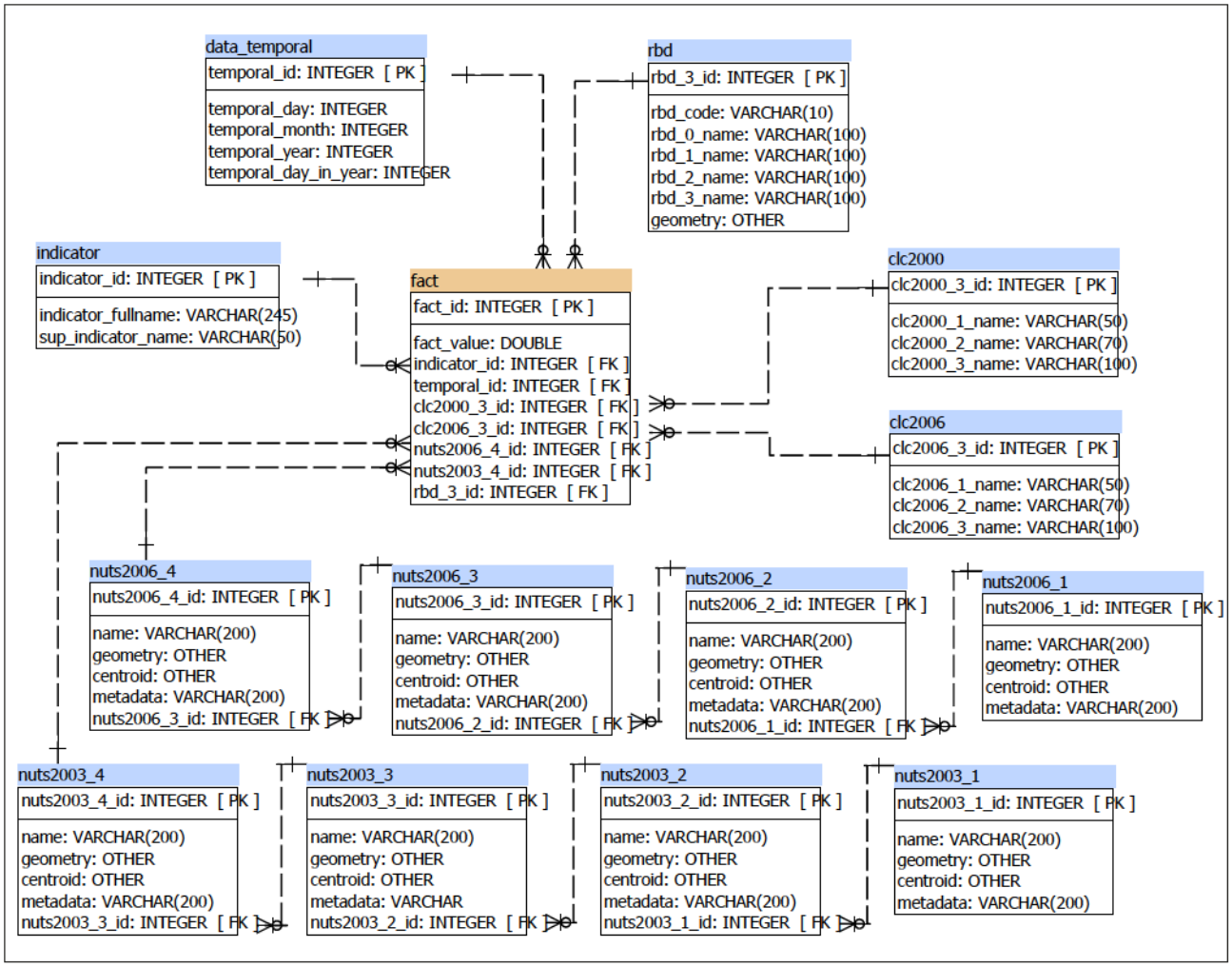



**Figure IV-8 : Structure mixte mise en place pour notre entrepôt**

Nous avons fait le pari de définir les indicateurs en tant qu'une dimension (à deux niveaux), plutôt que comme des mesures dans la table de faits. Cette structure a pour avantage de donner la possibilité d'exploiter une hiérarchie d'indicateurs. Par exemple, la population (niveau le plus haut) est subdivisée en deux catégories : la population féminine et masculine (niveaux inférieurs). En ajoutant un nouveau niveau, il serait possible de redéfinir chacune de ces répartitions par tranches d'âge, par exemple. Autre avantage tout aussi important : l'ajout d'un indicateur n'oblige pas forcément le recalcul de toutes les données du cube. S'il n'est pas en rapport avec ceux qui existent déjà (ajout de la pollution, alors que les indicateurs présents sont PIB et population), il n'y aura aucun problème.

La dimension temporelle (sur la figure IV-8 : *data_temporal*) est utilisée pour gérer les différents horodatages des indicateurs. Ainsi, pour obtenir la population de 2006, il faut sélectionner le membre « 2006 » de la dimension temporelle et le membre « population » de la dimension « indicateur ».

Ceci fait ressortir la principale contrepartie de ce choix : les données étant, par défaut, agrégées aux plus hauts niveaux, certains résultats sont totalement dénués de sens. En effet, l'agrégat utilisé pour les indicateurs que nous avons choisi est la somme, or, si nous ne sélectionnons pas de membre dans la dimension temporelle, les populations de 2001 et de 2006 (en admettant qu'il s'agisse des seules valeurs) seront sommées.

Pour croiser les données, il faut qu'elles soient stockées dans des dimensions différentes. L'un des objectifs exprimés étant de pouvoir effectuer l'intersection d'objets géographiques, chacun doit avoir une dimension dédiée. La même règle est appliquée pour gérer les multiples versions de ces découpages, ainsi pour représenter les NUTS en versions 2003 et 2006, deux dimensions sont nécessaires. Il en va de même pour la classification *Corine Land Cover*, des cellules de la grille européenne évoluent dans le temps : les zones urbaines « grignotent » les espaces ruraux, des forêts disparaissent au profit de terres arables, etc. Pour finir, les bassins versants constituent notre septième axe d'étude.

Comme brièvement évoqué un peu plus haut, nos « mesures » sont stockées dans une dimension nommée, sur la figure IV-8, « *indicator* ». Notre fait ne possède donc qu'une seule valeur « *fact_value* », et les clés étrangères des dimensions.

Pour finir sur la construction de notre hypercube, le travail de génération de données réelles est une tâche amont à notre projet, située à l'étape extraction, transformation et chargement, c'est pourquoi, même si les données des dimensions sont correctes, les valeurs des faits, quant à elles, ont été générées aléatoirement, et n'ont donc aucune véracité.

### *2. Spatial OLAP*

Une fois l'entrepôt modélisé, et son contenu défini, le serveur *Spatial OLAP* doit être installé et configuré. Il permet un accès multidimensionnel aux données stockées, comme nous l'avons exposé dans notre état de l'art. Nous avons, dans notre proposition, retenu GeoMondrian à cet office. Basé sur le célèbre serveur OLAP Mondrian, GeoMondrian a été conçu pour manipuler les objets géographiques inhérents à SOLAP et comprendre les fonctions qui les entourent.

Le paramétrage de GeoMondrian se déroule en deux étapes. D'abord, l'application doit être capable de se connecter aux sources de données. Dans le cas qui nous occupe, il s'agit de l'entrepôt précédemment évoqué et donc du SGBD qui le contient. Ce lien est établi grâce au fichier `datasource.xml` contenu dans le paquetage livré par Spatialytics. Les réglages sont triviaux et se limitent à nommer la source d'informations et définir les paramètres de connexion à la base de données.

La seconde partie de cette configuration réside dans la description de la structure de l'entrepôt. En effet, le serveur (S)OLAP ne peut pas « deviner » comment est censé fonctionner les tables et attributs contenus dans la base de données. Pour lui résoudre cette énigme, un fichier de définition doit alors être créé.

Un extrait de ce fichier est exposé dans le tableau IV-2 ci-après, et est entièrement retranscrit dans l'annexe D.

```
<Schema name="hyperatlascube">
  <!-- Description du cube -->
  <Cube name="Indicators" defaultMeasure="Measure" […]>

    <!-- Déclaration de la table de fait -->
    <Table name="fact" />

    <!-- Description de la mesure -->
    <Measure name="Measure" column="fact_value" aggregator="sum" />

    <!-- Description de la dimension NUTS2006 et de ses quatre niveaux -->
    <Dimension type="StandardDimension" foreignKey="nuts2006_4_id" name="NUTS2006">

      <!-- Hiérarchie par défaut de la dimension -->
      <Hierarchy hasAll="true" primaryKey="nuts2006_4_id"
                                              primaryKeyTable="nuts2006_4">
        <!-- Description du fonctionnement des tables entrant en jeu
                                                              (en flocon)-->
        <Join leftKey="nuts2006_3_id" rightKey="nuts2006_3_id">
          <Table name="nuts2006_4" />
          <Join leftKey="nuts2006_2_id" rightAlias="nuts2006_2"
                                              rightKey="nuts2006_2_id">
            <Table name="nuts2006_3"/>
            <Join leftKey="nuts2006_1_id" rightKey="nuts2006_1_id">
              <Table name="nuts2006_2" />
              <Table name="nuts2006_1" />
            </Join>
          </Join>
        </Join>

        <!-- Déclaration des niveaux, des clés, et des géométries
                                                              (Property) -->
        <Level name="NUTS2006 level 0" table="nuts2006_1" column="code"
                    nameColumn="name" […]>
          <Property name="geom" column="geometry" type="Geometry" />
        </Level>
        <Level name="NUTS2006 level 1" table="nuts2006_2" column="code"
                    nameColumn="name" […]>
          <Property name="geom" column="geometry" type="Geometry" />
        </Level>
        <Level name="NUTS2006 level 2" table="nuts2006_3" column="code"
                    nameColumn="name" […]>
          <Property name="geom" column="geometry" type="Geometry" />
        </Level>
        <Level name="NUTS2006 level 3" table="nuts2006_4" column="code"
                    nameColumn="name" […]>
          <Property name="geom" column="geometry" type="Geometry" />
        </Level>
      </Hierarchy>
    </Dimension>

    <!-- Description des autres dimensions […] -->

  </Cube>
</Schema>
```

**Tableau IV-2 : Fichier XML simplifié de description du contenu de l'entrepôt**

Basé sur XML, il dresse une liste des dimensions, hiérarchies et niveaux que le cube contient et énumère les mesures qu'il est possible d'interroger.

Tout d'abord, le schéma est déclaré : il englobe tous les cubes disponibles. Ceux-ci constituent le nœud suivant, ils sont décrits par un nom et déclarent une mesure par défaut. Les tables de faits sont

ensuite inventoriées ainsi que les mesures. Ces dernières sont nommées et l'opération de l'agrégat est déclarée. Ici, il n'y a qu'une table de fait et une seule mesure dont l'agrégat est la somme.

Les nœuds restants sont, pour l'essentiel, réservés aux dimensions. Ainsi chacune d'elles est nommée et rattachée à la table de fait par la clé étrangère que celle-ci référence. Nous avons vu, sur la figure IV-8, que la dimension NUTS2006 était modélisée sous forme « flocon » par le biais de quatre tables. Il est alors naturel de retrouver ce fonctionnement au-travers de ce fichier : celles-ci sont énumérées et les jointures déclarées grâce aux clés primaires et étrangères. Les niveaux closent la configuration des dimensions. C'est ici que les propriétés spécifiques peuvent être déclarées : c'est notamment le cas des géométries, stockées dans l'attribut « *geometry* » des tables en question.

Toutes les dimensions sont ainsi décrites. Quelques différences sont à relever : la dimension temporelle possède des attributs spécifiques, allégeant sa notation ; les dimensions basées sur une structure étoile sont, elles aussi, plus simplement configurées.

Une fois ce fichier configuré et installé dans le paquetage GeoMondrian et le tout déployé au sein d'un serveur Tomcat, le serveur SOLAP est à même d'exécuter les requêtes XMLA adressées par les logiciels tiers se connectant.

Le début de notre chaîne géodécisionnelle est désormais opérationnel : il est possible d'interroger l'entrepôt par le biais du serveur SOLAP, grâce aux requêtes MDX, portée par XMLA. Le chapitre suivant décrit la mise en place et le développement du serveur HyperAtlas³ exploitant cette architecture.

## C. Serveur HyperAtlas³

Le cœur de notre projet consiste en ce serveur que nous avons nommé HyperAtlas³, allusion à la fois à cette troisième version majeure du logiciel HyperAtlas, ainsi qu'au fait qu'elle repose entièrement sur un moteur multidimensionnel.

Nous avons pu constater, au cours de la section « chaîne géodécisionnelle », que l'entrepôt et le serveur SOLAP sont interconnectés, et que ce dernier est joignable par le biais des technologies mises en œuvre par *XML for Analysis*. Le serveur HyperAtlas³ se positionne en tant que consommateur de services XMLA et en fournisseur de solutions d'exploration des cubes.

Après s'être arrêté sur une vue globale de l'outil et avoir rappelé son positionnement dans notre architecture, ce chapitre détaille les modules composant notre serveur.

### 1. Vue d'ensemble

Le serveur HyperAtlas³ doit répondre à plusieurs objectifs : prendre en compte les sollicitations des clients, interroger le serveur OLAP, générer les cartes correspondant aux données collectées et fournir le tout en retour, comme le montre la figure IV-9.

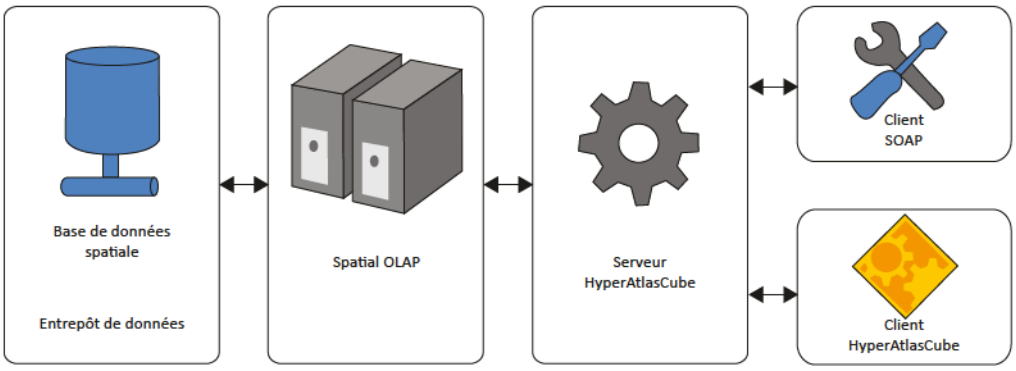



**Figure IV-9 : Positionnement du serveur HyperAtlas³ vis-à-vis des autres éléments**

Afin de bien séparer les différentes tâches, l'application a été subdivisée en modules, chacun devant s'acquitter d'une mission précise. La figure IV-10 ci-dessous illustre ce découpage.

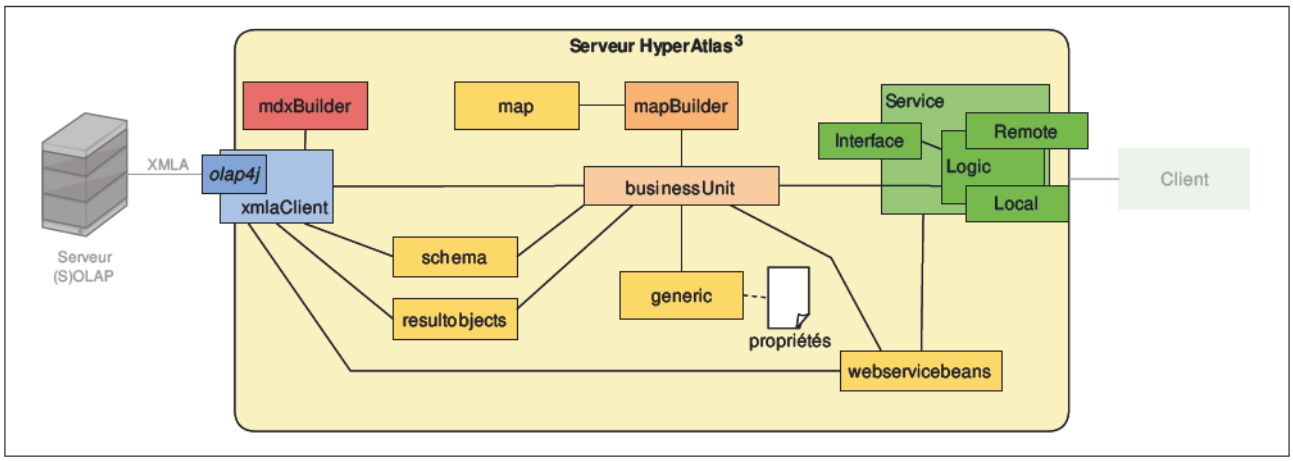



Figure IV-10 : Schématisation des modules d'HyperAtlas³

On peut aussi constater les canaux de communication entre les différents éléments : à gauche, connecté sur le module nommé `olap4j`, se trouve(nt) le(s) serveur(s) (S)OLAP ; à droite les clients sont interfacés via les logiques *remote* et *local* des services. Entre les deux, le chef d'orchestre, « *BusinessUnit* », cadence les demandes émanant des consommateurs jusqu'aux serveurs SOLAP qu'il connait. Il interprète ensuite les données récupérées, fait appel au module « *MapBuilder* » pour construire une carte si les informations s'y prêtent. Enfin il retransmet données transformées et images aux demandeurs.

Un fichier de configuration nommé « propriétés » référencie les divers paramétrages disponibles pour décrire le fonctionnement de l'application. Il contient notamment les variables de connexion aux serveurs OLAP : identifiant interne, nom public, adresse pour joindre le serveur et éventuellement un couple identifiant et mot de passe nécessaire à l'authentification. Nous appelons cette collection de données « projet ».

Les modules dont le fond est jaune (*map*, *schema*, *resultobjects*, *generic* et *webservicesbeans*) sont transversaux : ils permettent la communication entre les parties, certains sont réutilisés par le client. En vert, ce sont ceux nécessaires à la mise à disposition des services ; en bleu, celui chargé du contact vers les serveurs de données ; en rouge, le gestionnaire de requêtes MDX ; et en orange foncé, la génération de cartes.

Chaque module est spécialisé : ils ne répondent qu'à une mission très spécifique. Leurs objectifs et fonctionnements respectifs sont détaillés dans la section suivante.

## 2. Modules

### a. Service

Porte d'entrée de l'application, le module « *service* » abrite, comme son nom l'indique, les services, qu'ils soient exposés sur le Web ou non. Notre application est orientée services : elle est accessible par toute application tierce connaissant l'adresse des services qu'elle met à disposition. Toutes les actions ultérieures sont la conséquence d'une connexion sur ces services. Une structure Web n'est pas *stricto sensu* nécessaire pour notre projet, si ce n'est pour répondre aux exigences de la directive INSPIRE en matière de partage des informations.

### *i. Architecture*

Un service Web (ou *webservice*) est une application proposant au moins un mécanisme de communication standard et visant à mettre à disposition une partie des traitements et/ou des données qu'elle manipule. Cet échange de données entre programmes ou systèmes hétérogènes repose le plus souvent sur des messages XML, généralement sans intervention humaine et de manière synchrone.

Il existe essentiellement deux approches pour mettre à disposition des services Web : *Representational State Transfer* (REST) et *Web Services* (WS-*). Même s'ils répondent à des besoins utilisateurs différents, leur point de divergence majeur se situe sur les technologies sur lesquelles ils reposent [Poulenard, 2011].

REST est considéré comme « orienté ressource[40] ». Le protocole de communication repose sur l'utilisation de l'URL (*Uniform Resource Locator*) pour le passage de paramètres. Il exploite pleinement les capacités de HTTP, notamment sa standardisation et son riche vocabulaire. Du point de vue programmation, il est plus simple à mettre en place.

WS-*, quant à lui, est plutôt « orienté service[41] » et *Simple Object Access Protocol* (SOAP) en est une des spécifications les plus connues et utilisées. Celui-ci est généralement plus lourd à mettre en place (dans le sens où une architecture logicielle ad hoc est nécessaire) mais permet l'utilisation d'un vocabulaire spécifique, plus en adéquation avec un contexte métier non standardisé. Cette spécialisation est possible via un WSDL dédié, *Web Services Description Language*. Elle a cependant un coût. Comme SOAP est basé sur XML, il est beaucoup plus verbeux et donc plus lent à l'utilisation. En revanche, le cadre des messages est strict et rigoureux, protocolaire. Enfin, pour un maximum d'interopérabilité, les services sont interrogés par un message SOAP transitant généralement par HTTP.

La comparaison REST/SOAP est toujours sujette à de vifs débats. Nous retiendrons seulement que selon l'application et l'approche que l'on souhaite y donner, l'une ou l'autre des technologies conviendra.

Pour notre application, nous avons retenu la technologie SOAP pour plusieurs motifs.

La première est extérieure à notre application, mais cependant propre à notre domaine d'étude. L'absence de standards dans les explorations des cubes et de leurs réponses nous conforte dans le fait qu'un connecteur « sur mesure » est la réponse la plus appropriée, du moins pour le moment. Notre serveur, quant à lui, doit pouvoir fournir une information complète mais constituée d'éléments hétérogènes : données tabulaires et images constituant la carte et sa légende.

La seconde concerne la consommation de ces services : elle n'est prévue que par un autre programme (M2M *Machine to machine*), pas directement par un utilisateur humain. L'usage de SOAP permet le passage de messages complexes et implique la mise en place, côté client, d'un connecteur voué à cette connexion. Il s'agit d'un contrat passé entre les deux machines, tous les éléments de conversations sont normalisés. Ceci est possible grâce à WSDL. C'est aussi par son biais qu'il est aisé de générer cette « prise » du côté consommateur.

[40] Architecture orientée ressource : ROA *Resource Oriented Architecture.*

[41] Architecture orientée service : *SOA Service Oriented Architecture.* Ces services donnent accès à un traitement.

### *ii. Technologies*

Nos services sont construits sur des *Entreprise JavaBeans*. Ceux-ci nous ont permis de nous concentrer sur l'essentiel : la logique du métier. En effet, les EJB nécessitent une structure particulière pour pouvoir fonctionner, un conteneur. Celui-ci encapsule les composants, fournit un lot de services pour les atteindre (nommage, sécurité, accès concurrentiel) et gère leurs cycles de vie.

Les EJB sont des classes universellement accessibles : il peut s'agir d'un appel interne à la même application, à partir de la même machine virtuelle Java ou depuis un autre serveur sur le réseau. Ceci est possible grâce à l'interface de programmation *Remote Method Invocation* (RMI), API permettant l'appel de méthodes distantes. Un annuaire *Java Naming and Directory Interface* (JNDI) référencie les ressources proposées par le serveur. Ainsi, si une application interroge cet annuaire pour l'accès à une méthode, il lui renvoie le chemin d'accès à cette ressource. Grâce à ce chemin et à RMI, la méthode pourra être facilement atteinte.

Jusqu'à leur version 2.1, les classes EJB devaient être accompagnées de fichiers XML décrivant leur déploiement au conteneur. Dans ces fichiers, les paramétrages permettaient d'indiquer le cadre transactionnel de la classe ainsi que quelques autres informations, telles que son nom ou son adresse.

Depuis, la troisième version d'EJB permet de s'exonérer de cette fastidieuse configuration. Un nouveau principe est apparu, connu sous le nom d'annotations Java. Ce type de métadonnées permet de rendre le code auto-suffisant en embarquant en son sein tous les réglages auparavant listés dans les fichiers d'accompagnement.

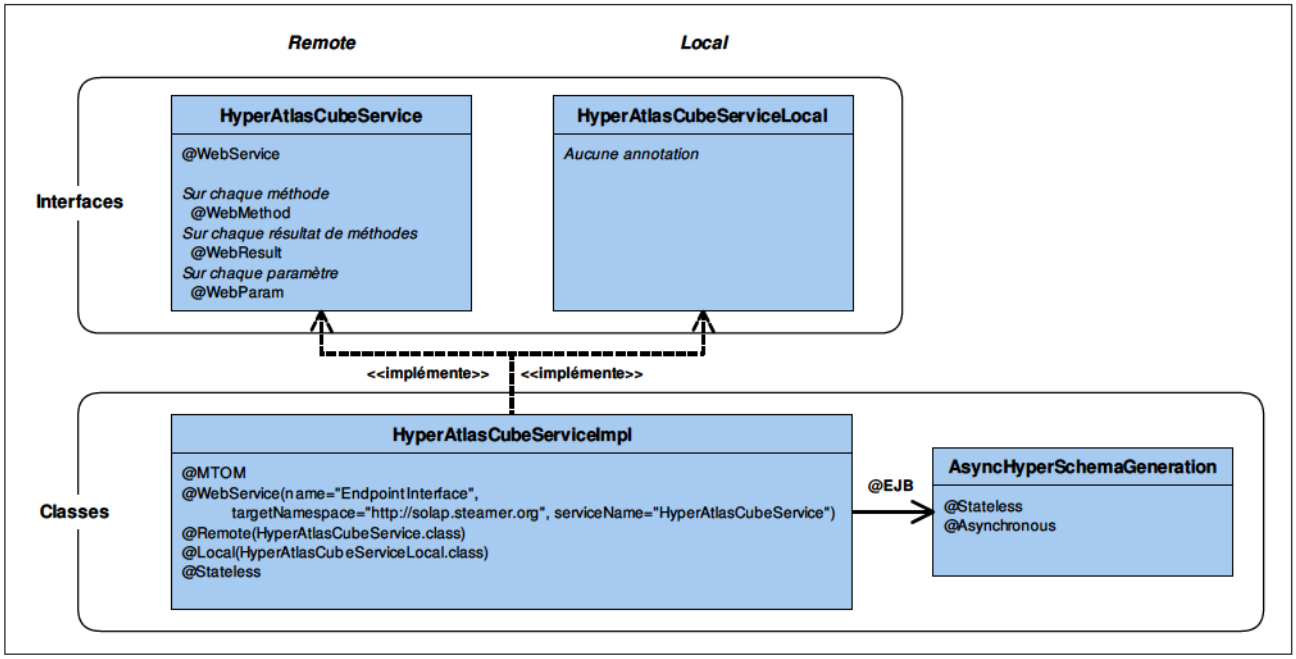



Figure IV-11 : Diagramme de classes simplifié référençant les annotations des interfaces et classes impliquées dans les services Web

La figure IV-11 illustre l'architecture de notre service. Du point de vue de Maven, ce module est subdivisé en deux. Le premier (partie supérieure) contient seulement les interfaces et une dépendance vers celui contenant les objets des services (non représentés ici). Le second (partie inférieure) recèle les implémentations réelles. Cette structure permet de réutiliser le module contenant les interfaces pour la construction du client (voir le chapitre « Client HyperAtlas³ »).

Considérons, sur cette même figure, uniquement les interfaces. À gauche se trouve l'interface `HyperAtlasCubeService` dédiée à l'accès distant des méthodes : elle offre l'exposition de nos services Web. À droite, `HyperAtlasCubeServiceLocal` n'est accessible que par RMI : cet accès, local, permet des performances accrues.

L'implémentation de ces deux interfaces est commune : il est inutile de scinder les méthodes car les comportements sont identiques. L'annotation `@MTOM` permet une optimisation de la bande passante utilisée pour le transfert de données binaires. Le *Message Transmission Optimization Mechanism* (MTOM) autorise en effet la transmission directe des pièces jointes, sans passer par un encodage en base 64, ce qui augmenterait substantiellement la taille du message. `@WebService` déclare les noms et espaces de nom du service. Ils seront utiles pour atteindre les méthodes en accès distant. `@Remote` et `@Local` rattachent les interfaces selon leur méthode d'accès. Enfin, `@Stateless` indique que la classe est « sans état », c'est-à-dire que d'un appel à un autre, aucun historique n'est préservé.

La version 3.1 des EJB, issue de la JSR318[42], apporte son lot de nouveautés, parmi lesquelles la simplification de la gestion asynchrone des méthodes EJB et la possibilité d'abandonner le recours aux interfaces locales.

Concernant l'asynchronie, il fallait auparavant faire appel à une gestion par file des messages, par exemple, via *Java Message Service* (JMS). Le principe « producteur/consomateur » de JMS réside en un producteur qui poste un message sur une file, tandis qu'un consommateur la parcourt et traite les messages qu'elle contient. La lecture de la file est complètement asynchrone vis-à-vis de l'écriture des messages, permettant en même temps de découpler les deux parties du système. Producteur et consommateur n'ont pas besoin de se connaître. Désormais, l'utilisation de l'annotation `@Asynchronous` sur une classe (ou l'une de ses méthodes) suffit à rendre asynchrone son appel. C'est le cas pour la classe `AsyncHyperSchemaGeneration`, comme on peut le constater sur la figure IV-11.

Enfin, nous n'avons pas abandonné l'utilisation des interfaces, et ce malgré la possibilité de le faire. Elles sont en effet un moyen efficace pour limiter le couplage client/serveur et facilite la testabilité des services.

Pour clore ce paragraphe, l'annotation `@EJB` permet l'injection de dépendance dans une classe. Grâce à elle, le développement est simplifié, rendant superflu l'usage des méthodes de localisation (JNDI) des EJB. Ceci n'est cependant possible que sur des éléments que le conteneur connaît.

La sous-section suivante s'arrête sur les méthodes exposées par le service.

### *iii. Génie logiciel*

Notre serveur expose trois méthodes relatives à l'exploration et l'exploitation du cube :

- `getProjects` : référencie les projets que le serveur connait. Il s'agit du reflet du contenu du fichier de configuration de l'application. Le message de retour contient une liste identifiant interne/nom public de chacun des projets ;

- `getSchemaInfo` : livre la méta-structure du projet dont l'identifiant a été passé en paramètre. Il s'agit de l'arborescence décrivant chaque schéma contenu dans le projet, des cubes aux membres ;

- `executeRequest` : applique la requête MDX au projet sélectionné puis retourne la réponse sous forme textuelle et, éventuellement, cartographique.

[42] *Java Specification Request*. JSR318 : http://jcp.org/en/jsr/detail?id=318.

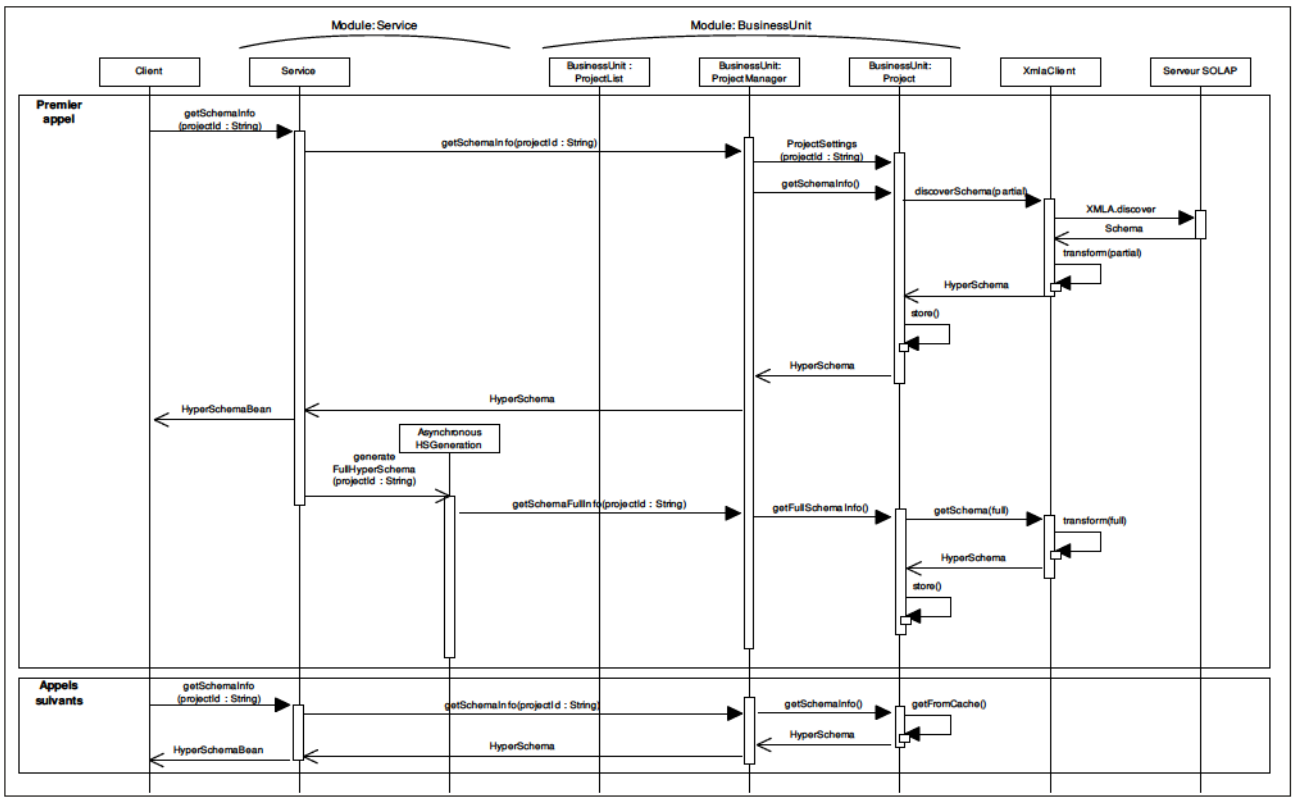



Figure IV-12 : Diagramme de séquence simplifié du service getSchemaInfo

Le service `getSchemaInfo` adopte un fonctionnement particulier, illustré par la figure IV-12. Les opérations de récupération et de retranscription du méta-schéma sont très consommatrices de temps et de ressources serveur, c'est pourquoi plusieurs mécanismes sont en place pour accélérer et mutualiser leurs utilisations.

Nous avons pu constater que le schéma était organisé hiérarchiquement : cubes, dimensions, hiérarchies, niveaux et membres. Cette structure peut être volumineuse : si l'on a un schéma à *n* cubes, ayant chacun *m* dimensions, *o* hiérarchies et *p* niveaux dans chacune et, pour finir, *q* membres par niveaux, on arrive à *n*m*o*p*q* actions pour achever l'extraction. Ce nombre peut être rapidement un souci car le temps de traitement est limité par le délai d'attente des clients du service Web (*timeout*). Pour palier à ce problème, la fourniture du méta-schéma est décomposée en deux temps. D'abord, une version allégée contient la partie constituée des éléments situés jusqu'aux niveaux : les membres, éléments les plus nombreux, sont donc ignorés. Dans un second temps, le processus reprend la première version et la complète.

Le premier appel à `getSchemaInfo` pour un projet donné déclenche la récupération et la constitution du schéma allégé. Ensuite, le service appelle de façon asynchrone l'EJB `AsyncHyperSchemaGeneration` qui va enrichir la version allégée. Enfin, le client reçoit cette première version, accompagné d'un indicateur précisant le caractère incomplet du modèle. Plus tard, lorsque `getSchemaInfo` sera à nouveau sollicité sur ce même projet, la version complète sera livrée directement, ce qui présente le second mécanisme introduit pour optimiser cette méthode : un cache. Ce dernier est géré en niveau du module « *BusinessUnit* » et sera exposé dans la section éponyme (page 86).

### *b. XmlaClient*

Le module « XmlaClient » constitue l'extrémité opposée aux services : si ces derniers fournissent aux clients les données transformées, le client XMLA apporte la matière première au système. Reçue sous forme d'un tableau à *n* dimensions, il transforme puis complète les informations des cellules

la constituant afin d'y ajouter des indications supplémentaires : les en-têtes de colonnes et de lignes qui la constitue, les valeurs voisines identiques, la présence d'une géométrie, etc. Cette étape prépare les opérations suivantes. Ainsi renseignées, les cellules pourront faire l'objet de démarches particulières : les données de celles contenant des géométries seront extraites en vue de générer la carte, les cellules contigües dont les valeurs sont égales peuvent être fusionnées, etc.

La figure IV-13 illustre la structure mise en place.

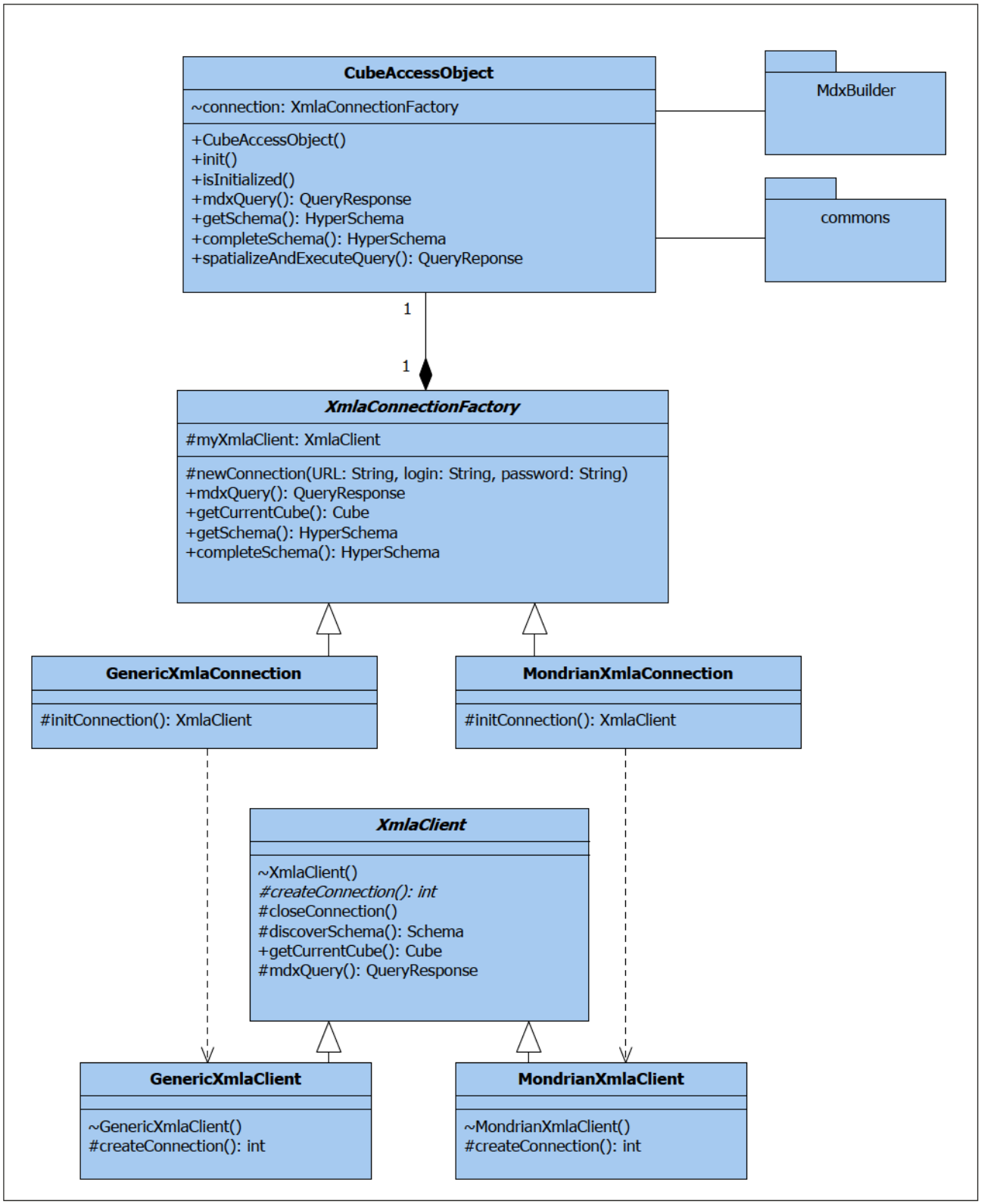



Figure IV-13 : Diagramme de classe partiel du module XmlaClient illustrant le patron de conception « fabrique »

Le client XMLA permet non seulement de gérer les connexions aux serveurs OLAP et SOLAP externes mais aussi de directement créer des liaisons vers des systèmes de gestion de bases de données hébergeant des entrepôts. En effet, ce module utilise le paquetage `olap4j`, édité par un groupement d'entreprises dont Pentaho, et le moteur de Mondrian est embarqué dans cette bibliothèque, permettant de gérer au sein de l'application, un serveur OLAP.

Nous avons modélisé le module en nous basant sur le patron de conception « fabrique » (*factory method*). Le but de ce patron est d'introduire une méthode abstraite de création d'un objet en reportant aux sous-classes concrètes la création effective. Puisque la classe qui instancie les objets ne peut connaître à l'avance le type précis de connexion, ce patron correspond à notre contrainte. De plus, si un nouveau procédé permettant la liaison entre l'application et une source de données devait être ajouté, il suffirait alors d'implanter les deux nouvelles classes correspondant à ce procédé.

`XmlaConnectionFactory` est, comme son nom l'indique, la fabrique de cette architecture. Créateur abstrait, cette classe impose les méthodes que les sous-classes devront implémenter. C'est notamment le cas de `initConnection` qui créé le connecteur. `GenericXmlaConnection` et `MondrianXmlaConnection` héritent de cette fabrique et réalisent la méthode d'initialisation de la liaison.

`XmlaClient` est le « résultat » de notre fabrique : dans notre cas, un client capable d'envoyer et de récupérer des informations auprès d'une source de données XMLA. Cette classe abstraite introduit plusieurs méthodes communes aux clients `GenericXmlaClient` et `MondrianXmlaClient` : les actions de découverte du schéma (`discoverSchema`), d'exécution des requêtes (`mdxQuery`) ou de fermeture de la liaison (`closeConnection`). Seule la création de la connexion est déléguée aux classes-filles car propre à chacun des comportements mis en jeu.

Enfin, la classe `CubeAccessObject` est la seule du paquetage à être accessible depuis l'extérieur. Elle se veut le seul point d'entrée : de ce fait, une instance de cette classe est rattachée à chacun des projets et agit comme une façade du processus de connexion vis-à-vis du reste de l'application.

### *c. Business unit*

Cœur du système, le module « métier » est imbriqué entre les services, qui exposent les méthodes mises à disposition des clients, et le client XMLA, permettant la connexion aux sources de données. Il fait aussi appel au générateur de cartes (« *MapBuilder* »), si les données récupérées dans le cube s'y prêtent, ou au constructeur de requêtes MDX, pour « spatialiser » certaines demandes.

#### *i. Sources multiples*

Nous avons présenté, en introduction de ce chapitre, le fichier de propriétés offrant les options régissant le comportement du serveur. Dans ce document se trouvent les paramètres de connexion aux serveurs livrant leurs données, autrement dit, les sources. L'ensemble formé de cette configuration et du moyen d'accéder aux données est considéré comme un « projet », et ce module est en charge de la gestion des différentes instances de ces projets. En effet, une première connexion sur un serveur (S)OLAP créée une instance dédiée à ce serveur. Ensuite, les informations le concernant sont chargées du fichier de propriétés puis la méta-structure y est stockée après le premier appel au service `getSchemaInfo` .

Si l'accès simultané à différents serveurs n'est pas possible, l'utilisateur peut néanmoins interroger, tour à tour, les différents cubes. Tous profiteront des méthodes et processus proposés par le serveur HyperAtlas³, notamment l'aspect cartographique, et ce, dans le même client, à condition que celui-ci le permette.

### *ii. Décision de spatialisation des requêtes*

Si l'action de « spatialisation » à proprement parler est déléguée au module « *MdxBuilder* », elle est déclenchée par l'unité métier. La figure IV-14 ci-dessous illustre la démarche. Le processus est déclenché par l'appel au service *executeRequest*. Celui-ci accepte plusieurs paramètres parmi lesquels on trouve l'identifiant du projet sur lequel le client souhaite agir, la requête MDX sous forme de chaîne de caractères ou d'un objet dédié (décrit dans la section « *MdxBuilder* », page 88) et un niveau spatial de référence (SLR, *Spatial Level Reference*). Complétant cette liste, de nombreuses options permettent la personnalisation de la carte, des indicateurs à afficher et leur style.

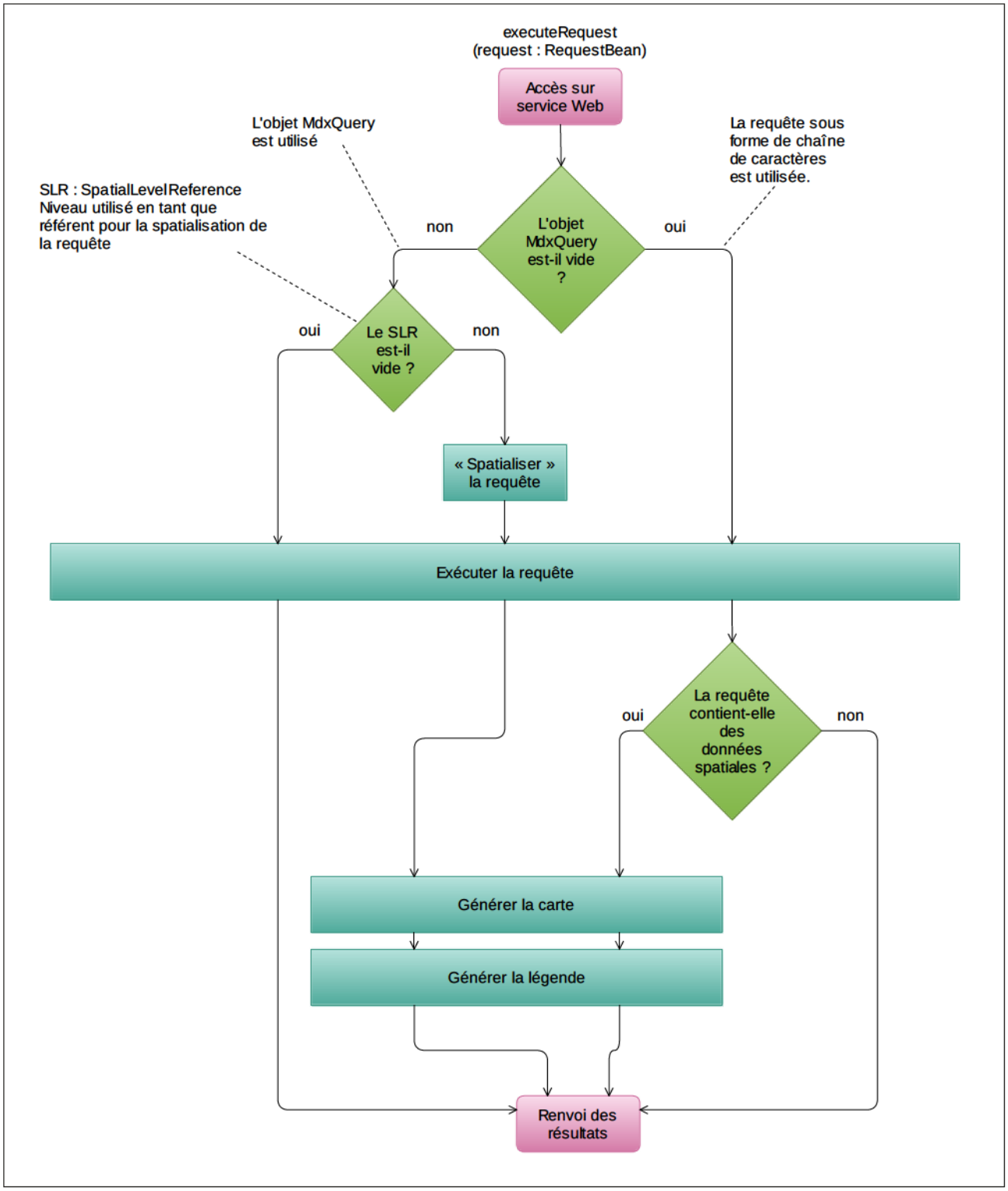



Figure IV-14 : Gestion d'une requête par le service executeRequest

Compte tenu des paramètres qui lui sont transmis, le système peut fonctionner selon deux modes.

Tout d'abord, le comportement optimal est déclenché par la présence simultanée de l'objet `MdxQuery` et du niveau spatial de référence. L'objet propriétaire `MdxQuery` contient une version de la requête MDX qui en permet une manipulation aisée. Créé sur mesure pour répondre à la problématique MDX, il est nécessaire pour une gestion complète et optimale : des éléments peuvent être ajoutés, intercalés, déplacés ou supprimés dans la requête. Le SLR fait partie du nombre. Celui-ci permet d'ajouter à la requête les éléments nécessaires à cette « spatialisation » : préfixe de requête et alias sur une colonne supplémentaire récupérée.

Le second mode est plus orienté vers une exploitation experte du service. Dans ce cas, seule la chaîne de caractères constituant la requête MDX est prise en compte. Aucune modification amont n'est effectuée : une fois vérifiée, elle est directement soumise au serveur. Cette utilisation permet un usage plus permissif de l'application mais la contrepartie réside dans le fait qu'elle n'est pas aidée et nécessite une connaissance approfondie de MDX.

#### *iii. Cache*

Nous avons évoqué le fait que la récupération de la méta-structure était très grande consommatrice de ressources (section « Service »). Pour répondre à cette problématique, l'extraction et la livraison de cette structure a été décomposée en deux temps : une version partielle mais suffisante pour constituer des requêtes basiques et une version complète.

Pour compléter ce dispositif, un système de cache mutualise l'effort nécessaire à sa constitution. Que ce soit pour la version allégée ou complète, lorsque l'une des deux existe, elle sera fournie directement depuis le cache de l'application. Cette technique part du principe que la structure des cubes ne peut changer à travers le temps. Si ce doit être le cas, notamment lorsqu'un ETL a mis à jour le contenu des dimensions, le cache de l'application devra être reconstitué.

Donc, pour chaque projet, ce cache est construit. Les méta-structures sont élaborées lors du premier appel et sont ensuite livrées directement et sans délai.

### *d. MDXBuilder*

La construction d'une requête MDX nécessite des connaissances spécifiques, d'autant plus dès que l'on doit faire usage des fonctions spatiales. Néanmoins, l'une des règles de l'informatique décisionnelle stipule que l'utilisateur ne doit pas posséder ce type de connaissance pour exploiter les outils mis à sa disposition. Il faut alors que la partie génération de requête soit effectuée par le logiciel de façon transparente pour les usagers de l'application. De plus, comme nous l'avons évoqué dans la section « *BusinessUnit* », dans certains cas, la requête doit se parer d'opérateurs supplémentaires afin de revêtir sa dimension spatiale.

Le module « *MDXBuilder* » se charge de tout ce qui a trait à la requête MDX et les aspects décrits précédemment sont deux des principaux objectifs qu'il remplit. Dans ce système, il va sans dire qu'il s'agit d'un élément crucial.

#### *i. Architecture*

Une requête MDX est constituée de plusieurs parties et de multiples opérateurs, comme nous avons pu le constater dans notre état de l'art. Afin de profiter de la structure de ces requêtes, nous l'avons recréé dans notre système, comme l'illustre la figure IV-15.

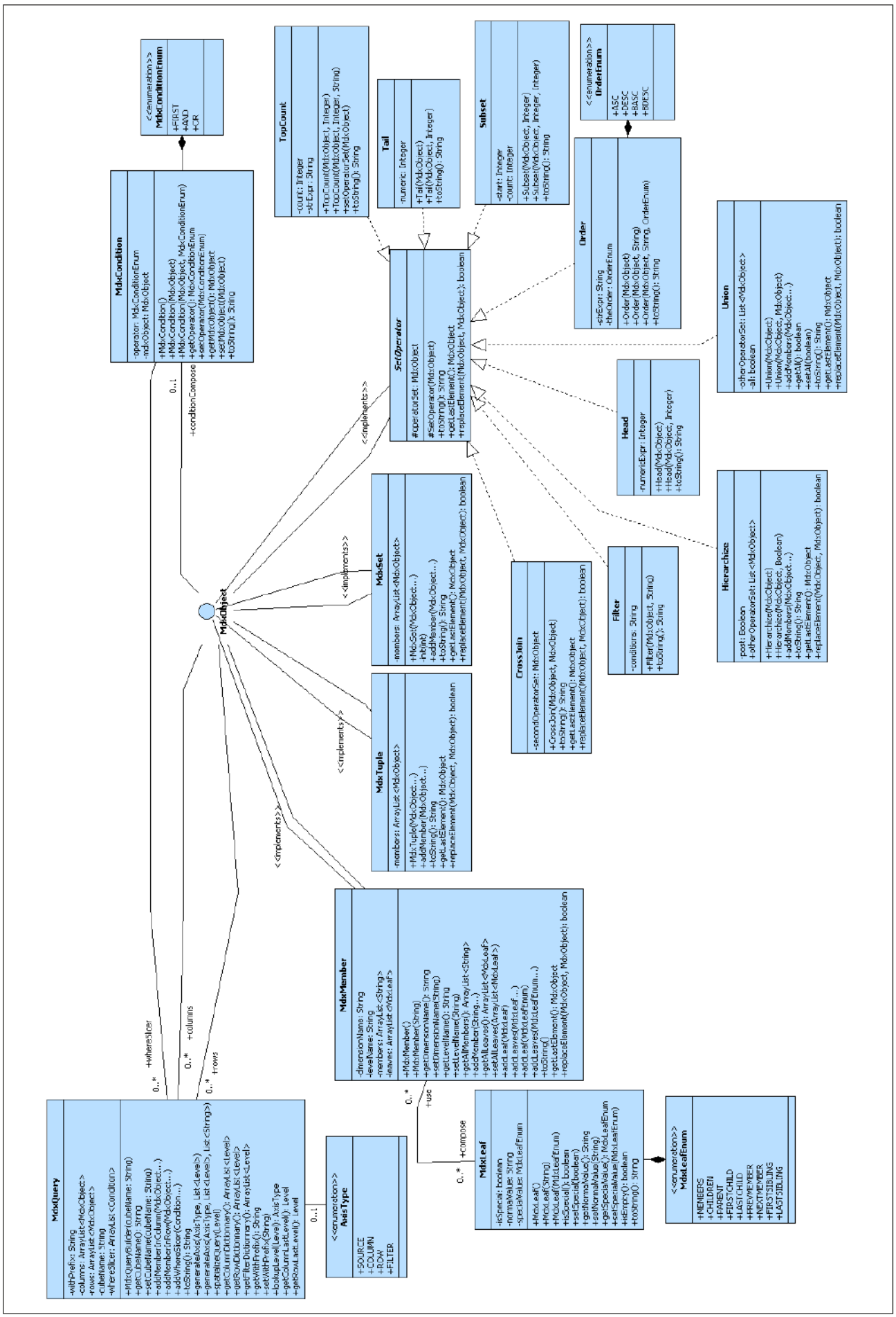



Figure IV-15 : Diagramme de classes du module « MDXBuilder »

Le point d'entrée du module est la classe `MDXQuery`, située en haut à gauche sur la figure IV-15. Les éléments constituant la requête lui sont ajoutés au fur et à mesure de leur création. Profitant du polymorphisme inhérent à Java, toutes les classes étendent l'interface `MdxObject`. En effet, que ce soit en ligne ou en colonne, les objets utilisés sont toujours de ce type, peu importe leur complexité. Il est en effet impossible de savoir à l'avance si l'on doit avoir recours à un membre, un tuple ou un *set* ou à tout autre opérateur pour la construction de la requête.

Ce module a été entièrement écrit et validé grâce à la technique du développement piloté par les tests (*Test Driven Development* TDD). Cette méthode se décompose en un cycle de cinq étapes :

- Écrire un nouveau test ;
- Vérifier qu'il échoue, le code à tester n'existant pas encore ;
- Ajouter la fonctionnalité ;
- Constater le succès du test ;
- Remanier le code pour éviter les répétitions.

TDD se prête très bien à notre cas de figure. Connaissant le résultat escompté, une requête MDX, il est aisé de rédiger les tests selon les niveaux sélectionnés en entrée. Le code métier permettant la génération de la requête est ensuite rédigé. Une nouvelle exécution du test valide le fonctionnement unitaire du code tout juste implanté. Pour finir, les techniques de remaniement (*refactoring*) donnent l'occasion de généraliser et consolider les instructions écrites en évitant notamment leur duplication. Les parties du code employées pour produire une requête étant très largement réutilisées d'une exécution à une autre, jouer tous les tests permet d'attester de la non-régression des autres fonctionnalités et du bon fonctionnement global du module. Ces tests, comme tous les autres, sont construits grâce au cadriciel JUnit. Grâce à l'étape test du cycle de Maven, à chaque compilation, ils sont tous exécutés et le défaut d'un d'entre eux entraîne l'annulation de la construction du paquetage, obligeant l'équipe de développement à apporter un correctif.

### *ii. Construction automatique des requêtes*

Parmi les règles sous-tendant l'informatique décisionnelle, il en est une qui prône la simplicité de l'outil, du point de vue de l'utilisateur, exonérant celui-ci d'avoir à acquérir des connaissances informatiques particulières. Le logiciel est entièrement tourné vers l'analyste et toutes les contraintes techniques lui sont abstraites.

Ceci peut paraître anodin, à première vue, mais du point de vue du développement, cette relative simplicité va rapidement se révéler être un défi. Le challenge réside essentiellement dans la construction de la requête MDX, reflet de la sélection de l'utilisateur.

Le processus de sélection se résume à choisir un niveau et l'affecter à un axe : colonne, ligne ou filtre. Derrière cette manipulation se passent, en coulisses, de nombreuses vérifications et actions. Comme tout langage, MDX est soumis à des règles. Certaines ont un impact direct sur les possibilités de maniement de l'interface et doivent donc être vérifiées au niveau du client (voir page 104). D'autres, en revanche, ne concernent que l'élaboration de la requête. Nous focalisons ici sur cette deuxième catégorie.

`MdxQuery`, via sa méthode `generateAxis`, accepte, outre l'axe sur lequel travailler, la liste de niveaux, c'est-à-dire la sélection de l'utilisateur. L'application se charge ensuite de construire la combinaison des instances des objets formant la requête.

Les interactions des opérateurs spécifiques, tels que `CrossJoin`, `Ancestor` ou `Filter`, sont une mécanique quasi-formelle et les cas qui en découlent sont autant de règles :

1. Un niveau, pas de membre sélectionné. Il s'agit du cas le plus simple. Le niveau est inclus à l'axe et une feuille `.members` lui est adjointe, permettant l'affichage de tous les membres qu'il contient ;

*Sélection de l'utilisateur :* niveau 0 de la dimension spatiale NUTS.

```
[NUTS].[NUTS niveau 0].members
```

2. Un niveau, dont des membres sont sélectionnés. Un *set* est construit auquel tous les membres sont ajoutés par leur nom unique ;

*Sélection de l'utilisateur :* niveau 0 de la dimension NUTS, membres France et Allemagne.

```
{[NUTS].[France], [NUTS].[Allemagne]}
```

3. Plusieurs niveaux de dimensions différentes, aucun membre sélectionné. Les niveaux doivent être imbriqués grâce à la fonction `CrossJoin`, ce qui a pour résultat le produit cartésien des membres de chaque niveaux ;

*Sélection de l'utilisateur :* niveau 0 des NUTS, niveau 1 des CLC.

```
CrossJoin
(
  [NUTS].[NUTS niveau 0].members,
  [CLC].[CLC niveau 1].members
)
```

4. Plusieurs niveaux de dimensions différentes, quelques membres sélectionnés. Il s'agit d'un mélange des deux précédentes règles : les niveaux dont des membres ont été sélectionnés sont générés en tant que *set* contenant lesdits membres, puis ils sont croisés grâce à l'opérateur `CrossJoin` ;

*Sélection de l'utilisateur :* niveau 0 de la dimension NUTS, membres France et Allemagne, niveau 1 des CLC

```
CrossJoin
(
  {[NUTS].[France], [NUTS].[Allemagne]},
  [CLC].[CLC niveau 1].members
)
```

5. Plusieurs niveaux d'une même dimension, aucun membre sélectionné. Les niveaux partageant une dimension unique doivent être classés dans l'ordre de leur profondeur (par exemple, pour la dimension temporelle, le niveau année précède celui des mois, précédant lui-même celui des jours), ajoutés à un *set*, puis hiérarchisés grâce à l'instruction `Hierarchize` ;

*Sélection de l'utilisateur :* niveau 0 des NUTS, niveau 1 des NUTS

```
Hierarchize
(
  {
    [NUTS].[NUTS level 0].members,
    [NUTS].[NUTS level 1].members
  }
)
```

6. Plusieurs niveaux de dimensions identiques, quelques membres sélectionnés. Une fois encore, les niveaux dont des membres sont sélectionnés sont créés en tant que *set*. L'opérateur `Hierarchize` permet l'imbrication des niveaux entre eux. Deux nouvelles fonctions sont utilisées pour rendre cohérent la sélection : `Filter` et `Ancestor`. En effet, en sélectionnant un membre du niveau le plus haut, les niveaux inférieurs doivent être filtrés. Prenons l'exemple sur la dimension spatiale : les niveaux « Pays » et « Régions » sont sélectionnés. Sans membre, toutes les régions de chaque pays seront affichées. Si « France » est sélectionné au niveau pays, seules les régions françaises doivent être affichées. C'est la mission de `Filter`. `Ancestor` quant à lui renvoie l'ancêtre du membre sélectionné ;

*Sélection de l'utilisateur :* niveau 0 des NUTS, membre France, niveau 1 des NUTS

```
Hierarchize
(
  ([NUTS].[France]),
  Filter
  (
    {[NUTS].[NUTS niveau 1].members},
    (
      Ancestor
      (
       [NUTS].CurrentMember,
       [NUTS].[NUTS niveau 0]
      )
      IN ( [NUTS].[France] )
    )
  )
)
```

7. Plusieurs niveaux, certains partageant la même dimension, d'autres pas, dont certains ont des membres sélectionnés. Toutes les précédentes règles sont appliquées, à des degrés différents.

*Sélection de l'utilisateur :* niveau 0 des NUTS, membre France, niveau 1 des NUTS, niveau 1 des CLC

```
Hierarchize
(
  Union
  (
    CrossJoin
    (
      [NUTS].[France],
      [CLC].[CLC level 1].members
    ),
    CrossJoin
    (
      Filter
      (
        { [NUTS].[NUTS level 1].members },
        (
```

```
            Ancestor
            (
              [NUTS].CurrentMember,
              [NUTS].[NUTS level 0]
            )
            IN ( [NUTS].[France] )
          )
        )
      )
    )
)
```

Une fois les règles isolées et implémentées, la construction de la requête n'a plus qu'à suivre le processus décrit par la figure IV-16.

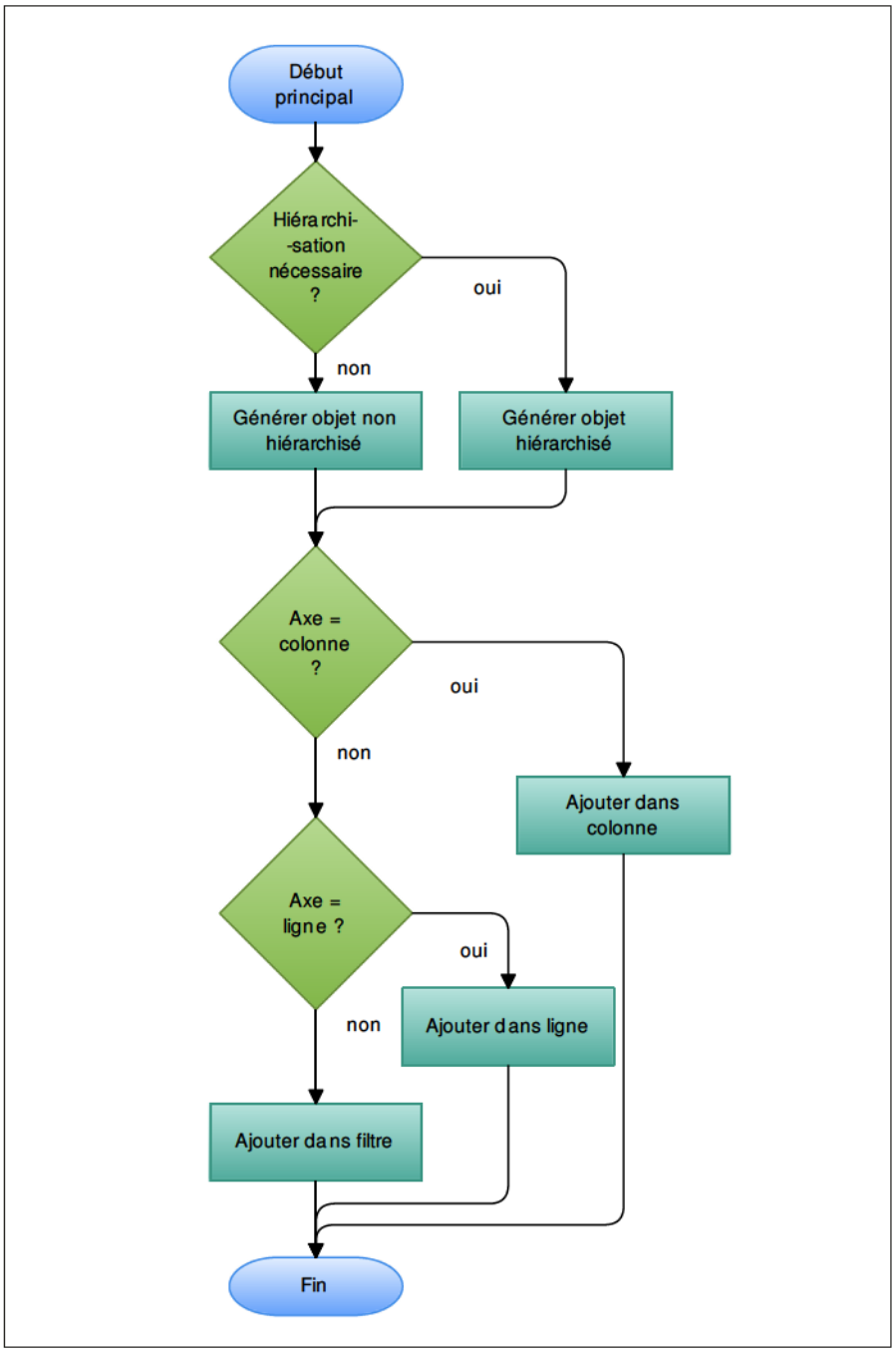



Figure IV-16 : Diagramme de flux simplifié pour la construction d'un axe

### *iii. Spatialisation*

Bien que l'entrepôt contienne des données géoréférencées ainsi que les formes des géométries les hébergeant, rien, pour le moment, ne permet d'en exploiter les capacités. C'est ce à quoi répond la méthode `spatializeQuery`.

Le terme « spatialisation » désigne l'action d'envoyer dans l'espace[43]. Dans notre cas, il s'agit uniquement d'activer spatialement la requête MDX générée suite aux choix de l'utilisateur. Ceci dans le but de situer la donnée dans son référentiel spatial. Pour réaliser cette transformation, il faut modifier quelques éléments de la requête MDX.

En plus de la requête initiale, l'outil a besoin d'un référent sur lequel se baser pour ensuite dessiner les formes, le « niveau spatial de référence ». Ce niveau est nécessaire pour définir quelles sont les données géométriques à utiliser en tant que référentiel. Dans le cas où seule une dimension spatiale est impliquée, la solution est triviale. Mais dès que l'on en croise plusieurs, le choix est laissé à l'appréciation de l'utilisateur.

Pour mémoire, une requête MDX est constituée de quatre parties. La première contient les membres calculés. Elle est optionnelle et commence par le mot-clé WITH. Les axes viennent ensuite, généralement au nombre de deux, introduits par SELECT. Suit la désignation de l'aire d'étude, préfixée de FROM. Enfin, débutant par WHERE, le filtre clos la requête. L'opération de « spatialisation » touche les deux premières.

L'idée est donc d'ajouter un membre contenant les données géométriques correspondantes au niveau de référence. Si l'on veut ajouter la géométrie d'un niveau utilisé en ligne, celle-ci devra être ajoutée en colonne, et inversement. Prenons un exemple pour illustrer ces propos. L'utilisateur demande l'affichage du nombre d'habitants, réparti par pays (en lignes) et par environnement CLC de niveau 1 (en colonnes). En ajoutant la géométrie comme convenu, une nouvelle colonne sera adjointe, comme le montre la figure IV-17 suivante (le nombre de lignes est volontairement limité à quelques enregistrements).

| | | | | | | Geom |
|---|---|---|---|---|---|---|
| NUTS2006 level 0 | Artificial surfaces | Agricultural areas | Forest and semi natural areas | Wetlands | Water bodies | |
| Belgium | 1,085,974 | 1,006,993 | 1,269,565 | 439,948 | 479,904 | POLYGON ((-648193.1574175087 133333.9131855... |
| Bulgaria | 689,219 | 741,526 | 910,343 | 360,187 | 340,542 | MULTIPOLYGON (((286832.5452936192 1119307.7... |
| Estonia | 100,814 | 113,414 | 177,070 | 58,162 | 90,968 | POLYGON ((-750871.133157 333.9131855731,936... |
| Finland | 451,270 | 409,068 | 642,212 | 261,062 | 359,285 | POLYGON ((-648193.1574175087 133333.9131855... |
| France | 2,296,649 | 2,399,645 | 2,785,701 | 1,135,035 | 959,821 | MULTIPOLYGON (((-481726.3937627985 -939114.2... |
| Germany | 9,495,544 | 11,464,370 | 12,559,425 | 5,132,731 | 4,812,022 | POLYGON ((-537626.164812003 275348.69712913... |
| Greece | 1,055,260 | 1,245,589 | 1,233,971 | 611,869 | 597,229 | MULTIPOLYGON (((-939114.285094 481726.39376... |
| Ireland | 214,499 | 186,341 | 228,384 | 74,916 | 67,457 | POLYGON ((-648193.1574175087 133333.9131855... |
| Italy | 2,739,798 | 2,867,558 | 3,562,426 | 1,352,580 | 1,356,249 | MULTIPOLYGON (((-913185.3937627985 -939114.2... |

Figure IV-17 : Extraction des géométries des pays (en lignes) sur la dernière colonne (Geom)

Cependant, il y a une règle à observer. Elle consiste à interdire toute utilisation de deux niveaux ou membres d'une même dimension sur plus d'un axe. Cette précision a pour conséquence qu'il est donc impossible, dans notre cas, d'ajouter sur l'axe des colonnes la géométrie des pays, de la dimension NUTS, alors que celle-ci est déjà utilisée sur l'axe des lignes.

Pour déjouer cette contrainte, il nous faut passer par la construction d'un nouveau membre, grâce à la première partie de la requête MDX. Puis nous devrons l'ajouter dans la requête. Ce nouveau membre sera chargé d'extraire la géométrie mais sera référencé comme le dernier élément de l'axe

[43] Dictionnaire Larousse en ligne. http://www.larousse.fr/dictionnaires/francais/spatialiser

opposé. Ainsi, le mélange des dimensions est évité. La procédure suit le pseudo-algorithme suivant :

1. Trouver l'axe contenant le niveau spatial de référence ;
2. Récupérer le dernier élément de l'axe opposé au niveau spatial ;
3. Fabriquer un nouvel objet destiné à contenir la géométrie ;
4. Renseigner le préfixe ;
5. Générer un nouveau set contenant le dernier élément de l'axe opposé et l'objet géométrie (fabriqué à l'étape 3) ;
6. Ajouter cet élément en lieu et place du dernier objet, trouvé à l'étape 2.

Toujours en reprenant l'exemple utilisé au départ, la requête initiale est la suivante :

```
SELECT
{
  ([CLC2000].[CLC2000 level 1].MEMBERS)
} ON COLUMNS,
{
  ([NUTS2006].[NUTS2006 level 0].MEMBERS)
} ON ROWS
FROM [Indicators]
```

En fin de processus, elle a été modifiée pour devenir celle-ci :

```
WITH  member [CLC2000].[Geom]
as 'ST_UnionAgg([NUTS2006].[NUTS2006 level 0].currentMember.children, «geom»)'
SELECT
{
  {
    ([CLC2000].[CLC2000 level 1].MEMBERS), ([CLC2000].[Geom])
  }
} ON COLUMNS,
{
  ([NUTS2006].[NUTS2006 level 0].MEMBERS)
} ON ROWS
FROM [Indicators]
```

Le nouvel élément créé, nommé `[CLC2000].[Geom]` (représenté en violet), contient les géométries des pays (`[NUTS2006].[NUTS2006 level 0]`). Celui-ci est placé en dernière position sur l'axe opposé à celui du niveau spatial de référence, au sein d'un nouveau *set*. Tous les autres ajouts sont marqués par la couleur bleu-vert.

Grâce à l'opérateur spatial `ST_UnionAgg`, la géométrie est extraite sous une forme textuelle (*Well-Known Text* WKT) directement compréhensible par le module « *MapBuilder* ».

L'opération de « spatialisation » n'est effectuée que sur l'objet `MdxQuery`. Rien n'empêcherait, à priori, d'effectuer la manœuvre sur la chaîne de caractères, bien que ce soit plus contraignant, mais nous avons pris le parti de considérer les utilisateurs de ce canal comme des experts. De ce fait, aucune modification n'est effectuée sur la requête communiquée, elle est directement exécutée auprès du serveur.

### *e. MapBuilder*

Dernière brique de notre édifice, mais pas des moindres, le module de génération de cartes est en charge d'interpréter les données récupérées, de tracer les contours des formes impliquées, de représenter les valeurs des indicateurs choisis en effectuant au préalable les calculs nécessaires à leur bonne visualisation.

« *MapBuilder* » a été écrit dans le but d'être facilement transposable et réutilisable dans d'autres projets. De ce fait, les entrées sont standardisées, alignées sur les artefacts proposés par notre prédécesseur, Laurent Poulenard, pour son projet HyperAtlas WMS, effectué lors de son mémoire CNAM [Poulenard, 2011].

Les sorties, quant à elles, sont conventionnelles : deux images, résultats du traitement, représentant respectivement la carte et sa légende.

#### *i. Interprétation des données*

Pour pouvoir tracer les géométries et les indicateurs demandés, le module doit d'abord reconnaître les différentes informations dont il a besoin. La matrice de données constituant la réponse fournie par l'entrepôt, transférée depuis le module « *XmlaClient* » à celui-ci, via « *BusinessUnit* », contient ces données. Dans cette structure, un travail préalable a été opéré pour différencier les géométries des données attributaires. Selon les niveaux et les axes choisis, les informations ne se trouvent pas au même endroit, comme l'illustre la figure IV-18.

| NUTS2006 level 0 | Artificial surfaces | Agricultural areas | Forest and semi natural areas | Wetlands | Water bodies | |
|---|---|---|---|---|---|---|
| Belgium | 1,085,974 | 1,006,993 | 1,269,565 | 439,948 | 479,904 | POLYGON ((-648193.1574175087 133333.9131855731, |
| Germany | 9,495,544 | 11,464,370 | 12,559,425 | 5,132,731 | 4,812,022 | POLYGON ((-537626.164812003 275348.69712913345, |
| France | 2,296,649 | 2,399,645 | 2,785,701 | 1,135,035 | 959,821 | MULTIPOLYGON (((-481726.3937627985 -939114.2850947428, |
| Italy | 2,739,798 | 2,867,558 | 3,562,426 | 1,352,580 | 1,356,249 | MULTIPOLYGON (((47631.62509300179 -1301608.5315753499, |

| CLC2000 level 1 | Belgium | Germany | France | Italy |
|---|---|---|---|---|
| Artificial surfaces | 1,085,974 | 9,495,544 | 2,296,649 | 2,739,798 |
| Agricultural areas | 1,006,993 | 11,464,370 | 2,399,645 | 2,867,558 |
| Forest and semi natural areas | 1,269,565 | 12,559,425 | 2,785,701 | 3,562,426 |
| Wetlands | 439,948 | 5,132,731 | 1,135,035 | 1,352,580 |
| Water bodies | 479,904 | 4,812,022 | 959,821 | 1,356,249 |
| | POLYGON ((-648193.1574175087 133333.9131855731, | POLYGON ((-537626.164812003 275348.69712913345, | MULTIPOLYGON (((-481726.3937627985 -939114.2850947428, | MULTIPOLYGON (((47631.62509300179 -1301608.5315753499, |

**Figure IV-18 : Emplacements des informations de géométries : en haut, sur les lignes ; en bas, sur les colonnes**

Pour chaque ligne, lorsque la géométrie est en colonne (ou pour chaque colonne, lorsque la géométrie est en ligne), une unité spatiale est construite contenant à la fois la géométrie sous forme WKT et les données attributaires. Pour prendre un exemple à partir de la figure IV-18, l'unité spatiale « France » contient, outre sa forme et son nom, les données biophysiques de type CLC dans un tableau de correspondance du type clé/valeur : « *Artificial surfaces* » = 2 296 649, « *Agricultural areas* » = 2 399 645, etc.

Les données sont parcourues différemment selon la forme que revêt la matrice d'origine. Lors de ce parcours, les valeurs extrêmes (minimum et maximum) sont relevées et la somme est calculée par colonne (ou ligne) et par unité pour les besoins ultérieurs des représentations.

En fin de traitement, les géométries sont générées sous forme d'objets `shape`, objets qui constituent les parties graphiques utilisées par l'outil de tracé. Le centre de chaque forme est déterminé lors de ce même processus. Dans le cas de formes constituées de plusieurs polygones, cas de la France par exemple (France continentale et Corse), le polygone dont la surface est la plus grande est gardé pour en définir le centre. Cette manipulation évite de placer le « centre » au-dessus d'un autre pays ou d'une étendue d'eau.

Les unités spatiales construites sont stockées dans une classe contenant en outre toutes les informations nécessaires à l'élaboration de la carte : dimensions de l'image, système de référence spatial, le cadre définissant la zone visible (appelé *boundary box* ou BBox), le type de carte à tracer (indicateur ou ratio), la représentation selon le type (disques à taille proportionnelle, circulaire, crête de coq).

### *ii. Remaniement d'HyperAtlas*

Pour son mémoire, [Poulenard, 2011] avait travaillé au remaniement et à l'extraction du code nécessaire à la génération cartographique depuis HyperAtlas. Ses travaux l'ont mené à organiser ses services REST WMS en deux parties. La première contient les interfaces et les classes abstraites fondées sur les objets résolument tournés vers une architecture à base de servlets : `HTTPServletRequest` et `HTTPServletResponse`. La seconde apporte la matière nécessaire à leur concrétisation. Nous avons à notre tour remanié cet existant afin d'y intégrer nos contraintes et y avons aussi apporté un travail supplémentaire.

Les objets dérivés des servlets ne peuvent nous convenir puisqu'en utilisant la technologie EJB, nos propres objets ont déjà été extraits, de façon transparente, au niveau des services. Nous avons donc développé une nouvelle collection de classes abstraites pour remplacer l'existante, basée, cette fois, sur notre structure.

Cette substitution a permis de réutiliser, à peu de frais, la partie concrète. Celle-ci est en charge de réellement générer une image représentant une carte : calculs préalables, tracé des formes et application des styles. La sous-section suivante s'arrête sur les principes entrant en jeu dans la création cartographique ainsi que sur les évolutions que nous avons apportées au module existant.

### *iii. Génération de carte*

Grâce à la collection d'unités spatiales et aux attributs dédiés aux paramétrages, il est possible de générer une carte conforme aux attentes tant graphiques que d'analyses de l'utilisateur.

Un ordonnancement des tâches à effectuer est nécessaire, sans quoi, la carte serait illisible. C'est pourquoi elle est générée par couches successives, du fond vers le premier plan.

1. D'abord, les contours des unités spatiales sont tracés. Une couleur neutre, les désignant comme faisant partie de l'aire d'étude, est ajoutée en fond de chacune ;
2. Dans le cas d'une carte choroplèthe, un nouveau fond est donné aux unités, dont la couleur correspond au résultat du calcul du ratio des indicateurs sélectionnés ;
3. Dans le cas d'une carte à symboles de tailles proportionnelles, le centre de chaque unité est décoré d'une forme permettant la représentation d'un ou plusieurs indicateurs ;
4. Pour finir, la légende est dessinée, selon le type de carte.

Par les principes même du SOLAP et du mode d'extraction des géométries choisi, un fond de carte mettant en jeu les unités spatiales non sélectionnées n'est pas possible. En effet, seuls les membres de la sélection de l'utilisateur sont extraits de l'entrepôt lors de l'exécution de la requête. Pour obtenir le fond de carte permettant une mise en situation plus aisée, une seconde requête

serait nécessaire, alourdissant le processus. Or, nous avons pris le parti de répondre au plus vite aux attentes de l'utilisateur, comme l'illustre la figure IV-19.

Les étapes 2 et 3 sont dédiées à la représentation des données sélectionnées par l'utilisateur. Elles sont le sujet de la sous-section suivante.

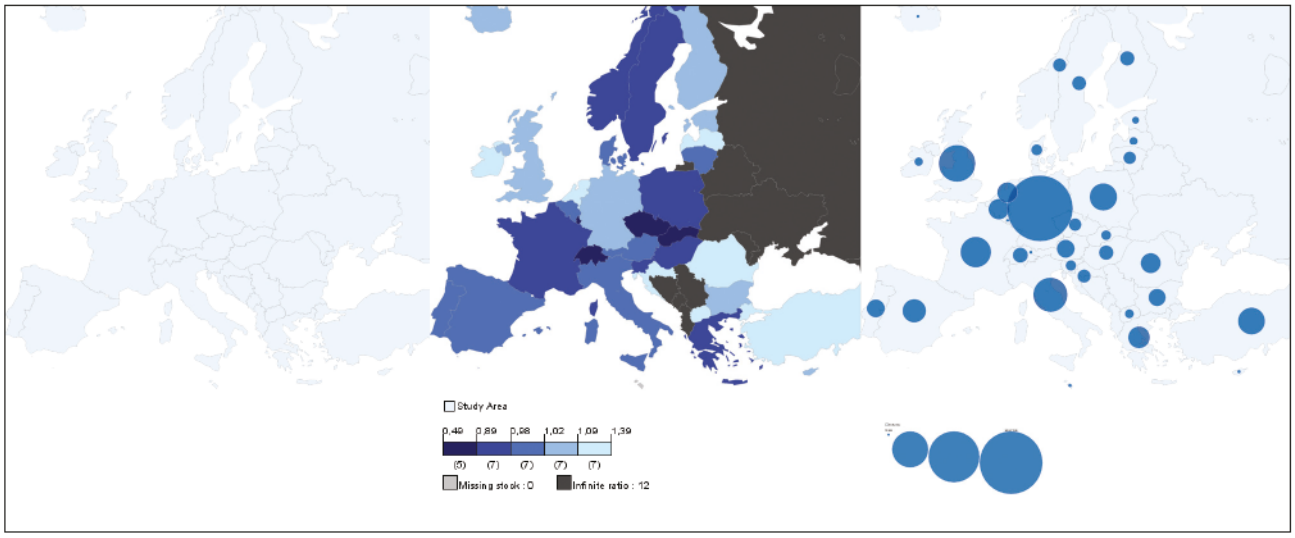



Figure IV-19 : Etapes de génération d'une carte. D'abord, fond de carte ; puis aplats des cartes choroplèthes, symboles de tailles proportionnelles ; enfin légende

### *iv. Représentation des données*

Sur une carte, les données peuvent être représentées de plusieurs manières, notre état de l'art en a dressé une liste. Notre module, essentiellement basé sur le code d'HyperAtlas, donne accès à deux de ces procédés : les cartes à symboles de taille proportionnelle et les cartes choroplèthes.

#### Cartes à symboles de taille proportionnelle

Ce type de carte met en jeu des symboles faisant sens dans le contexte du message que l'utilisateur veut communiquer. Ceux-ci ont une taille proportionnelle à la valeur qu'ils représentent.

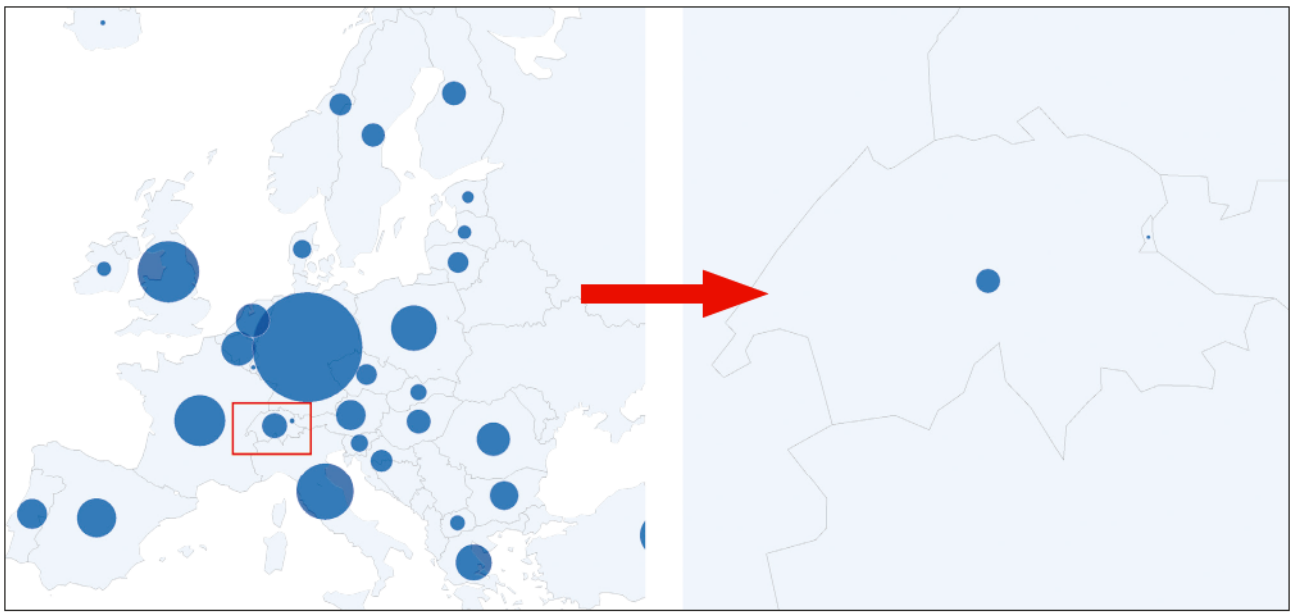


Figure IV-20 : Recalcul de la surface occupée par les indicateurs lors d'un grossissement

Assis sur les valeurs extrêmes (minimum et maximum) relevées lors de la génération des unités spatiales, ainsi que sur les résultats des sommes effectuées lors de ce même processus, le calcul des surfaces allouées aux symboles considère l'espace total visible de la carte et les données à représenter. En effet, les symboles étant de tailles dynamiques, il nous incombe de définir la surface visuelle occupée maximale réservée à ces symboles. Trop petite, les symboles seraient illisibles et les différences entre les valeurs seraient infimes rendant inutile la représentation. Trop grande, les symboles se chevaucheraient tous, occultant par la même les contours des unités spatiales. Pour finir, quand l'utilisateur opère un grossissement de la zone d'étude (zoom), les unités spatiales encore visibles doivent être agrandies. Il en va tout autrement des symboles : le taux de grossissement est inversement appliqué au calcul de la taille des symboles, sans quoi ils occuperaient une surface proportionnellement plus élevée (figure IV-20 ci-contre).

Directement extraite des sources existantes, la représentation des données par des disques, dont la circonférence est fonction de la valeur représentée, fait partie des productions cartographiques les plus communément rencontrées. Elle permet l'affichage d'un unique indicateur dont la comparaison est aisée entre les unités spatiales. La figure IV-21 montre un exemple de cette représentation au sein d'HyperAtlas³.

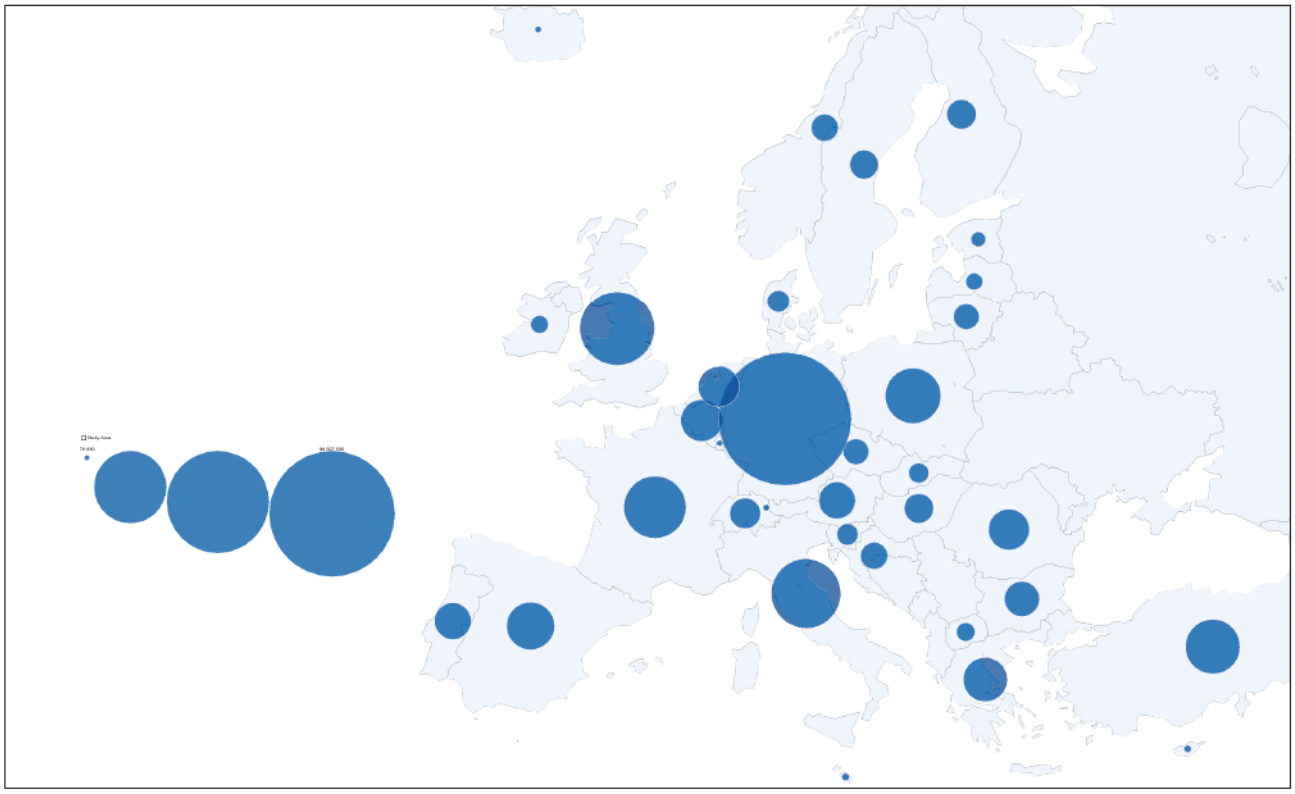


**Figure IV-21 : Carte à disques de taille proportionnelle**

Efficace pour un indicateur, son utilisation est difficile pour en représenter plusieurs.

Les diagrammes circulaire et à crête de coq[44] font partie des évolutions visant à combler ce manque et ainsi permettre la représentation de plusieurs indicateurs simultanément.

[44] Le premier diagramme à « crête de coq » (aussi appelé « rose » ou « coxcomb ») possédait un nombre de parts fixé à 12, représentant les mois de l'année. Il a été dessiné par Florence Nightingale, pendant la guerre de Crimée (1853-1856) (Source : http://www.datascope.be/sog/SOG-Chapter5.pdf).

Le diagramme circulaire présente les différentes parties d'un tout. La valeur de l'angle central de chaque segment représente la proportion de chacun par rapport à leur somme totale : plus la valeur correspondante à l'indicateur est élevée, plus l'angle sera ouvert. Le tableau IV-3 indique les données de la répartition des salariés français dans les trois secteurs d'activités économiques[45], tandis que la figure IV-22 illustre cette génération.

| France | Agriculture (A) | Industrie (B) | Services (C) |
|---|---|---|---|
| **Nombre de salariés** | 1 113 400 | 7 119 900 | 21 066 700 |
| **Pourcentage** | 3,8% | 24,3% | 71,9% |
| **Angle alloué** | 13,80° | 87,60° | 258,60° |

**Tableau IV-3 : Calculs effectués pour la génération du diagramme circulaire**

En plus de la ventilation des salariés au sein d'une même unité spatiale, le disque constituant le diagramme circulaire indique, par sa surface, la somme de ces actifs, à des fins de comparaison entre les pays (figure IV-22).

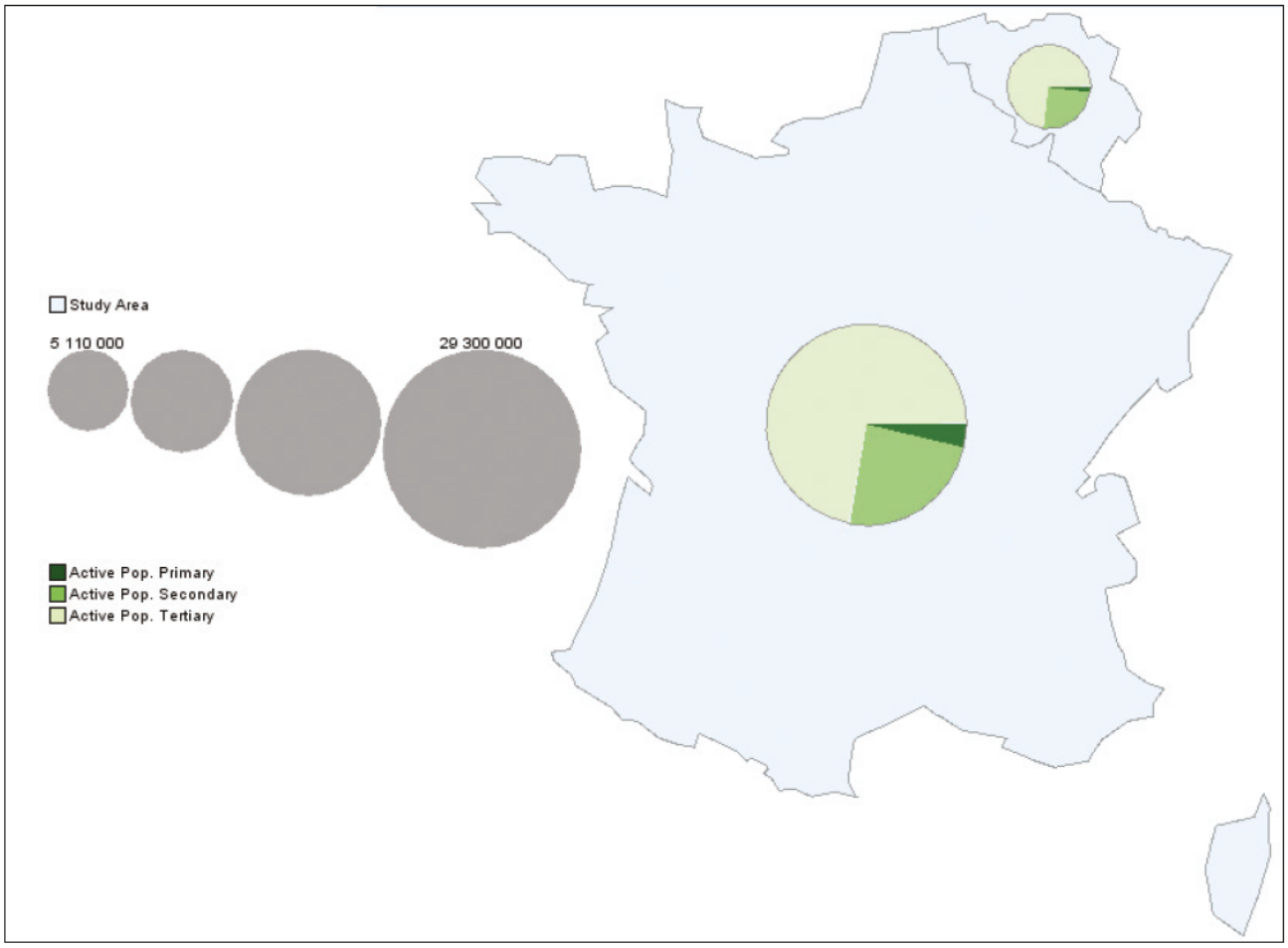



**Figure IV-22 : Représentation par diagrammes circulaires des données de la France et de la Belgique à des fins de comparaisons**

[45] Données extraites du site : https://www.cia.gov/library/publications/the-world-factbook/geos/fr.html

Le diagramme en crête de coq est à l'intersection du diagramme circulaire et de l'histogramme. Il découpe les 360 degrés du cercle en autant de parts égales qu'il y a d'indicateurs à produire. C'est pourtant bien au niveau de la surface que les segments se différencient : il faut donc jouer sur la longueur du rayon. La figure IV-23 reprend l'exemple de la répartition des salariés français et belges selon les secteurs d'activités.

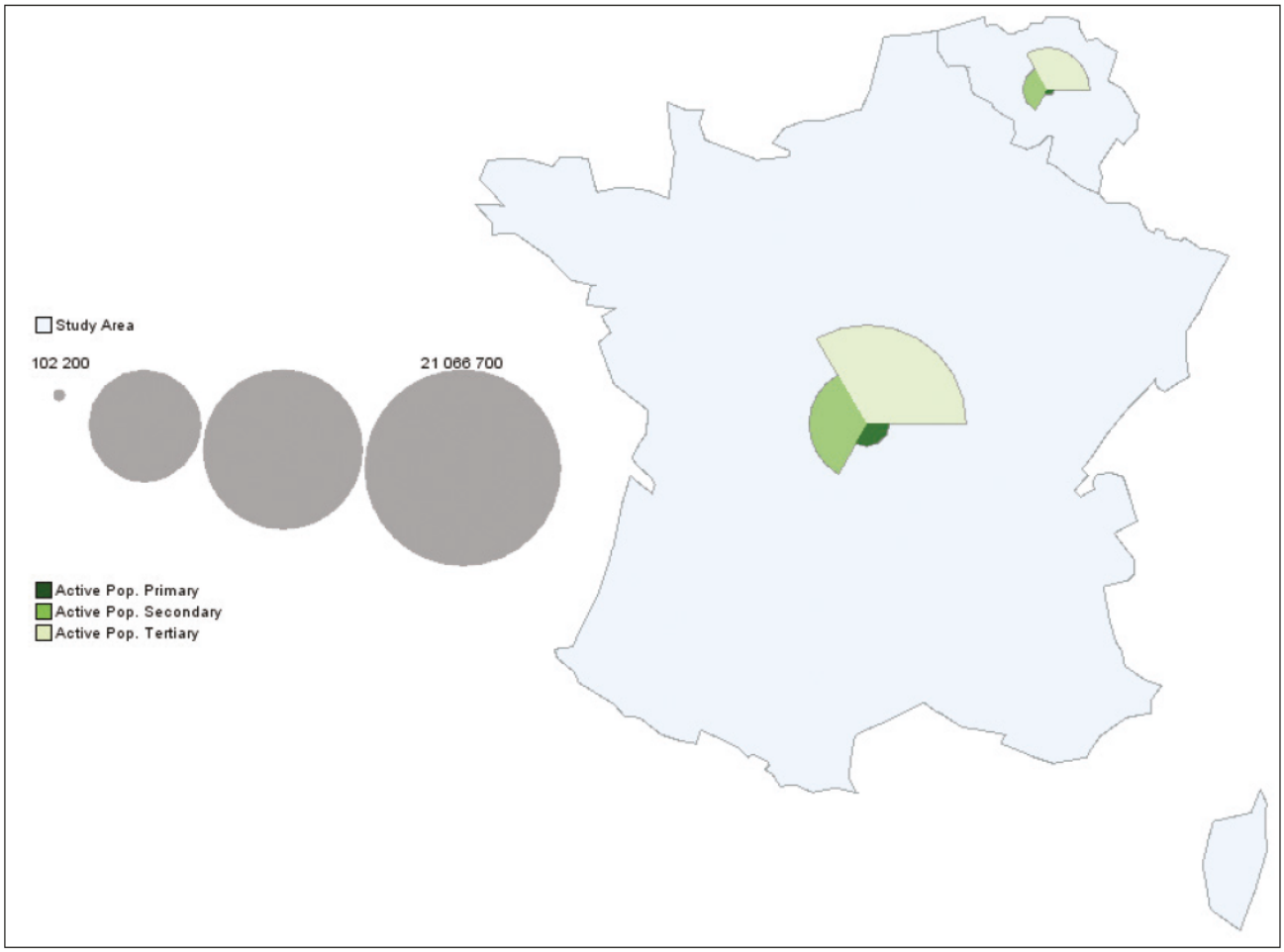



Figure IV-23 : Carte à diagrammes en crête de coq

Les calculs sont pondérés par le coefficient évoqué plus haut pour que la surface allouée aux indicateurs reste dans la proportion décidée.

### Carte choroplèthe

En suivant les principes de la sémiologie, les cartes choroplèthes représentent les ratios entre indicateurs, et ce, par des aplats de couleurs dont la gamme est divisée en un nombre définis de classes. En même temps que les unités spatiales, « *MapBuilder* » reçoit les paramètres spécifiques dédié aux cartes de ratio, notamment un couple de deux indicateurs qui font office de numérateur et de dénominateur, un nombre de classes et un type de calcul de la progression (arithmétique ou géométrique).

Le processus de génération d'une carte choroplèthe est précédé de quelques étapes. Tout d'abord, le calcul du ratio est appliqué à toutes les unités spatiales et le résultat est stocké sur chacune d'elle comme une nouvelle entrée dans le tableau clé/valeur. Puis, les unités sont passées en revue, afin de les classer selon le résultat obtenu : celles qui sont exemptes de résultat ou qui ont un résultat en erreur (division par zéro) sont mises de côté dans des catégories spécifiques. Les données des autres sont collectées afin de définir l'amplitude maximale des informations (minimums et maximums) et

ainsi définir la distribution des unités dans le nombre de classes demandé. Cette représentation est implémentée à partir du code d'HyperAtlas. Elle suit les mêmes principes et contraintes qu'énoncés par nos prédécesseurs, notamment Laurent Poulenard [Poulenard, 2011].

### *f. Commons*

Le module « *Commons* » contient plusieurs sous-modules. Il s'agit de briques transversales fournissant des éléments de communication entre les modules, ainsi qu'un socle de classes et méthodes communes et réutilisables. Cette sous-section propose de brièvement parcourir ces composants afin de parfaire la vision d'ensemble du serveur HyperAtlas³.

#### Generic

Contient les outils nécessaires à tous les autres modules, notamment quelques constantes communes et une gestion globale des exceptions.

#### Map

Le module « *Map* » abrite notamment la classe BBox, directement extraite du paquetage des services WMS. Celle-ci encapsule les délimitations de la carte en y précisant les frontières de champ visible (*Boundary Box*). Grâce à elle et aux méthodes qu'elle propose, l'utilisateur a la main mise sur la gestion du périmètre affiché à l'écran, ainsi que sur les options de zoom.

#### Schéma

« Schéma » possède la méta-structure du cube sous forme d'objets implémentant l'interface *serializable*. Comme l'illustre la figure IV-24, l'architecture correspond à celle décrite dans notre état de l'art et coïncide avec celle livrée par le paquetage `olap4j` dont le module « XmlaClient » se sert pour l'extraction des informations du serveur OLAP.

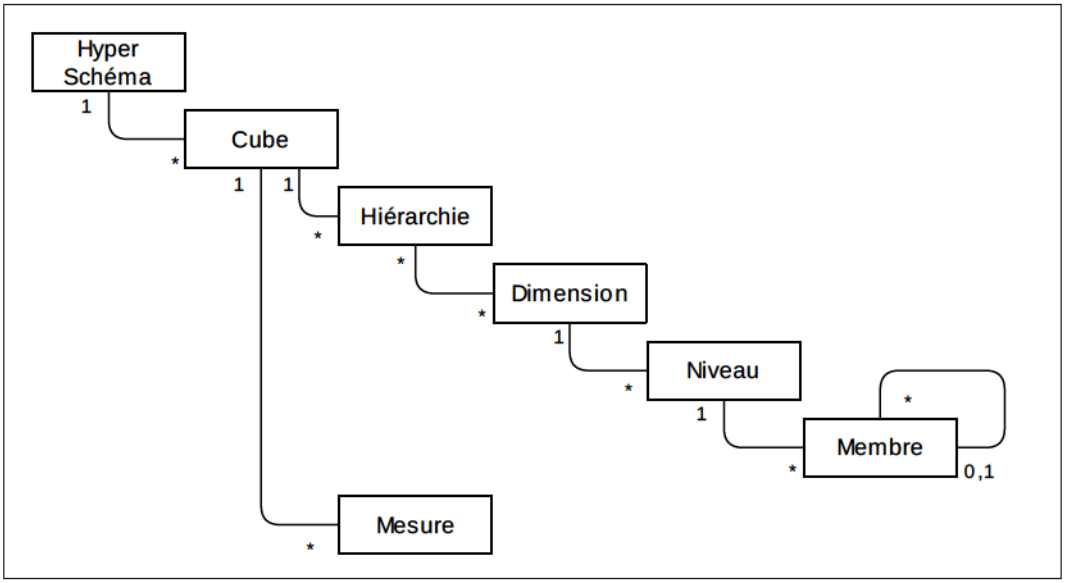



Figure IV-24 : Hiérarchie des classes constituant la méta-structure

#### ResultObjects

Pendant de « *Schema* », « *ResultObjects* » abrite les objets contenant les données extraites du cube et résultat de la requête du client. Parmi eux, on trouve notamment une structure matricielle

constituée de cellules. Ces dernières portent la donnée en elle-même ainsi que diverses métadonnées, comme les en-têtes de lignes et de colonnes, le type d'informations qu'elle contient (attributaire ou géométrique), etc.

### WebServicesBeans

Les échanges entre le serveur et le client s'effectuent via nos services. Lors des transactions, les objets de ce module sont transférés dans un sens ou dans l'autre. La figure IV-25 montre les moments et les finalités des objets utilisés.

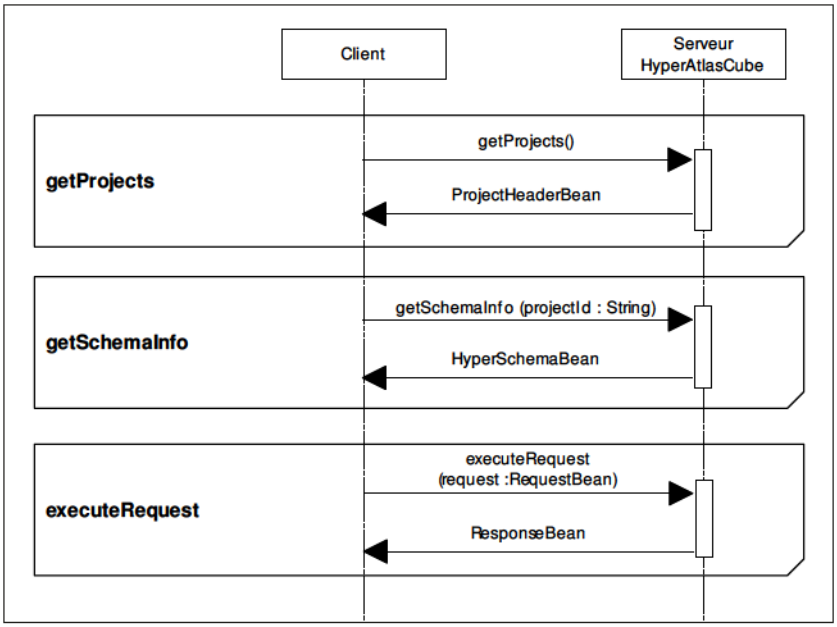



Figure IV-25 : Diagramme de séquence simplifié des services exposés par HyperAtlas³ et des objets échangés

Concernant la méthode `getProjets`, le serveur n'attend rien du client. Lui, à contrario, lui fournit l'objet `ProjectHeaderBean`, contenant essentiellement une liste de projets à deux entrées indiquant les identifiants et noms des sources de données auxquelles il peut fournir un accès.

Le client obtient la méta-structure du projet dont il a donné l'identifiant par le biais de la méthode `getSchemaInfo`. Cette structure est portée par l'objet `HyperSchemaBean`. Celui-ci contient en outre le statut indiquant le caractère complet ou incomplet du schéma.

Pour l'exécution d'une requête, le client fournit un `RequestBean`. Cet objet contient, à minima, l'identifiant du projet de travail et, soit l'objet `MDxQuery`, soit la chaîne de caractères constituant la requête MDX. Dans le premier cas, le niveau spatial de référence peut être renseigné. Son absence signifie seulement que le processus de « spatialisation » ne sera pas déclenché, impliquant un fonctionnement OLAP simple. Dans le second cas, le SLR est purement ignoré. Utilisé de façon plus complète, l'objet `RequestBean` contient les préférences du client en matière de génération cartographique : dimensions de l'image, son type MIME (*Multipurpose Internet Mail Extensions*), son système de référence spatial, le type de carte à tracer, les indicateurs à afficher, les calculs à opérer (dans le cas du ratio), le nombre de classes, etc.

Une fois les opérations effectuées côté serveur, ce dernier renvoi l'objet `ResponseBean` contenant la matrice des valeurs extraites du cube, la requête MDX réellement exécutée (dans le cas de la « spatialisation »), les deux images constituant la carte et sa légende, les dimensions de la légende, son type MIME.

### 3. Documentation

Si ce mémoire est une source documentaire pour la compréhension globale du projet, notamment pour ce qui est de ses objectifs et du contexte dans lequel il a été mené, il est techniquement insuffisant pour l'ingénieur qui doit reprendre ou maintenir l'application. Pendant toute la durée de développement, une attention particulière a été portée à la documentation tant technique que fonctionnelle du serveur et du client.

Pour répondre à cette récurrente problématique, de nombreux outils existent. Nous avons choisi d'utiliser ceux offerts par Maven : ils sont intégrés à notre solution, Maven agissant en tant que gestionnaire de notre projet.

Premier pan de ce point, la JavaDoc. Classique mais néanmoins indispensable, elle offre une documentation technique très détaillée car rédigée granulairement. Le principe de la JavaDoc repose sur des annotations embarquées au code source. Ainsi, classes, attributs et méthodes sont introduits par un laïus expliquant les entrées et sorties de l'élément commenté ainsi que quelques éclaircissements sur sa fonctionnalité. Maven extrait ces indications en même temps que la compilation et les rend sous la forme d'un site normalisé. Le résultat, bien que commun, reste cependant parfois abscons : la compréhension des interactions entre classes nécessite une lecture attentive et de multiples allers-retours entre les pages. Dans le but de faciliter l'immersion du lecteur, et donc la maintenance de l'application, nous avons adjoint à cette documentation une mise en contexte de la classe consultée sous forme d'un diagramme de classes UML. Ces figures sont interactives, un clic sur l'une des classes représentées mène directement à la page lui étant dédiée. Ces représentations sont automatiquement générées lors de la construction de la JavaDoc grâce à un greffon de Maven.

La JavaDoc est intégrée à une collection de documents plus vaste, second sujet de cette section. Cette collection accueille de nombreux autres aspects, notamment techniques, via le recueil des résultats des rapports : couverture des tests, vérification du style d'écriture, métriques, répétition de code, etc. Ceux-ci sont autant de modules adjoints au projet Maven pour aider les développeurs à travailler et attester de la qualité des sources. Mais dans cette grande bibliothèque, il est également possible d'ajouter d'autres types de documents[46]. Ces derniers peuvent contenir n'importe quel type d'informations. Nous nous sommes basés sur cette option pour décrire les aspects fonctionnels de l'application. Les fichiers créés sont autant de pages directement accessibles dans un site englobant toutes les pièces. Générés par le but « *site* » de Maven, ils sont interprétés et mis en forme lors de la création du paquetage complet et rangés de sorte à garder l'architecture que possèdent les modules.

Des exemples de pages de cette bibliothèque sont proposés dans l'annexe C.

## *D. Client HyperAtlas³*

Interface entre l'utilisateur et le serveur, le client HyperAtlas³ apparaît comme la partie émergée de notre projet, aux yeux de l'analyste.

Ce logiciel poursuit principalement deux buts. Il est d'abord notre outil de visualisation dans le sens où l'entend notre énoncé de mémoire. Il donne accès aux informations résultant des demandes de l'usager. Que ce soit sous forme de tableaux, de diagrammes ou de cartes, l'utilisateur dispose de multiples représentations pour mettre en valeur ces requêtes.

[46] Il peut s'agir de pages HTML (*HyperText Markup Language*) ou de fichiers APT (*Almost Plain Text*).

Le client offre aussi les commande pour interagir avec le serveur, en permettant une manipulation aisée des niveaux et des membres via la configuration des axes. Il se charge de la transformation de la sélection de l'utilisateur grâce à la réutilisation du module « *MDXBuilder* ». Par les nombreuses options mises à disposition, l'analyste peut, à loisir, modifier et affiner ses requêtes, les vues demandées, etc.

Ce chapitre est divisé en trois sections. La première expose les technologies utilisées ainsi que les raisons de leur adoption. La seconde s'arrête sur l'interface : elle en décrit les différentes zones, leurs interactions et les fonctionnalités qui y sont présentes. La dernière section est consacrée au guide utilisateur, disponible en ligne.

## 1. Technologie

Cette interface a été pensée pour répondre à deux problématiques. La première concerne la facilité d'accès, que nous voulions maximale. La seconde prône la nécessité de s'assurer que les utilisateurs manipulent la dernière version du logiciel.

Nous ne voulions pas d'un client à installer sur le poste, pour des raisons de praticité : notre outil étant le résultat d'une expérimentation évoluant en fonction des retours des utilisateurs, il doit être facilement diffusable. Tous les usagers potentiels possèdent un navigateur et une connexion au réseau, rendant notre outil, en ligne, accessible à tous. L'avantage principal à ne pas mettre à disposition l'outil côté client est la maîtrise des mises à jour. En effet, le cycle de développement de notre application étant très court, des évolutions sont ajoutées très régulièrement. Le meilleur moyen pour s'assurer que tous les utilisateurs accèdent à la version la plus à jour est de garder tout le logiciel côté serveur, en l'occurrence, sous forme d'application Web. Ce dernier aspect est plus en faveur de la convivialité et de l'ergonomie. Les utilisateurs sont habitués à manipuler des interfaces telles que celles des sites Internet aussi bien dans leur environnement professionnel que personnel.

Pour que l'utilisation de l'application soit la plus naturelle possible, elle est disponible en deux langues, français et anglais. Il est bien entendu possible d'en ajouter d'autre, à condition de constituer un fichier contenant les traductions pour cette nouvelle langue. Celle du navigateur est sélectionnée par défaut

### a. Connexion aux services

Nous avons pu constater lors de la présentation du service, côté serveur, que ceux-ci étaient joignables par deux biais : soit en local par RMI, soit à distance par les services Web.

Les deux principes ont donc aussi été implantés au sein du client HyperAtlas³. Il peut aussi bien être installé sur un serveur distinct et avoir recours à la connexion distante, ou exploiter les ressources de la même machine que le serveur et interagir via la partie locale du service (figure IV-26).

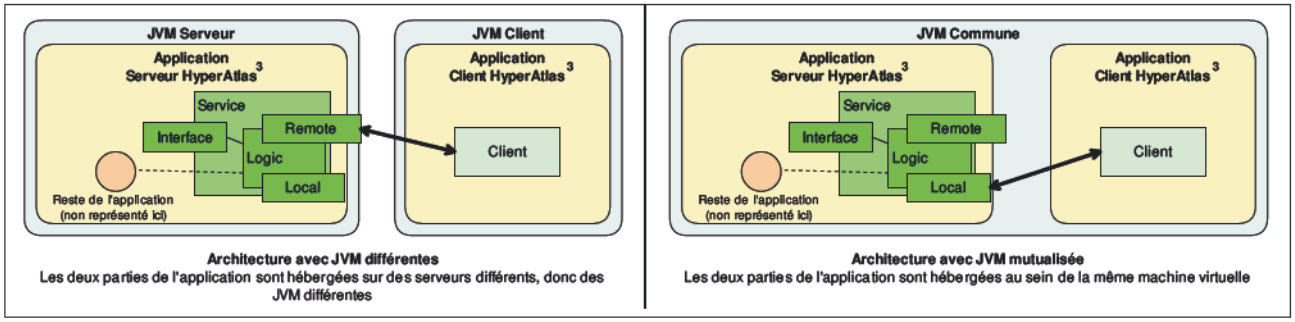



Figure IV-26 : Configurations possibles pour l'interconnexion du client et du serveur

Le connecteur réutilise le sous-module contenant les interfaces du service du serveur, ainsi que ceux contenant les objets partagés (sous-modules de « *commons* »).

La configuration de la méthode d'accès ainsi que son paramétrage sont possibles par le biais d'un fichier dédié à cet usage.

### *b. Génie logiciel*

Lors de l'introduction de cette section, nous avons évoqué la décision de produire notre client sous forme d'une application en ligne. Par ce choix, notre logiciel s'inscrit dans la catégorie des applications Internet riches (*Rich Internet Application* ou RIA). Il s'agit d'un outil en ligne offrant des capacités très proches de celles d'un logiciel installé sur le poste client. Il possède l'avantage d'être utilisable de n'importe quel endroit disposant d'un accès à Internet, à la seule condition de posséder un navigateur récent. Par définition, il ne nécessite aucune installation sur la machine de l'utilisateur.

Dans cette architecture, la grande majorité des opérations sont effectuées côté serveur. Pour celles qui sont déléguées au client, elles sont exécutées grâce aux technologies basées sur JavaScript, c'est le cas de l'Ajax.

Diverses techniques sont disponibles pour développer une telle application et passent, en général, par l'adoption d'un cadriciel (*framework*) pour des raisons de simplification. Pour arrêter un choix parmi les options valables, notre critère essentiel est la rapidité de mise en place afin de nous assurer un cycle de développement le plus court possible. Le second consiste à donner la priorité maximale à l'interface utilisateur : elle doit être accueillante et ergonomique. Enfin, d'autres éléments nous ont servis de référence : simplicité dans les phases de développement, extensible en matière de composants graphiques et standardisé afin d'en faciliter la maintenance.

Parmi les solutions possibles, on retrouve les ténors *Java Server Faces* [47] (JSF 2) et Struts [48], mais aussi Spring MVC [49] ou encore *Google Web Toolkit* [50] (GWT), sans que la liste soit exhaustive.

Apache Struts est la doyenne des alternatives. Apparu il y a 12 ans, ce cadriciel constitue une solution mature et pérenne. Struts est basé sur le patron de conception « modèle, vue, contrôleur » (MVC) et orienté action : il est intimement liée au cycle de requête HTTP. Plébiscité dans le cadre d'une application de grande taille, il est, en conséquence, assez lourd à mettre en place et trouve ses limites en termes d'interfaces.

Les premières spécifications de JSF sont apparues il y a huit ans et la dernière version, nommée « 2.0 », est sortie courant 2009. JSF profite du patron MVC qui a fait ses preuves pour Struts, mais repose sur une structure plus légère. Orienté composant, il permet la manipulation d'éléments graphiques complexes, offrant ainsi un gain de productivité au développement : les fonctionnalités usuelles d'interface Web sont directement disponibles, évitant de les recoder. JSF2.0 est intégrée à la sixième mouture de Java, ce qui en fait une solution standardisée : tout serveur d'applications compatible avec Java6 est alors capable d'accepter une application bâtie sur JSF. Cette version de Java apporte de grandes avancées à JSF notamment par l'utilisation d'annotations. Celles-ci permettent notamment de manier les objets de façon plus aisée, d'en gérer la portée et le périmètre (*scope*). Enfin, il est désormais impossible d'évoquer JSF sans parler des bibliothèques de composants basés sur *Facelets*. Ces derniers proposent un éventail de briques graphiques complémentaires simplifiant d'autant plus la création de la couche de l'interface utilisateur. C'est aussi sur ce côté rendu qu'il

[47] JavaServer Faces : http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html
[48] Apache Struts : http://struts.apache.org/
[49] Spring Source : http://www.springsource.org/
[50] Google Web Toolkit : https://developers.google.com/web-toolkit/?hl=fr-

pêche : la prolifération de balises sur une même page JSF a tendance à rapidement rendre le code illisible.

À l'instar de Struts, Spring MVC est orienté action. Sa gestion du cycle de requête HTTP est très fine, permettant notamment la surcharge des étapes. La première version de Spring MVC date de 2004. Il intègre la gestion de nombreuses représentations, parmi lesquelles l'utilisation de JSP, d'Excel ou de PDF. Son exploitation est basée sur l'utilisation intensive de fichiers de configurations XML, inversant la tendance actuelle qui tente de s'en séparer, le rendant par la même occasion plus difficile à maintenir.

*Google Web Toolkit* est la solution la plus « exotique » de notre panel : tandis que toutes les autres prônent une démarcation prononcée entre la partie consacrée à l'interface utilisateur et celle exécutant le code métier, GWT considère toutes les parties comme un seul tenant. Aucune autre technologie que Java n'est nécessaire, ni possible, à utiliser. Comme tout est Java, tous les outils de mesures et des tests traditionnels fonctionnent parfaitement, ce qui n'est bien souvent pas le cas pour l'analyse de JSP, par exemple. À la compilation, GWT va ensuite transformer le code Java en éléments HTML, CSS (*Cascading Style Sheet*) et JavaScript sans que l'ingénieur ne puisse jamais intervenir dessus. GWT va construire un paquet pour chaque navigateur, optimisant JavaScript pour répondre au mieux à cette cible. Cette optimisation a un coût en temps, payé à chaque compilation, rendant plus laborieux le cycle de développement.

Au regard de nos critères et afin de faire correspondre les capacités offertes des cadriciels étudiés, nous avons opté pour JSF2. Ce choix entériné, l'étape suivante consiste à retenir une librairie de composants. JSF possède un jeu de balises pauvrement doté, c'est pourquoi il est d'usage d'ajouter, nous l'avons vu précédemment, une bibliothèque supplémentaire de composants. L'ajout d'une telle extension permet l'accès plus rapide à une banque de fonctionnalités bien plus vaste et offre des raccourcis non négligeables dans l'utilisation des technologies Ajax. Ces dernières permettent une meilleure interaction entre l'utilisateur et l'application en ne rechargeant que les parties de l'écran qui ont besoin de l'être. Il faut cependant n'en choisir qu'une : les différentes bibliothèques sont généralement incompatibles.

Parmi la galaxie des paquetages existants dans le monde libre, on peut citer Tomahawk[51], Richfaces[52], ICEFaces[53] ou PrimeFaces[54]. La sélection de cette extension est basée sur la correspondance des facilités qu'elle offre et nos besoins. Ces derniers reposent sur six axes :

- Gestion dynamique des tableaux multi colonnes ;
- Gestion du rafraichissement partiel (Ajax) ;
- Représentations graphiques proposées (diagramme) ;
- Outils autour des images ;
- Compatibilité avec les navigateurs les plus répandus ;
- Produit en licence libre et dont la communauté est active (cycles d'évolution et de maintenance courts, pérennité de la solution).

RichFaces, dans sa dernière version (4.2), ne supporte pas la gestion dynamique de tableaux multi colonnes couplée à Ajax, empêchant la mise à jour des données de la vue tabulaire. PrimeFaces est le seul à proposer des fonctionnalités de sélection partielle d'images, via son extension. Il est basé sur jQuery[55], une des bibliothèques JavaScript fournissant un support Ajax parmi les plus connues.

[51] Apache MyFaces Tomahawk : http://myfaces.apache.org/tomahawk/index.html
[52] JBoss RichFaces : http://www.jboss.org/richfaces
[53] ICESoft ICEFaces : http://www.icesoft.org/java
[54] PrimeFaces : http://primefaces.org
[55] jQuery : http://jquery.com

Ce sont autant de raisons qui nous ont fait adopter Primefaces.

L'architecture de la partie logique de notre client est basée sur le patron de conception MVC, puisque JSF le permet. De plus, il a depuis longtemps fait ses preuves : il aide à mieux structurer une application. Il consiste en une séparation en trois parties des données (modèle), des traitements (contrôleur) et de l'interface (vue). Les figures IV-27 et IV-28 illustrent le fonctionnement de MVC.

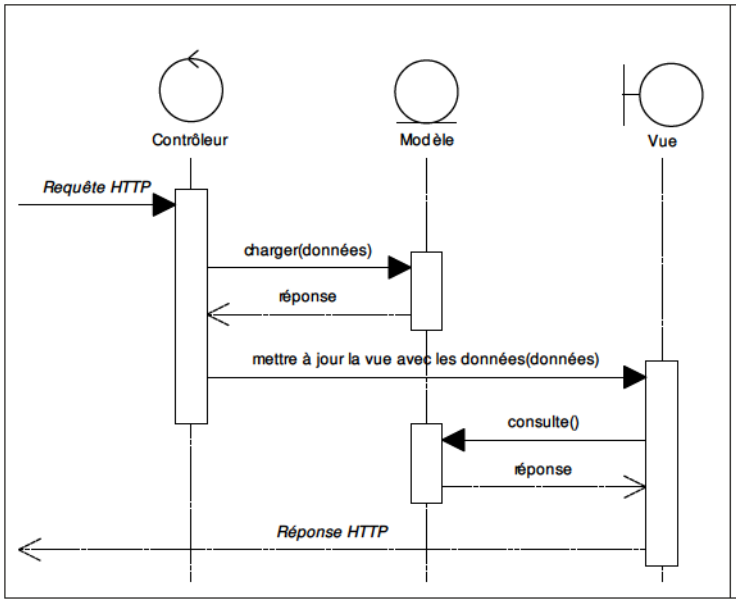



Figure IV-27 : Diagramme de séquence du patron MVC

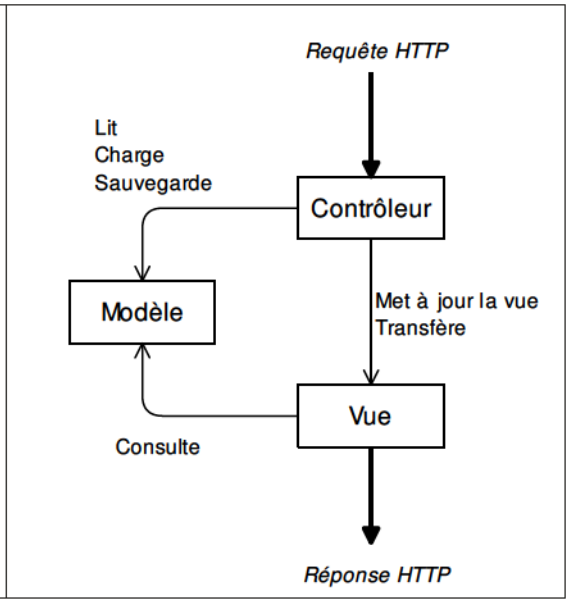



Figure IV- 28 : Architecture du patron MVC

Ce patron apporte clarté et maintenabilité dans le code car les rôles de chacune des couches sont nettement définis et le modèle est bien connu des ingénieurs.

### 2. Interface

À partir de l'étude des logiciels menée dans notre état de l'art (II.B.1.b, page 36), nous avons dégagé les principes mis en jeu dans cette interface. Nous avons déterminé les points communs, les éléments positifs et les axes d'améliorations possibles pour chacun de ces outils. En résulte une liste des conventions en vigueur dans ce genre d'applications, les schémas à reproduire et les sujets nécessitant un travail complémentaire. Ce travail nous a permis d'identifier les blocs nécessaires pour interagir avec l'application ainsi que ceux dédiés au rendu des réponses.

Les maquettes présentées successivement, dont la première est incluse dans la section III.B.3.c (page 58) sous la forme de la figure III-6, ont constitué un support pour affiner nos expérimentations. En exposant nos essais à la critique, les retours des futurs utilisateurs nous ont permis de raccourcir le cycle des essais/retouches pour atteindre plus rapidement l'interface concordant à la fois aux usagers et aux contraintes techniques.

De cette étude des logiciels existants, nous avons convenu que quatre parties constitueraient notre interface :

- Sélection des niveaux et paramétrage des axes ;
- Tableau croisé multidimensionnel ;
- Représentation par diagrammes ;
- Rendu cartographique.

Les emplacements de ces parties ont été répartis sur la surface de notre interface suite aux échanges avec les utilisateurs, comme l'illustre la figure IV-29.

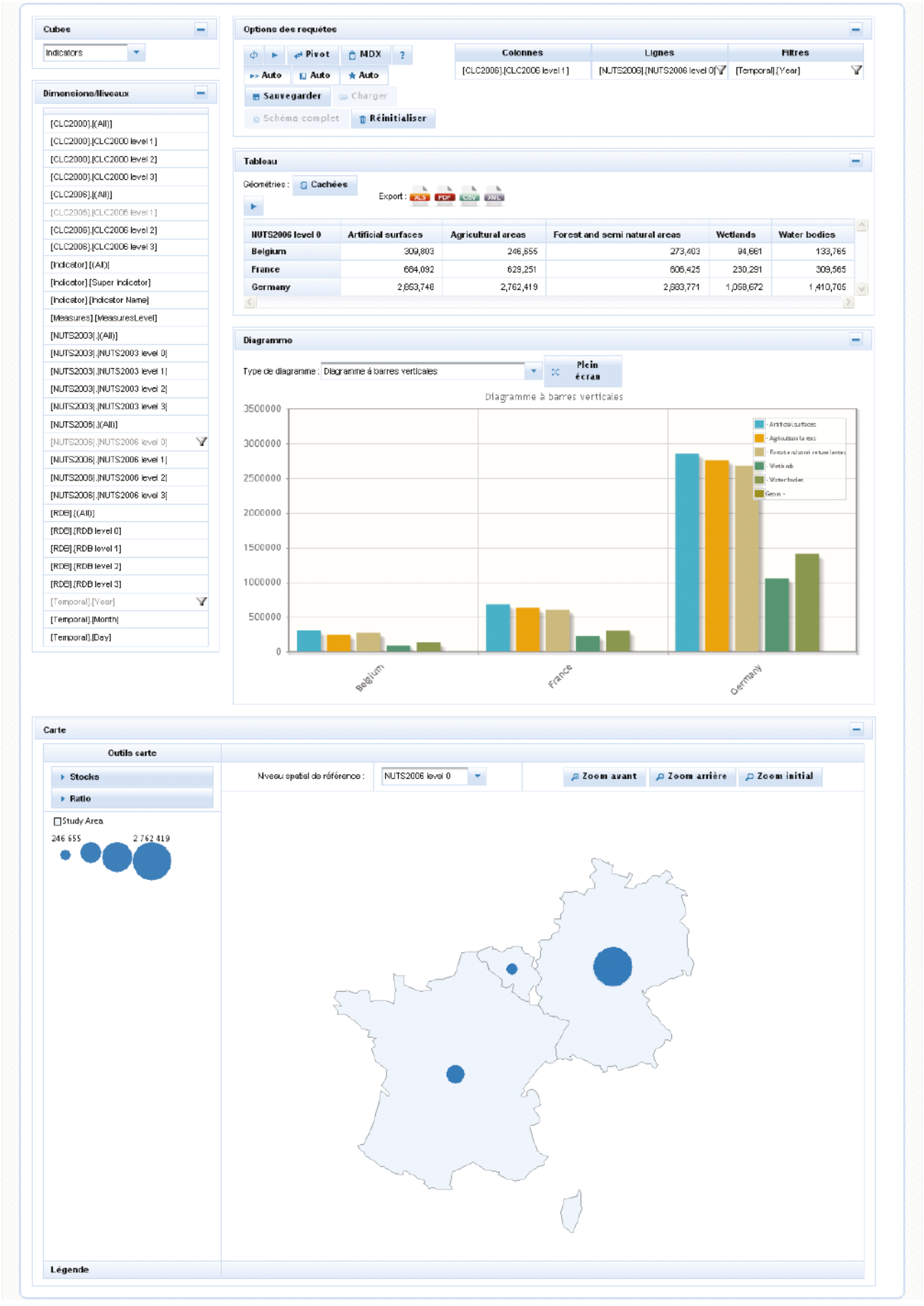



Figure IV-29 : Vue complète de l'interface principale du client

### *a. Sélection des niveaux et paramétrage des axes*

Même si, à première vue, sélection des niveaux et paramétrage dans les axes semblent être deux éléments distincts, il n'en est rien, en réalité : la liste des niveaux constitue la source des informations nécessaires pour les axes.

Le processus de sélection est simplifié au maximum : l'utilisateur choisit un niveau et l'affecte à un axe d'étude. Cette action a pour résultat l'ajout du niveau dans ledit axe.

Pour permettre cette manipulation, un menu contextuel est proposé lors du choix d'un niveau, qu'il soit déjà assigné à un axe (figure IV-30) ou non (figure IV-31).

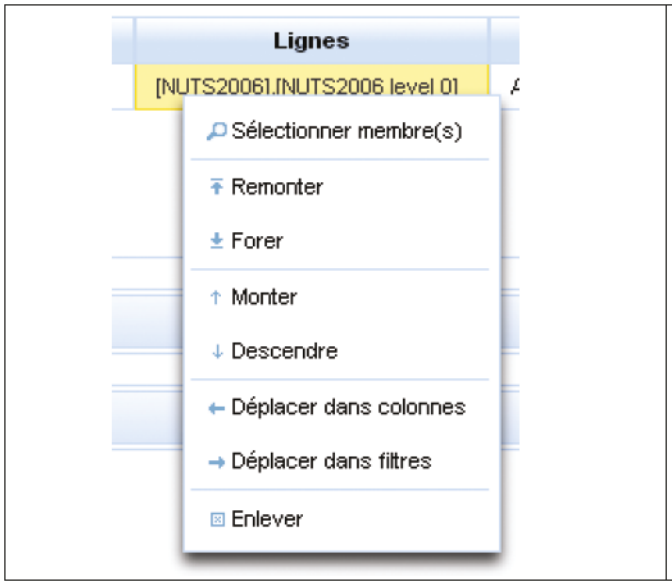



Figure IV-30 : Menu contextuel d'un niveau assigné à l'axe « ligne »

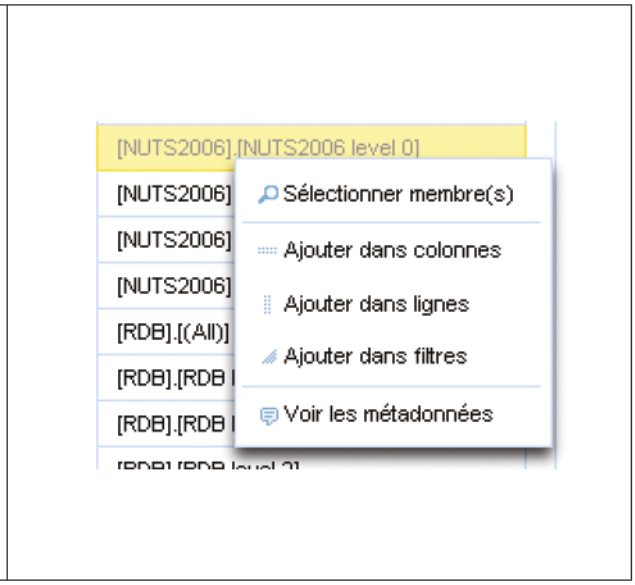



Figure IV-31 : Menu contextuel d'un niveau non assigné

Dans une section précédente (MDXBuilder), les contraintes inhérentes à la construction des requêtes MDX ont été évoquées. Nous allons à présent étudier celles qui ont un impact direct sur la manipulation des éléments nécessaires à leur construction, et ce, au niveau du client.

Parmi elles, il en est une qui consiste à ne pas mélanger les niveaux d'une même dimension, considérée lors de la section « Spatialisation » (page 94) pour exprimer les difficultés rencontrées pour l'ajout des objets chargés de récupérer les formes géométriques. Bien que le serveur dispose de mécanismes vérifiant cette obligation et afin de ne pas le surcharger inutilement, le client est aussi chargé d'opérer cette vérification.

Quand bien même les niveaux d'une même dimension ne sont pas dispersés au travers de plusieurs axes, d'autres contrôles s'imposent. Il faut notamment que les niveaux soient classés dans l'ordre de leur profondeur. Ainsi, concernant la dimension temporelle, le niveau « année » sera toujours placé antérieurement à celui des « mois », même si « année » est ajouté à l'axe après « mois ». Dans le même esprit, les niveaux d'une même dimension ne peuvent être séparés. L'utilisateur peut déplacer un niveau au sein du même axe, grâce aux options « Monter » et « Descendre » visibles sur la figure IV-30. Si sur l'axe contenant le niveau sur lequel il souhaite agir, se trouvent trois niveaux, dont deux de la même dimension, alors il sera le déplacement sera contraint à cette règle. La figure IV-32 illustre ce cas : l'analyste demande le déplacement à une position antérieure du niveau « Année » (`[Temporal].[Year]`, troisième position) alors que deux niveaux CLC se trouvent sur les deux premières positions. Après appui, le niveau « Année » est directement placé en première position.

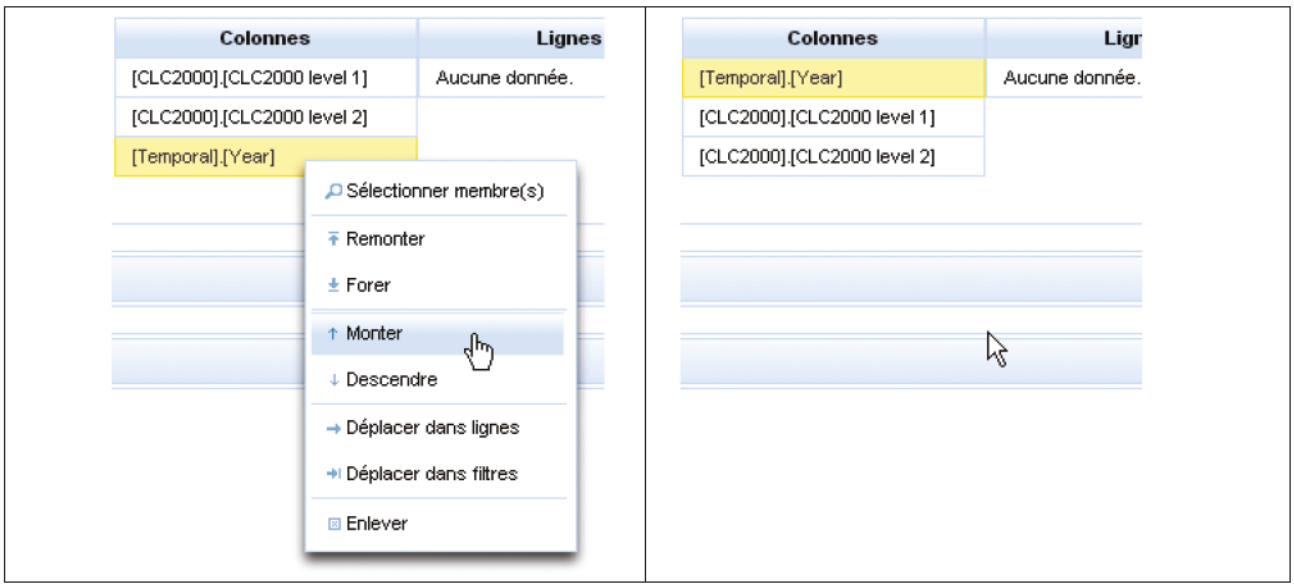



Figure IV-32 : Contrainte sur le déplacement de niveaux

Pour aller plus loin, toujours à partir de cet exemple, si en plus d'« Année », l'axe contient « Mois » et que l'utilisateur souhaite déplacer ce premier niveau, alors « Année » et « Mois » seront déplacés comme un seul élément.

Pour affiner ces recherches, l'utilisateur peut choisir les membres à afficher au sein des niveaux. L'option correspondante, « Sélectionner membre(s) », visible sur les figures présentant le menu contextuel, ouvre une fenêtre listant, d'un côté, les éléments disponibles et de l'autre, ceux qui sont sélectionnés (voir figure IV-36, page 113).

Les options « Remonter » et « Forer » permettent d'opérer les actions, classiques en BI, de remontée et de forage. Elles ajoutent le niveau immédiatement supérieur (respectivement inférieur) à celui sélectionné, permettant par la même, de le généraliser (respectivement décomposer).

Le lien « Voir les métadonnées », disponible sur le menu d'un niveau non affecté, offre l'affichage de données supplémentaires sur ledit niveau, comme l'illustre la figure IV-33.


Figure IV-33 : Métadonnées du niveau NUTS 0

De nombreuses options sont disponibles dans cet emplacement de l'interface, dont les boutons sont référencés par la figure IV-34.

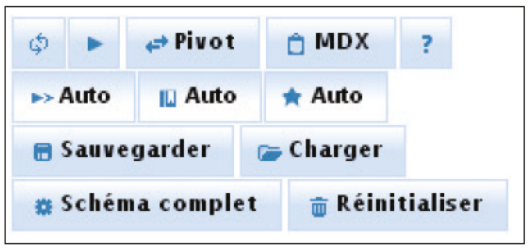



Figure IV-34 : Principales options du client HyperAtlas³

Le tableau IV-4 propose une description succincte de ces options.

| | |
|---|---|
|  | Met à jour tous les composants de l'interface. |
|  | Envoie la requête MDX au serveur et met à jour l'interface. |
|   | Exécute l'opération de pivot en échangeant les niveaux des colonnes avec ceux des lignes et inversement. |
|   | Ouvre une fenêtre permettant la saisie de requêtes MDX. Elle donne accès à la dernière requête créée par le générateur automatique (*MDXBuilder*), ainsi qu'à la dernière réellement exécutée par le serveur, différente de la première dans le cas d'une « spatialisation ». |
|  | Affiche l'aide en ligne. Cette fonctionnalité est le sujet de la section « Guide utilisateur » (page 116). |
|   | Agissent sur le comportement du client. Ces trois boutons à état modifient les actions par défaut de l'interface. Le premier active/désactive l'exécution automatique des requêtes. Enclenché, il provoque un envoi de requête à chaque modification sur l'interface (pivot, ajout ou retrait de niveaux à un axe, modification de membres, etc.). Les deux autres conditionnent la mise à jour respective du diagramme et de la carte. |
|   | Sauvegarde/restaure une configuration. L'utilisateur peut, à tout moment, enregistrer le paramétrage des axes. Un nom lui est affecté lors du processus de stockage. Plus tard, après avoir éventuellement de nouveau manipulé l'interface et modifié les axes, il pourra rappeler une configuration précédemment sauvegardée. Elle sera alors restaurée à l'identique. Les configurations sont propres à la session de l'utilisateur et ne sont ni partagées ni partageables à d'autres. L'utilisateur peut créer autant de configurations qu'il le souhaite. Toutes seront perdues en fin de session. |
|   | Envoie au serveur une requête demandant le méta-schéma. La génération de la méta-structure étant décomposée en deux temps (voir page 93), le client récupérant un schéma partiel ne pourra donner accès à la sélection des membres. Ce bouton indique si le schéma complet est connu du client (bouton désactivé) et, le cas échéant, permet de déclencher une demande de récupération. Tant qu'il ne possède pas la version complète, le client la demandera en même temps que chaque requête `executeRequest`. |
|   | Réinitialise le client à son état d'origine. Cette action vide les axes, retire la sélection des membres, applique l'affichage par défaut du tableau, du diagramme et de la vue cartographique. |

Tableau IV-4 : Description des principales options d'HyperAtlas³

Cette sous-section a passé en revue les options permettant les interactions globales entre l'analyste et le client. Les trois points suivants traitent des moyens de rendre les résultats et présentent les fonctionnalités leur étant spécifiques.

### *b. Tableau croisé multidimensionnel*

Rendu le plus classique d'un système décisionnel, le tableau croisé est le retour le plus proche des données brutes contenues au sein de l'entrepôt, à l'agrégation près.

Le tableau est créé à partir de la matrice fournie par le serveur, en retour de l'appel à la méthode `executeRequest`. Les en-têtes sont détectés via l'indicateur ad hoc que les cellules transportent. Il y a au moins un en-tête par axe. Le nombre maximal n'est limité que par le nombre de niveaux disponibles en source. Si le cube contient *n* niveaux, il est donc possible, dans l'absolu et aux contraintes près, d'affecter à un axe *n*-1 niveaux, le dernier devant être affecté à l'axe opposé.

L'ordre des éléments affichés dans les colonnes et les lignes reflète celui de la requête. Si l'utilisateur a, par exemple, sélectionné les niveaux « année » puis « pays » sur l'axe des lignes, la hiérarchisation sera préservée (figure IV-35).

| Lignes |
|---|
| [Temporal].[Year] |
| [NUTS2003].[NUTS2003 level 0] |

| Year | NUTS2003 level 0 | Artificial surfaces |
|---|---|---|
| 2001 | France | 692,675 |
| 2001 | Germany | 2,991,384 |
| 2001 | Greece | 363,822 |
| 2001 | United Kingdom | 909,945 |
| 2006 | France | 665,481 |
| 2006 | Germany | 3,069,040 |
| 2006 | Greece | 370,598 |
| 2006 | United Kingdom | 801,343 |

| Lignes |
|---|
| [NUTS2003].[NUTS2003 level 0] |
| [Temporal].[Year] |

| NUTS2003 level 0 | Year | Artificial surfaces |
|---|---|---|
| France | 2001 | 692,675 |
| | 2006 | 665,481 |
| Germany | 2001 | 2,991,384 |
| | 2006 | 3,069,040 |
| Greece | 2001 | 363,822 |
| | 2006 | 370,598 |
| United Kingdom | 2001 | 909,945 |
| | 2006 | 891,343 |

Figure IV-35 : Hiérarchisation des niveaux dans les axes.
À gauche, « Année » puis « Pays » ; à droite, « Pays » puis « Année »

Le fonctionnement sera identique dans le cas d'une sélection de membres sur un niveau. En choisissant d'abord le membre « France » puis « Danemark », « France » sera affiché sur la ligne (ou colonne) précédente celle du « Danemark » (figure IV-36).

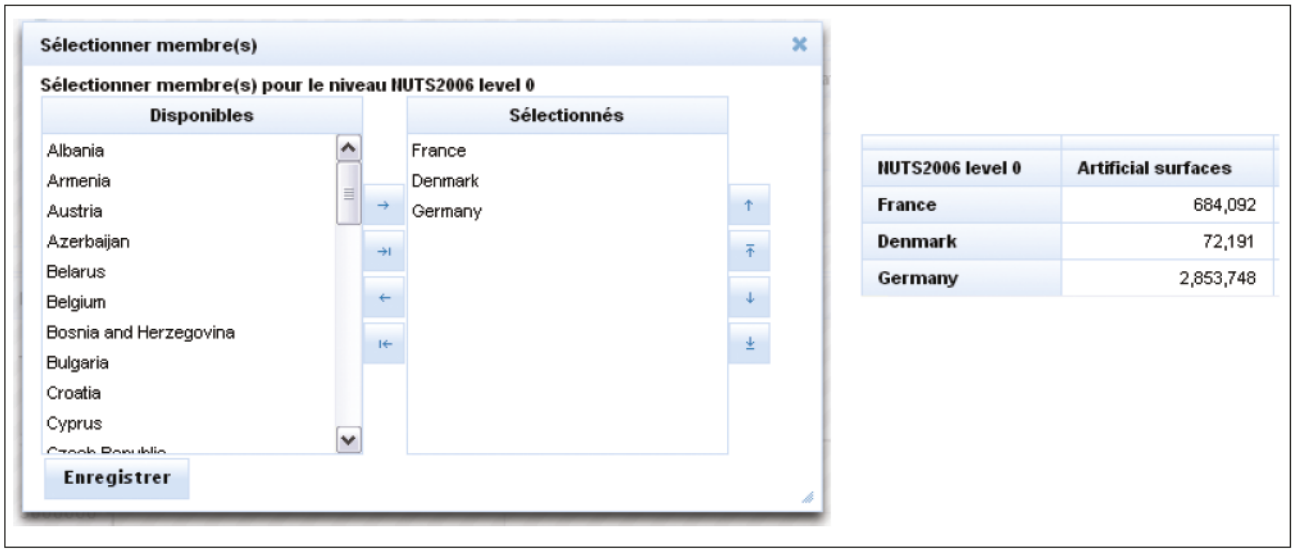



Figure IV-36 : Sélection et classement des membres d'un niveau

La vue tabulaire propose deux options. La première permet d'afficher ou de cacher les formes WKT. Par défaut, cette fonctionnalité est sur la position « cacher » car, pour l'analyste, les formes n'ont pas d'intérêt. Elles peuvent être utiles, en revanche, dans le cas d'un export, seconde option de cette vue. Celle-ci autorise l'extraction sous plusieurs formes :

- tableur XLS ;
- formaté PDF ;
- plat avec séparateur CSV ;
- plat structuré XML.

Les WKT exportés peuvent être réinterprétés par un logiciel tiers, dans le cas, par exemple, d'une extraction XML.

### *c. Représentation par diagrammes*

Les bibliothèques de composants JSF proposent généralement des objets permettant la conception de diagrammes. Leur qualité a été l'un des critères nous ayant fait retenir PrimeFaces, c'est pourquoi leur production lui est entièrement déléguée.

Pour permettre à PrimeFaces la génération de ces diagrammes, il nous faut alimenter des objets dédiés. Ils mettent en jeu les données tabulaires en établissant une correspondance ligne/abscisse et colonne/ordonnée (figure IV-37).

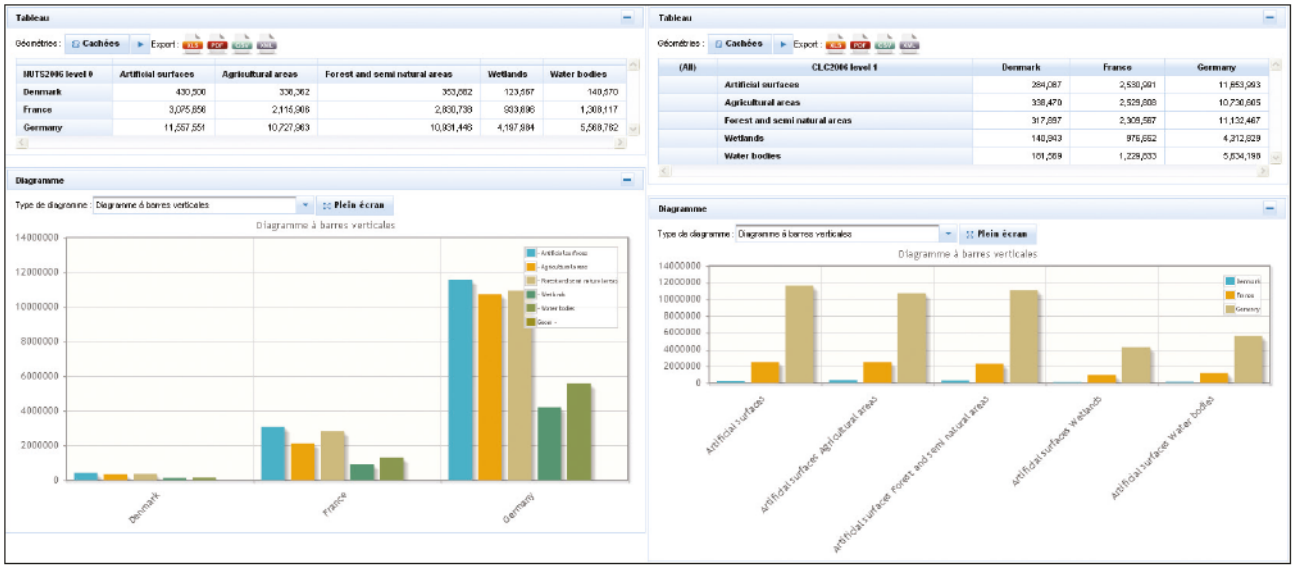



Figure IV-37 : Impact des niveaux en colonne et en ligne sur l'abscisse et l'ordonnée du diagramme

Lorsque plusieurs niveaux sont affectés sur un même axe, leurs membres sont concaténés.

Les représentations disponibles implantées dans notre client sont au nombre de cinq : diagramme à barres verticales, horizontales, verticales empilées, horizontales empilées et linéaire.

Pour une lecture plus aisée, notamment lorsque de nombreuses données sont représentées, une fonction de grossissement est disponible sur les diagrammes, ainsi qu'un affichage en plein écran.

### *d. Rendu cartographique*

La vue cartographique est la réelle plus-value de notre client décisionnel. Bien qu'elle ne fasse qu'afficher la production du serveur et ne génère rien elle-même, les utilisateurs manipulent la dimension spatiale à travers elle.

La zone consacrée à la manipulation cartographique renferme la carte, sa légende et les outils permettant le paramétrage des deux premiers éléments.

Ces outils sont divisés en deux catégories. La première met à disposition les fonctionnalités permettant d'agir globalement sur la carte. On retrouve la sélection du niveau spatial de référence, automatiquement renseignée lors du paramétrage des axes. La gestion du zoom complète cette catégorie en proposant les actions classiques : zoom avant, arrière et réinitialisation. Le zoom peut également être effectué directement sur la carte : une zone de sélection apparaît sur l'image suite

à un clic et le grossissement est déclenché lorsque le bouton est relâché. La figure IV-38 illustre ce panneau de contrôle.


Figure IV-38 : Outils généraux du bloc carte

La seconde catégorie contient les outils dédiés à la configuration plus fine de la carte. Elle donne accès aux fonctionnalités relatives à la génération à proprement parlé : cartes de stocks ou de ratio.

Concernant les options des cartes de stocks, l'analyste peut ici choisir le ou les indicateurs qu'il souhaite voir apparaître. Dans le cas où il n'en sélectionne qu'un, des disques simples sont dessinés. Dans le cas contraire, l'utilisateur a le choix entre deux représentations : diagramme circulaire ou à crête de coq (figure IV-39).

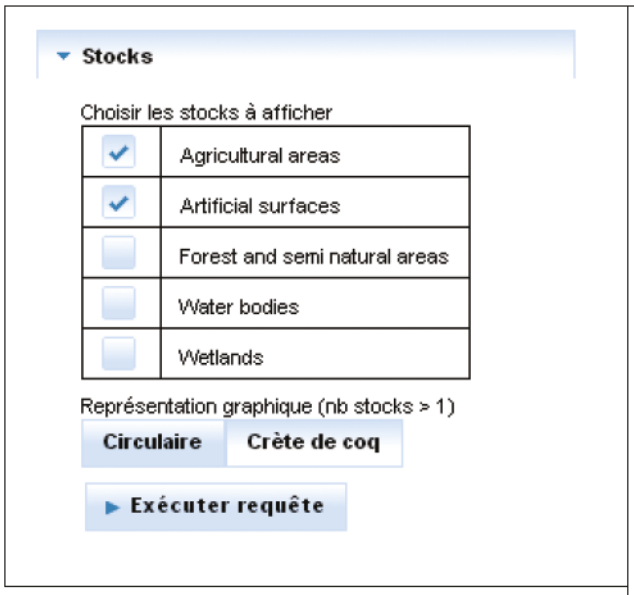



Figure IV-39 : Outils dédiés à la génération de cartes à symboles à taille proportionnelle

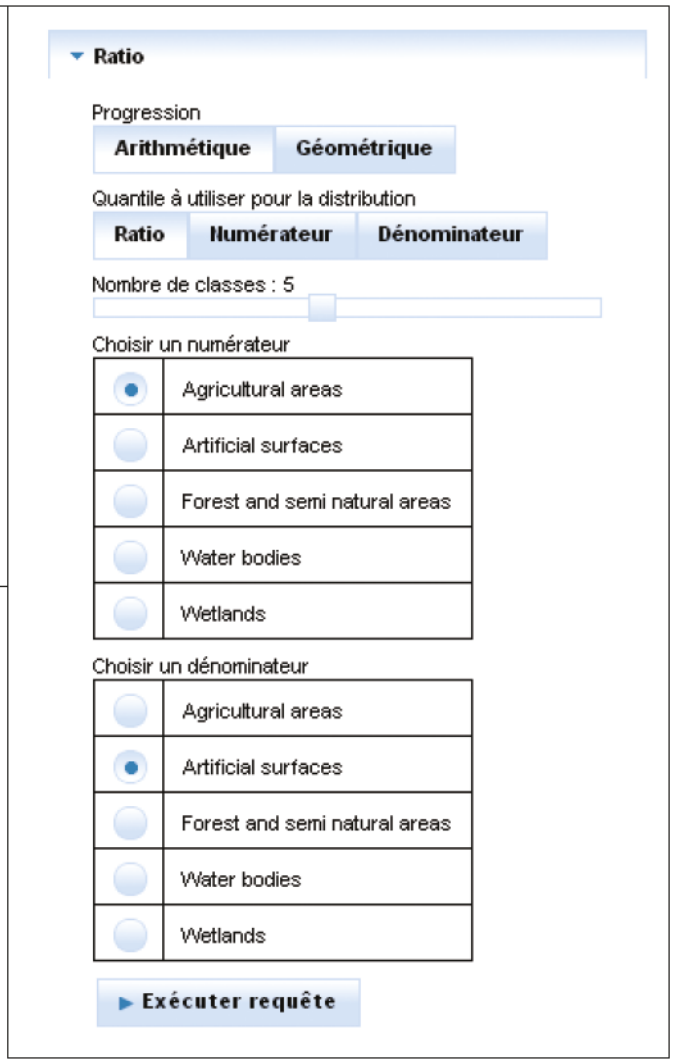



Figure IV-40 : Outils dédiés à la génération de cartes choroplèthes

Les options dédiées aux cartes choroplèthes (figure IV-40) permettent la constitution d'un ratio par le choix d'indicateurs en tant que numérateur et dénominateur. Le type de progression et le quantile à utiliser pour la distribution sont sélectionnables par le biais de boutons. Une glissière permet la modification du nombre de classes pour la ventilation des ratios.

### 3. Guide utilisateur

Une application, aussi ergonomique soit elle, doit disposer d'une notice explicative pour être complète. En effet, si la plupart des fonctionnalités par leur nom, par l'icône qu'elle arbore, ou par l'aide proposée en info-bulle, se suffisent à elles-mêmes, le périmètre ou le but de quelques autres peuvent être flous.

Il est d'usage, dans le cadre d'un logiciel en ligne, de présenter cette documentation intégrée à l'outil. Sur l'interface principale, un clic de souris sur le bouton représentant le classique point d'interrogation mène à ce guide destiné à l'utilisateur (figure IV-41).


Figure IV-41 : Affichage du guide utilisateur dans un navigateur

Contrairement à la documentation technique, extraite dynamiquement lors de l'empaquetage de l'application et destinée aux ingénieurs en charge de la maintenance de l'outil, cette notice est tournée vers les analystes et est donc purement fonctionnelle. Chaque fonctionnalité est détaillée et son champ d'action délimité. À l'instar de l'application, le guide utilisateur est proposé en deux langues, l'anglais et le français.

# Partie V - Conclusion

**Cette dernière partie conclut ce mémoire par cinq sections. Elles rappellent les objectifs, résument les réalisations, synthétisent les résultats obtenus, ouvrent quelques perspectives d'évolutions et dressent finalement un bilan plus personnel de ce stage.**

## *A. Rappel des objectifs*

Avec l'émergence du *Spatial OLAP*, de nouvelles solutions voient le jour au croisement des mondes des systèmes d'informations géographiques et de l'informatique décisionnelle.

L'objectif principal de ce stage et de ce mémoire ingénieur était la mise en place d'un environnement géodécisionnel dans le but d'expérimentations et tests des capacités de ces nouveaux outils.

## *B. Synthèse des réalisations*

Afin de mener à bien cette mission, nous avons d'abord dressé un état de l'art des deux composantes de nos objectifs. Nous avons débuté par l'informatique décisionnelle, parcourant son architecture et les principes qui la sous-tendent. Nous nous sommes arrêtés plus longuement au cœur de la structure, notamment sur les méthodes de construction de l'entrepôt et sur les technologies relatives au SOLAP, avant de repartir sur les différentes manières de représenter les données, spécialement dans les systèmes décisionnels. Nous avons continué nos recherches en étudiant une partie de la géomatique via les particularités des informations géoréférencées et les différents moyens de les représenter. Enfin, pour compléter nos connaissances théoriques et déterminer les conventions du domaine, nous avons étudié quelques-uns des outils permettant la restitution des informations, qu'elles que soient leurs formes.

La partie « démarche et proposition » a décrit notre méthodologie, abordé nos décisions préliminaires, fondations de notre projet, et détaillé notre feuille de route.

Les chapitres de la partie « réalisation » ont d'abord présenté notre cadre de travail puis ont développé les différents composants constituant notre projet. Ainsi le premier traite des outils utilisés pour gérer notre projet et nos livrables, tant au niveau du code que des documentations ou des indicateurs de qualité. Le second touche à la mise en place de l'entrepôt de données et du serveur SOLAP. Le troisième s'arrête plus longuement sur l'architecture du serveur HyperAtlas[3], de ses différents modules et des problématiques qu'ils résolvent. Enfin, le quatrième évoque notre client, directement connecté au serveur, ainsi que son guide d'utilisation. Nonobstant le fait d'être construite en tant qu'application Internet, elle donne accès à une interface riche en fonctionnalités et permet une exploitation complète des rendus sous plusieurs formes.

Du point de vue quantitatif, la partie « client » occupe 21 classes, tandis que la partie « serveur » est réalisée par 147 classes. Le tableau V-1 indique la ventilation de ces classes entre les différents modules.

| Partie | Module | Nombre de classes |
|---|---|---|
| Client | HyperAtlasCube | 21 |
| Serveur | BusinessUnit | 5 |
| | Commons | 34 |
| | MapBuilder | 65 |
| | MdxBuilder | 23 |
| | Service | 4 |
| | XmlaClient | 17 |
| **Total** | | **168** |

Tableau V-1 : Nombre de classes Java par modules

En plus de ces 168 classes s'ajoutent 50 tests unitaires JUnit. Ils couvrent majoritairement deux modules, comme l'atteste la figure V-1, extraite de la génération du tableau de bord global de Maven : *MDXBuilder*, développé à l'aide de la méthode TDD et l'application Web constituant le client HyperAtlas[3].

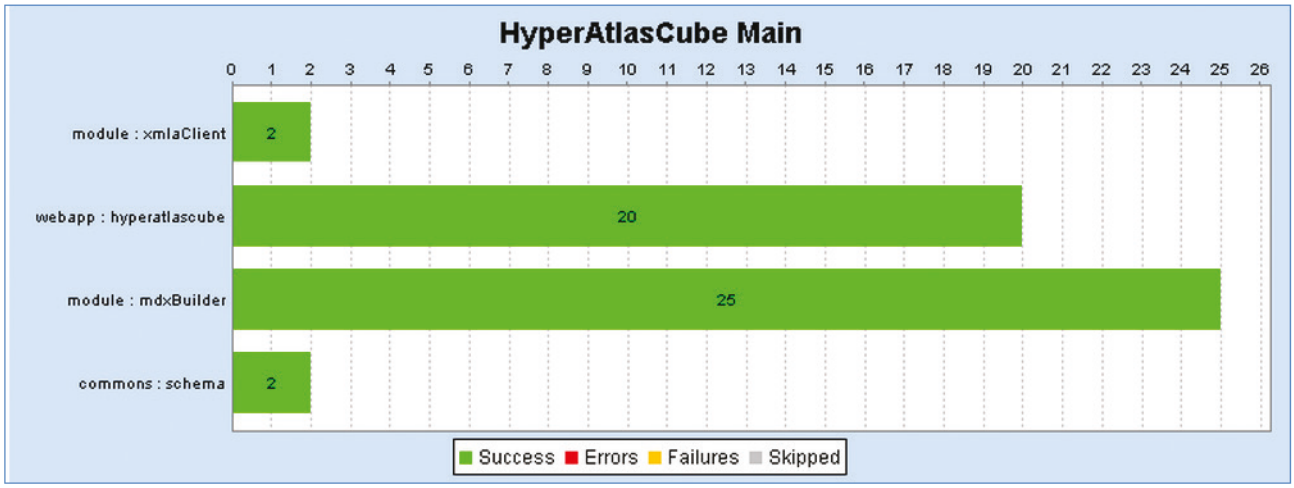



Figure V-1 : Répartition des tests par modules

L'architecture retenue pour notre outil est particulière. Le découpage en deux éléments de la partie située entre le serveur SOLAP et l'utilisateur répond, nous l'avons vu, aux problématiques de partage soulevées par la directive INSPIRE. Par cette structure et les développements que nous avons menés, HyperAtlas[3] est conforme aux attentes de la directive du point de vue des services de consultations (données du tableau accessibles, carte à fond transparent permettant la superposition, manipulation des échelles et des légendes) et de téléchargement (les données sont accessibles par les services). Un travail complémentaire reste cependant à fournir au niveau des services de recherche et de transformation.

## *C. Synthèse globale*

Notre travail est la première pierre d'un édifice géodécisionnel, une expérimentation menée dans l'objectif de tester les capacités des outils SOLAP. Il s'agit d'un prototype dégrossissant les grandes lignes constituant les domaines à la jonction desquels il se positionne. En effet, ces domaines sont très vastes et il n'était pas possible, dans le temps imparti, de couvrir entièrement tous les aspects auxquels un outil complètement abouti devrait répondre.

Le champ de l'informatique décisionnelle à lui seul est étendu, s'étendant sur de nombreux sujets. Le chemin parcouru par l'information est long avant qu'elle n'atteigne les écrans de l'analyste : de la récupération de la donnée brute en provenance de systèmes opérationnels, à sa restitution auprès de l'utilisateur final, en passant par sa transformation par les outils ETL, puis par son stockage au sein d'entrepôt, à son extraction par l'OLAP et enfin à sa manipulation par les outils de rendu.

La représentation des données est, lui aussi, un sujet considérable. Les affichages les plus traditionnels font cependant consensus et sont donc très souvent utilisés : il s'agit notamment des tableaux ou les diagrammes. De plus en plus d'outils se sont enrichis par l'adjonction d'une dimension spatiale et offrent une vue cartographique conventionnelle, telle que celle que nous avons mise en place. Ceci est le résultat de la généralisation de la géomatique. Cependant, d'autres représentations cartographiques plus « artistiques » existent, comme les cartes déformées selon le poids que chaque unité spatiale représente par rapport au total, ou les schématisations, qui sont plus difficilement automatisables et qui nécessitent une implémentation spécifique, pour un résultat aléatoire (risques de superposition).

Afin d'obtenir un outil fonctionnel et en accord avec l'objectif de notre stage, le périmètre de nos investigations a été délimité sur la partie extraction depuis l'hypercube jusqu'au rendu à l'utilisateur. Concernant la représentation, nous nous sommes également limités à deux types de cartes et quelques diagrammes.

Grâce à cette expérimentation, nous pouvons aussi déterminer ce que peuvent apporter ces technologies au projet HyperAtlas.

Logiquement d'abord, de par la nature du décisionnel, les opérateurs de forage, de remontée et de pivot apportent une plus-value non négligeable dans l'exploitation des données ainsi qu'une souplesse dans la navigation. La gestion naturelle des dimensions au sein du SOLAP lui confère la possibilité de stocker plus d'une perspective spatiale à des fins de comparaison ou de manipulation via les fonctionnalités spatiales telles que l'intersection, etc.

Physiquement ensuite et parce qu'il s'appuie sur XMLA, moyen d'accès aux données standardisé, notre serveur peut être connecté à n'importe quelle source (S)OLAP, apportant une certaine interopérabilité qui fait pour l'instant défaut à HyperAtlas.

Enfin, nous avons organisé les données pour faire en sorte d'obtenir un comportement proche de celui d'HyperAtlas. Bien qu'elle nous offre des possibilités supplémentaires, nous n'exploitons pas encore pleinement les aptitudes de l'informatique décisionnelle. En effet, cette technologie est pensée pour manipuler de grandes quantités de données. Avec l'avènement des *smartphones*, possédant généralement un GPS, les messages et les photographies sont géolocalisés. Ces appareils pourraient agir en tant que système multi-agents et les informations qu'ils produisent pourraient être traitées comme sources de données.

## *D. Perspectives*

De par les contraintes relatives au délai, notre travail est centré sur la partie aval de la chaîne décisionnelle, de l'entrepôt à la restitution. Un environnement de *Business Intelligence* complet englobe aussi les traitements en amont, notamment l'enrichissement de l'entrepôt à partir de sources externes via les outils ETL. Les travaux de l'Université Autonome de Barcelone, qui nous ont en partie servi à construire notre entrepôt, s'inscrivent dans cette optique. Les opérations de transformation nécessaires à la répartition des indicateurs sur les différents découpages spatiaux restent cependant expérimentales et doivent donc être encore affinées.

La gestion des sources de données du serveur HyperAtlas[3] s'effectue par le biais d'un fichier de propriétés. Pour faciliter les opérations courantes et rendre plus aisée cette administration, une interface idoine serait plus pratique à manipuler. Via cette console, il serait alors possible de restreindre l'accès à certaines configurations en les soumettant à une authentification. La création de comptes utilisateurs permettrait d'étendre la fonctionnalité de sauvegarde/restauration de la disposition des niveaux dans les axes. En effet, le système pourrait enregistrer la combinaison et la rattacher au compte connecté en vue d'une réutilisation ultérieure, lors d'une autre analyse afin d'en comparer les résultats. Une option de partage entre utilisateurs serait alors aussi envisageable. Il serait même concevable de constituer une banque de requêtes utilisables par n'importe quel analyste.

Réutilisant le moteur de génération cartographique d'HyperAtlas, notre projet profite de ses capacités mais souffre aussi des mêmes défauts. Les limites à l'affichage des géométries sont rapidement atteintes. Il s'agit notamment du nombre de points alloués au tracé des unités spatiales. Trop faible, la forme sera schématisée voire méconnaissable ; trop élevé, les performances s'en trouveront dégradée, empêchant une navigation fluide entre les vues. Perfectionner ce module en profitant de l'expérience acquise dans le but d'atteindre une version plus permissive et plus complète semble être la meilleure solution. De nouvelles représentations pourraient alors être ajoutées, comme par exemple la multicarte.

Quelques axes supplémentaires d'amélioration peuvent aussi être envisagés.

D'abord, la synchronisation de toutes les parties de l'interface apporterait une homogénéité dans le maniement de l'application. Ce point peut être décomposé en deux parties. La première concerne la concordance entre la saisie de la requête MDX et l'interface. Pour le moment, toute modification sur la configuration des axes via l'interface est répercutée sur la requête, mais la réciproque n'est pas vraie. Un interpréteur de requête pourrait permettre un tel résultat. La seconde partie focaliserait sur la sélection simultanée des unités sur toutes les vues. Ainsi, un appui sur la cellule de la vue tabulaire mettrait en évidence cette même unité sur le diagramme et sur la carte.

Ensuite, le recours à l'opérateur de perçage ou aux fonctionnalités spatiales n'est possible que par le biais de la saisie manuelle de la requête. Ceux de remontée et de forage ne sont accessibles que par le menu contextuel. Ajouter ces commandes directement sur l'interface, que ce soit sur le tableau, le diagramme ou la carte, apporterait une expérience de navigation accrue.

Enfin, un dernier axe d'amélioration, serait l'ajout d'indicateurs de qualité des données. En partant du principe que les requêtes sont exécutées sans aucun contrôle de cohérence logique, en affectant aléatoirement un niveau sur chacun des axes, l'outil affichera un résultat. Celui-ci n'aura qu'une infime chance d'être cohérent. Prenons un exemple trivial. L'utilisateur attribue le niveau « pays » des NUTS en colonne et le premier niveau de la classification biophysique CLC en ligne. L'interface affichera un tableau référençant les axes demandés. Or, le cube, tel que nous l'avons construit, contient deux relevés démographiques (2001 et 2006) et l'agrégation par défaut est la somme. Les valeurs obtenues sont donc le résultat de la somme des populations des deux relevés, répartie

par pays et par CLC. Kamal Boulil [Boulil, 2012] propose la mise en place de règles pré et post traitements afin de déterminer le niveau de cohérence des résultats. Celui-ci est ajouté en tant que métadonnées, information ensuite lue par l'interface, qui peut indiquer par un code de couleur la qualité de la donnée. Ce système vérifie les incohérences logiques affectant les qualités des données entreposées, des agrégations et de l'exploration des données.

## *E. Bilan personnel*

Ce mémoire clôt ce stage en immersion dans le monde de la recherche. Au sein de l'équipe STEAMER, j'ai pu me familiariser avec les concepts de la géomatique, concepts qui m'étaient totalement inconnus avant de débuter l'épreuve TEST (Travail d'Etude et de Synthèse Technique) de l'unité d'enseignement ENG111. La production cartographique, en accord avec les règles sémiologiques, représentait alors pour moi un défi. Les membres de l'équipe ont su partager leur passion afin de me faire profiter d'une part de leur expérience et ainsi me permettre de forger la mienne sur ces thématiques.

L'autre aspect de notre sujet, l'informatique décisionnelle, représente à mes yeux l'un des domaines les plus porteurs que puisse proposer une société de services informatiques, ce qui, j'en suis persuadé, ajoute un plus non négligeable à mon parcours. Découvert lors de l'unité NFE115, j'ai pu, grâce à ce mémoire, perfectionner ma compréhension dans ce champ d'investigation.

Côtoyer au quotidien les enseignants chercheurs m'a donné l'envie de partager à mon tour cette connaissance. Ce souhait a été réalisé : j'ai pu découvrir l'autre facette du monde de l'enseignement, non plus en tant qu'étudiant, mais en tant qu'enseignant. Cette expérience a nécessité d'approfondir encore les notions de *Business Intelligence* afin de les maitriser entièrement et pouvoir les synthétiser avant de les retransmettre. Cette aventure, bien que relativement courte (une douzaine d'heures) a été intense humainement, si bien que j'aspire à la renouveler dès que l'occasion se présentera à nouveau.

Réaliser ce stage dans un environnement de recherche a été pour moi une expérience enrichissante, d'autant que j'étais attiré par le côté « recherche et développement » que l'on peut retrouver dans les entreprises privées de grande importance. Elle m'a fait réaliser la complexité de mettre en place un prototype viable dans un délai assez court, où les technologies ne sont pas encore toutes arrivées à maturité.

Partager avec des chercheurs, comparer les réalisations de chacun, prendre le temps de capitaliser, tout ceci m'a permis de prendre le recul nécessaire à l'évaluation de mes réalisations, ainsi qu'à la remise en cause de certaines de mes décisions afin de voir les problématiques sous un jour nouveau.

Ce mémoire clôt aussi ma formation CNAM débutée en 2007. L'apprentissage par les cours du soir et à distance est un travail nécessitant organisation, patience et ténacité. Les compétences techniques acquises tout le long de la formation m'ont permis de proposer des solutions pratiques au sein de mon entreprise, et, réciproquement, les travaux que j'y réalisais me permettaient de prendre du recul sur ces connaissances théoriques. La diversité des enseignements qu'il m'a été donné de suivre a permis à mon esprit d'élargir le champ des possibles. L'informatique est, plus que jamais, un domaine transversal où toutes les spécialités ont des besoins : ce mémoire prouve que c'est le cas de la géographie.

Enfin, ce mémoire ouvre une nouvelle page à ma vie professionnelle. Le titre d'ingénieur n'est pour moi qu'une étape. Ma soif d'apprendre n'est pas étanchée et je désire continuer ma formation au travers d'une maitrise en administration des affaires (MBA ou *Master of Business Administration*), afin d'atteindre de plus hautes responsabilités dans la gestion d'entreprises.

# Annexes

**Annexe A :** Exemple d'échange de messages XMLA de découverte ..... 124

**Annexe B :** Exemple d'échange de messages XMLA d'exécution ..... 126

**Annexe C :** Extraits de la bibliothèque documentaire ..... 129

**Annexe D :** Description de l'entrepôt ..... 132

# *Annexe A : Exemple d'échange de messages XMLA de découverte*

## Requête de découverte envoyée par le consommateur

```
<Discover xmlns="urn:schemas-microsoft-com:xml-analysis">
  <RequestType>DISCOVER_DATASOURCES</RequestType>
  <Restrictions>
    <RestrictionList/>
  </Restrictions>
  <Properties>
    <PropertyList>
      <Format>Tabular</Format>
    </PropertyList>
  </Properties>
</Discover>
```

## Réponse du fournisseur de données

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"[...]>
  <SOAP-ENV:Body>
    <cxmla:DiscoverResponse xmlns:cxmla="urn:schemas-microsoft-com:xml-analysis">
      <cxmla:return>
        <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset" [...] >
          <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" [...]
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»/>
[... Définition des autres valeurs utilisées par la réponse ...]
            </xsd:sequence>
```

```
            </xsd:complexType>
          </xsd:schema>
          <row>
            <DataSourceName>
Provider=Mondrian;DataSource=mixte042012_withindicator;
            </DataSourceName>
            <DataSourceDescription>
GeoMondrian mixte042012_withindicator Data Warehouse
            </DataSourceDescription>
            <URL>http://localhost:9090/geomondrian_wi/xmla</URL>
          </row>
        </root>
      </cxmla:return>
    </cxmla:DiscoverResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# *Annexe B : Exemple d'échange de messages XMLA d'exécution*

## Requête envoyée par le consommateur

```
<Execute xmlns=»urn:schemas-microsoft-com:xml-analysis»>
  <Command>
    <Statement>
SELECT ([NUTS2006].[NUTS2006 level 0].MEMBERS) ON COLUMNS,
([Temporal].[Year].MEMBERS) ON ROWS FROM [Indicators]
    </Statement>
  </Command>
  <Properties>
    <PropertyList>
      <Catalog>mixte042012_withindicator</Catalog>
       <DataSourceInfo>Provider=Mondrian;DataSource=mixte042012_withindicator;
</DataSourceInfo>
      <Format>Multidimensional</Format>
      <AxisFormat>TupleFormat</AxisFormat>
    </PropertyList>
  </Properties>
</Execute>
```

## Réponse du fournisseur de données

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"[...]>
  <SOAP-ENV:Header></SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <cxmla:ExecuteResponse xmlns:cxmla="urn:schemas-microsoft-com:xml-analysis">
      <cxmla:return>
        <root xmlns="urn:schemas-microsoft-com:xml-analysis:mddataset"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" [...] >
          <xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema [...] >

            <xsd:complexType name="MemberType">
              <xsd:sequence>
                <xsd:element name="UName" type="xsd:string" />
                <xsd:element name="Caption" type="xsd:string" />
                <xsd:element name="LName" type="xsd:string" />
                <xsd:element name="LNum" type="xsd:unsignedInt" />
                <xsd:element name="DisplayInfo" type="xsd:unsignedInt" />
                <xsd:sequence maxOccurs="unbounded" minOccurs="0">
                  <xsd:any processContents="lax" maxOccurs="unbounded" />
                </xsd:sequence>
              </xsd:sequence>
              <xsd:attribute name="Hierarchy" type="xsd:string" />
            </xsd:complexType>
[... Définition des autres types complexes utilisés par la réponse ...]
          </xsd:schema>
          <OlapInfo>
            <CubeInfo>
              <Cube>
```

```
            <CubeName>Indicators</CubeName>
          </Cube>
        </CubeInfo>
        <AxesInfo>
          <AxisInfo name="Axis0">
            <HierarchyInfo name="NUTS2006">
              <UName name="[NUTS2006].[MEMBER_UNIQUE_NAME]" />
              <Caption name="[NUTS2006].[MEMBER_CAPTION]" />
              <LName name="[NUTS2006].[LEVEL_UNIQUE_NAME]" />
              <LNum name="[NUTS2006].[LEVEL_NUMBER]" />
              <DisplayInfo name="[NUTS2006].[DISPLAY_INFO]" />
            </HierarchyInfo>
          </AxisInfo>
          <AxisInfo name="Axis1">
            <HierarchyInfo name="Temporal">
              <UName name="[Temporal].[MEMBER_UNIQUE_NAME]" />
              <Caption name="[Temporal].[MEMBER_CAPTION]" />
              <LName name="[Temporal].[LEVEL_UNIQUE_NAME]" />
              <LNum name="[Temporal].[LEVEL_NUMBER]" />
              <DisplayInfo name="[Temporal].[DISPLAY_INFO]" />
            </HierarchyInfo>
          </AxisInfo>
          <AxisInfo name="SlicerAxis">
            <HierarchyInfo name="Measures">
              <UName name="[Measures].[MEMBER_UNIQUE_NAME]" />
              <Caption name="[Measures].[MEMBER_CAPTION]" />
              <LName name="[Measures].[LEVEL_UNIQUE_NAME]" />
              <LNum name="[Measures].[LEVEL_NUMBER]" />
              <DisplayInfo name="[Measures].[DISPLAY_INFO]" />
            </HierarchyInfo>
            <HierarchyInfo name="NUTS2003">
              <UName name="[NUTS2003].[MEMBER_UNIQUE_NAME]" />
              <Caption name="[NUTS2003].[MEMBER_CAPTION]" />
              <LName name="[NUTS2003].[LEVEL_UNIQUE_NAME]" />
              <LNum name="[NUTS2003].[LEVEL_NUMBER]" />
              <DisplayInfo name="[NUTS2003].[DISPLAY_INFO]" />
            </HierarchyInfo>
[... Description des autres Axes ...]
          </AxisInfo>
        </AxesInfo>
        <CellInfo>
          <Value name="VALUE" />
          <FmtValue name="FORMATTED_VALUE" />
          <FormatString name="FORMAT_STRING" />
        </CellInfo>
      </OlapInfo>
      <Axes>
        <Axis name="Axis0">
          <Tuples>
            <Tuple>
              <Member Hierarchy="NUTS2006">
                <UName>[NUTS2006].[All NUTS2006s].[Albania]</UName>
```

```
                    <Caption>Albania</Caption>
                    <LName>[NUTS2006].[NUTS2006 level 0]</LName>
                    <LNum>1</LNum>
                    <DisplayInfo>1</DisplayInfo>
                  </Member>
                </Tuple>
                <Tuple>
                  <Member Hierarchy="NUTS2006">
                    <UName>[NUTS2006].[All NUTS2006s].[Armenia]</UName>
                    <Caption>Armenia</Caption>
                    <LName>[NUTS2006].[NUTS2006 level 0]</LName>
                    <LNum>1</LNum>
                    <DisplayInfo>131073</DisplayInfo>
                  </Member>
                </Tuple>
[... Valeur de tous les membres affichés  ...]

              </Tuples>
            </Axis>
          </Axes>
          <CellData>
            <Cell CellOrdinal="0">
              <Value xsi:type="xsd:double">1861</Value>
              <FmtValue>1,861</FmtValue>
              <FormatString>Standard</FormatString>
            </Cell>
            <Cell CellOrdinal="1">
              <Value xsi:type="xsd:double">1861</Value>
              <FmtValue>1,861</FmtValue>
              <FormatString>Standard</FormatString>
            </Cell>
            <Cell CellOrdinal="2">
              <Value xsi:type="xsd:double">820621</Value>
              <FmtValue>820,621</FmtValue>
              <FormatString>Standard</FormatString>
            </Cell>
            <Cell CellOrdinal="3">
              <Value xsi:type="xsd:double">3722</Value>
              <FmtValue>3,722</FmtValue>
              <FormatString>Standard</FormatString>
            </Cell>
[... Valeur de toutes les cellules affichées  ...]
          </CellData>
        </root>
      </cxmla:return>
    </cxmla:ExecuteResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# *Annexe C : Extraits de la bibliothèque documentaire*

Le site généré par Maven contient des informations à la fois fonctionnelles, par l'ajout de pages APT ou HTML (le guide utilisateur est inclus à cette documentation), et techniques par l'utilisation des greffons dédiés. Cette annexe montre diverses pages produites par ce processus.

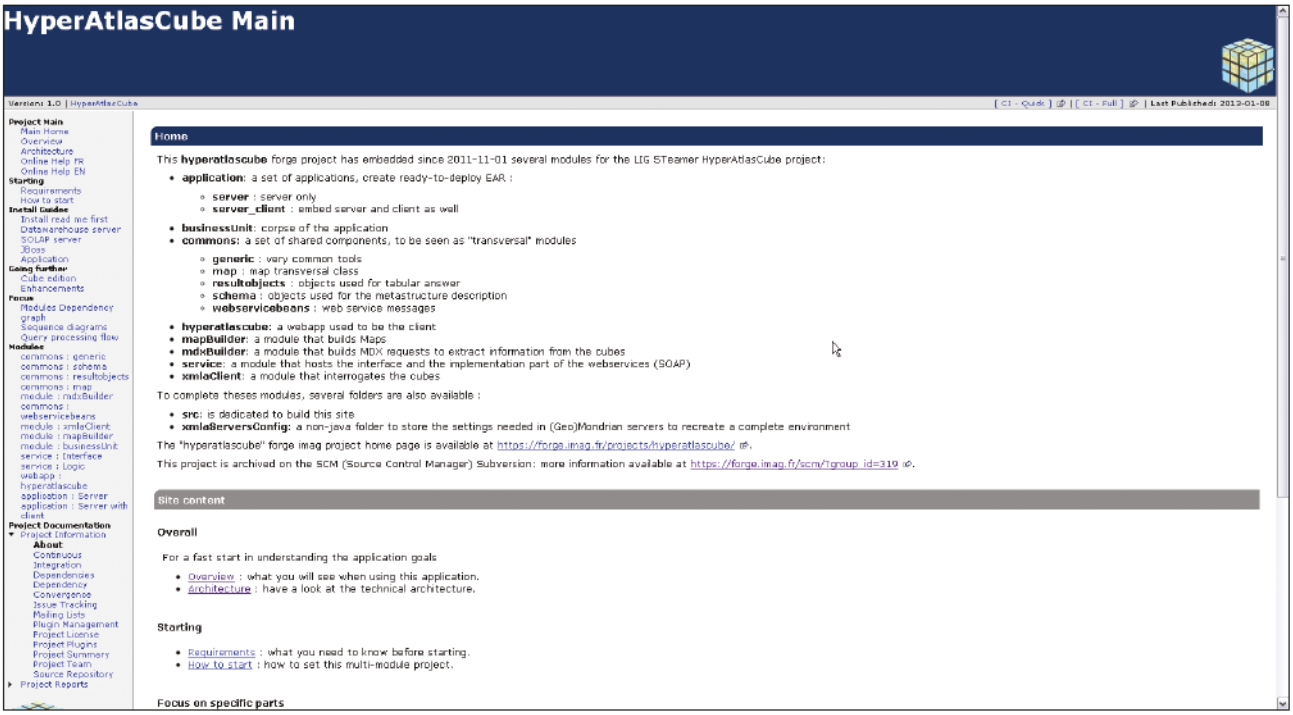



Figure A-C-1 : Page d'accueil du site

La figure A-C-1 illustre la première page du site conçu par Maven. Elle a été personnalisée afin d'offrir une vue d'ensemble du projet.

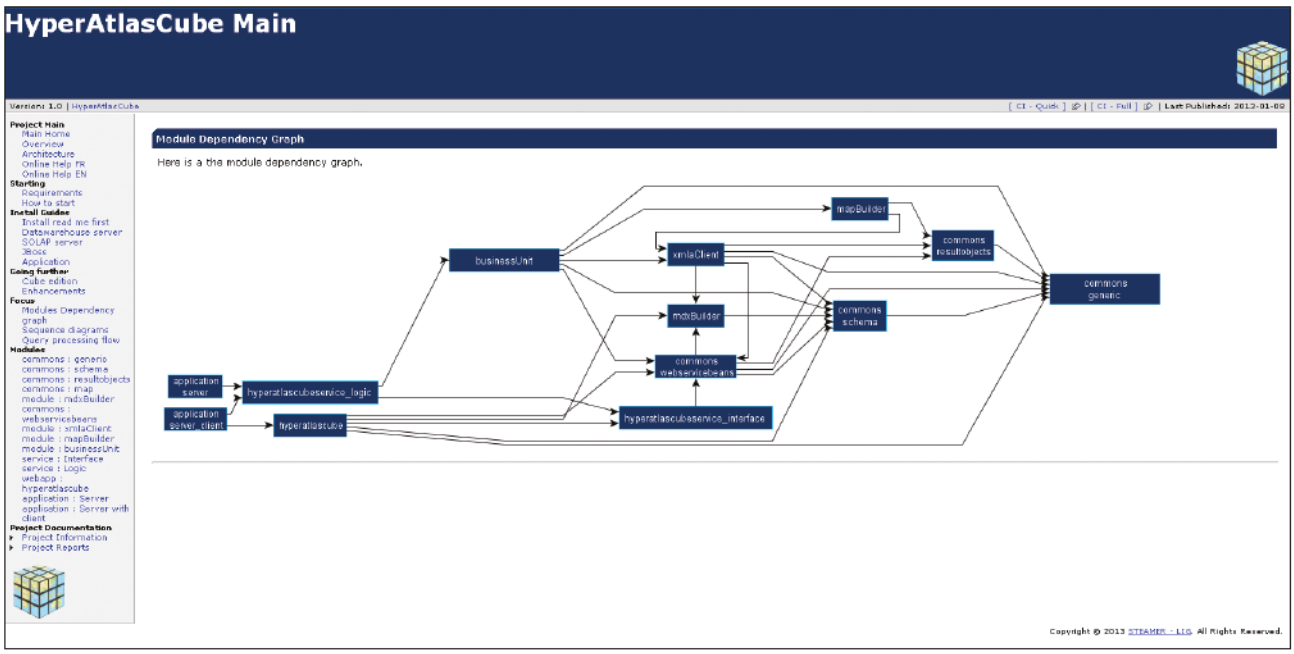



Figure A-C-2 : Page décrivant les dépendances entre les modules

La figure A-C-2 représente l'une des nombreuses pages informant sur l'architecture et le fonctionnement de l'application. Il s'agit ici du graphique de dépendances entre les modules.

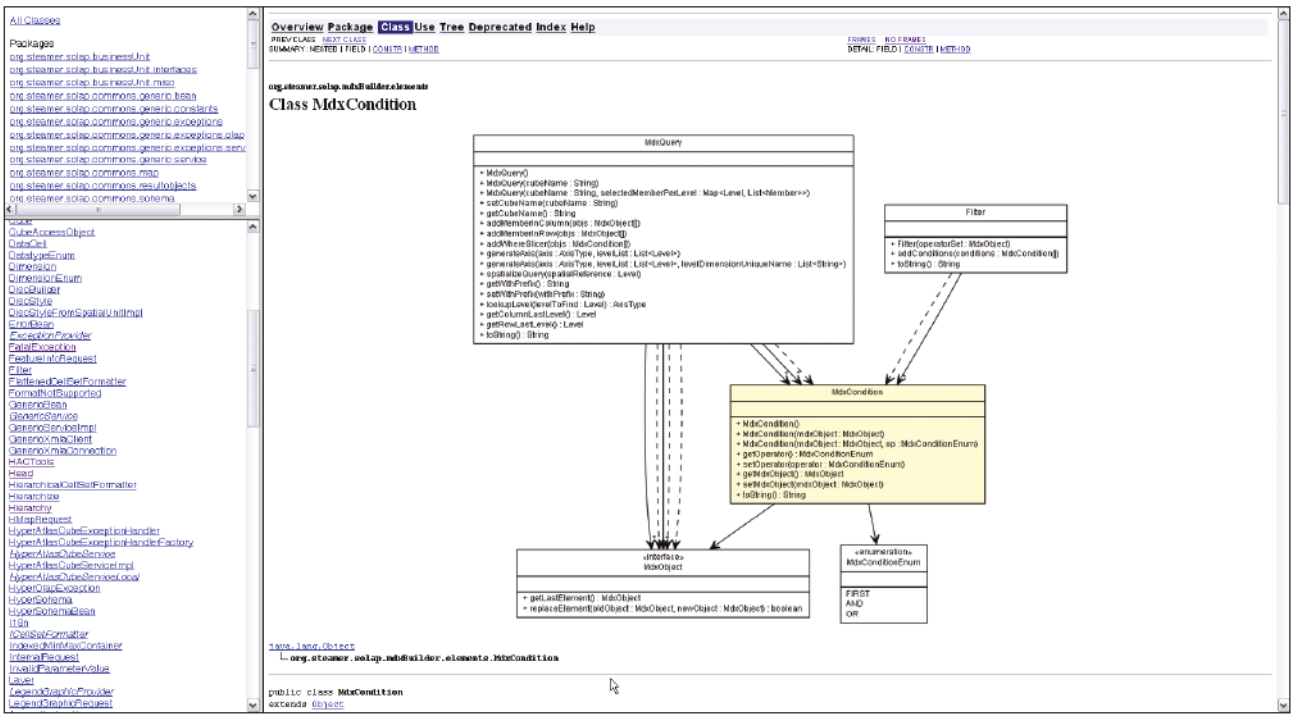



Figure A-C-3 : JavaDoc et diagramme de classe interactif de la classe MdxCondition

Livrable classique des projets Java, la JavaDoc est très utile au développeur. Les diagrammes de classes sont générés dynamiquement et immergent le lecteur dans le contexte d'utilisation de la classe (figure A-C-3).

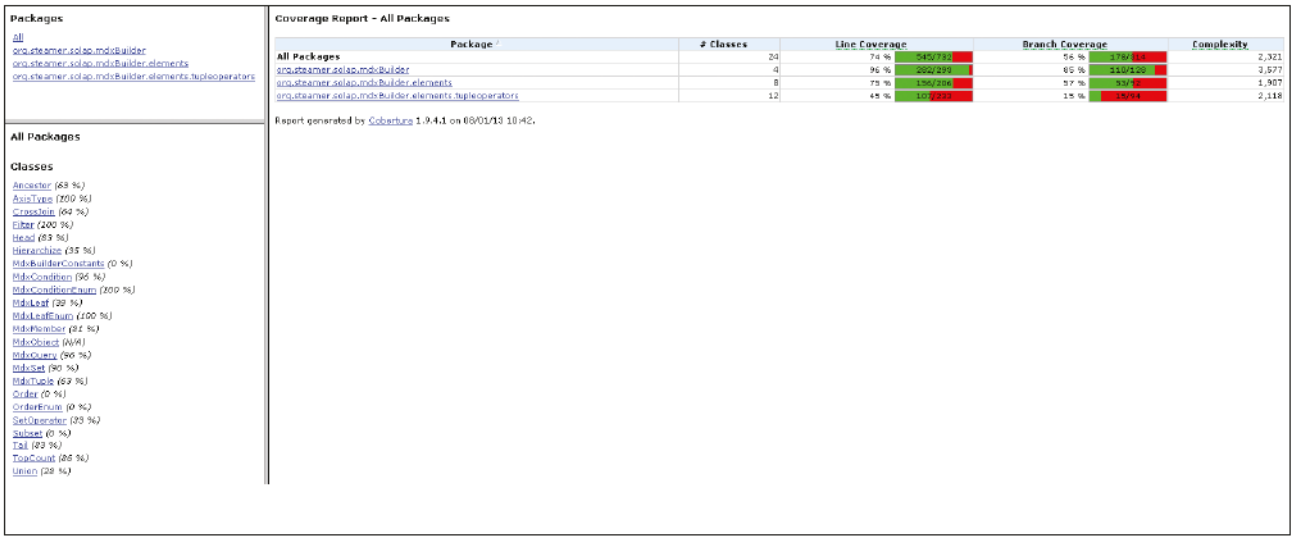



Figure A-C-4 : Synthèse de la couverture de code par les tests

De nombreux rapports et tableaux de synthèse apportent une lecture plus aisée des retours des greffons de Maven. Sur la figure A-C-4, le module « Corbertura » (http://cobertura.sourceforge.net) donne un aperçu de la couverture du code par les tests.

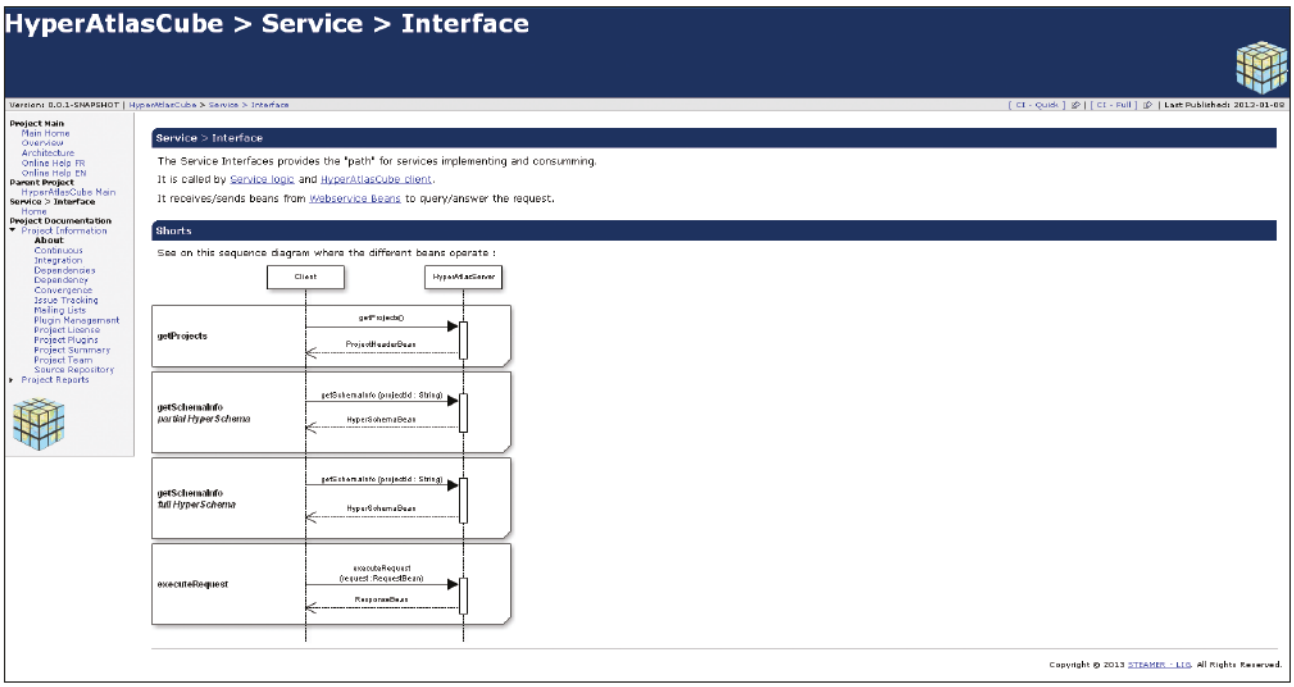



Figure A-C-5 : Informations détaillées, module par module

Enfin, bien que les synthèses existent et soient référencées au niveau du module père, il est possible d'accéder aux rapports et documents au niveau de chaque module, comme le montre la figure A-C-5 pour le module d'interface des services.

# *Annexe D : Description de l'entrepôt*

Ce fichier est le « traducteur » du contenu de l'entrepôt pour le serveur OLAP.

```
<Schema name="hyperatlascube">
  <!-- Description du cube -->
  <Cube name="Indicators" defaultMeasure="Measure" cache="true" enabled="true">
    <!-- Déclaration de la table de fait -->
    <Table name="fact" schema="public"></Table>
    <!-- Description de la dimension NUTS2006 et de ses quatre niveaux -->
    <Dimension type="StandardDimension" foreignKey="nuts2006_4_id" name="NUTS2006">
      <!-- Hiérarchie par défaut de la dimension -->
      <Hierarchy hasAll="true" primaryKey="nuts2006_4_id"
primaryKeyTable=»nuts2006_4»>
        <!-- Description du fonctionnement des tables entrant en jeu (en
flocon)-->
        <Join leftKey="nuts2006_3_id" rightKey="nuts2006_3_id">
          <Table name="nuts2006_4" schema="public"></Table>
          <Join leftKey="nuts2006_2_id" rightAlias="nuts2006_2"
                                                rightKey="nuts2006_2_id">
            <Table name="nuts2006_3" schema="public"></Table>
            <Join leftKey="nuts2006_1_id" rightKey="nuts2006_1_id">
              <Table name="nuts2006_2" schema="public"></Table>
              <Table name="nuts2006_1" schema="public"></Table>
            </Join>
          </Join>
        </Join>
        <!-- Déclaration des niveaux, des clés, et des géométries (Property) -->
        <Level name="NUTS2006 level 0" table="nuts2006_1" column="code"
                    nameColumn="name" type="String" uniqueMembers="true"
                    levelType="Regular" hideMemberIf="Never">
          <Property name="geom" column="geometry" type="Geometry" />
        </Level>
        <Level name="NUTS2006 level 1" table="nuts2006_2" column="code"
nameColumn="name" type="String" uniqueMembers="true"
levelType="Regular" hideMemberIf="Never">
          <Property name="geom" column="geometry" type="Geometry" />
        </Level>
        <Level name="NUTS2006 level 2" table="nuts2006_3" column="code"
nameColumn="name" type="String" uniqueMembers="true"
levelType="Regular" hideMemberIf="Never">
          <Property name="geom" column="geometry" type="Geometry" />
        </Level>
        <Level name="NUTS2006 level 3" table="nuts2006_4" column="code"
nameColumn="name" type="String" uniqueMembers="true"
levelType="Regular" hideMemberIf="Never">
          <Property name="geom" column="geometry" type="Geometry" />
        </Level>
      </Hierarchy>
    </Dimension>
```

```
    <!-- Description de la dimension NUTS2003 et de ses quatre niveaux -->
    <Dimension type="StandardDimension" foreignKey="nuts2003_4_id" name="NUTS2003">
      <Hierarchy hasAll="true" primaryKey="nuts2003_4_id"
                                                    primaryKeyTable="nuts2003_4">
        <Join leftKey="nuts2003_3_id" rightKey="nuts2003_3_id">
          <Table name="nuts2003_4" schema="public"></Table>
          <Join leftKey="nuts2003_2_id" rightAlias="nuts2003_2
 rightKey="nuts2003_2_id">
            <Table name="nuts2003_3" schema="public"></Table>
            <Join leftKey="nuts2003_1_id" rightKey="nuts2003_1_id">
              <Table name="nuts2003_2" schema="public"></Table>
              <Table name="nuts2003_1" schema="public"></Table>
            </Join>
          </Join>
        </Join>


        <Level name="NUTS2003 level 0" table="nuts2003_1" column="code"
 nameColumn="name" type="String"
 levelType="Regular" hideMemberIf="Never">
             <Property name="geom" column="geometry" type="Geometry" />
        </Level>
        <Level name="NUTS2003 level 1" table="nuts2003_2" column="code"
                    nameColumn="name" type="String"
levelType="Regular" hideMemberIf="Never">
             <Property name="geom" column="geometry" type="Geometry" />
        </Level>
        <Level name="NUTS2003 level 2" table="nuts2003_3" column="code"
                    nameColumn="name" type="String"
levelType="Regular" hideMemberIf="Never">
             <Property name="geom" column="geometry" type="Geometry" />
        </Level>
        <Level name="NUTS2003 level 3" table="nuts2003_4" column="code"
                    nameColumn="name" type="String" uniqueMembers="true"
 levelType="Regular" hideMemberIf="Never">
             <Property name="geom" column="geometry" type="Geometry" />
        </Level>
      </Hierarchy>
    </Dimension>

    <!-- Description de la dimension Temporelle et de ses trois niveaux -->
    <Dimension type="TimeDimension" foreignKey="temporal_id" name="Temporal">
      <Hierarchy hasAll="false" primaryKey="temporal_id">
        <Table name="data_temporal" schema="public"></Table>
        <Level name="Year" column="temporal_year" type="Numeric"
uniqueMembers="true" levelType="TimeYears" hideMemberIf="Never">
        </Level>
        <Level name="Month" column="temporal_month" type="Numeric"
             uniqueMembers="false" levelType="TimeMonths" hideMemberIf="Never">
        </Level>
        <Level name="Day" column="temporal_day" type="Numeric"
```

```
            uniqueMembers="false" levelType="TimeDays" hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>

    <!-- Description de la dimension INDICATOR et de ses deux niveaux -->
    <Dimension type="StandardDimension" foreignKey="indicator_id" name="Indicator">
      <Hierarchy hasAll="true" primaryKey="indicator_id">
        <Table name="indicator" schema="public"></Table>
        <Level name="Super Indicator" column="sup_indicator_name"
                  nameColumn="sup_indicator_name" uniqueMembers="false">
        </Level>
        <Level name="Indicator Name" column="indicator_fullname" type="String"
uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>

    <!-- Description de la dimension CLC2000 et de ses trois niveaux -->
    <Dimension type="StandardDimension" foreignKey="clc2000_3_id" name="CLC2000">
      <Hierarchy hasAll="true">
        <Table name="clc2000" schema="public"></Table>
        <Level name="CLC2000 level 1" column="clc2000_1_id"
            nameColumn="clc2000_1_name" type="String"
levelType="Regular" hideMemberIf="Never"
description="Highest level for Corinne Land Cover 2000">
        </Level>
        <Level name="CLC2000 level 2" column="clc2000_2_id"
            nameColumn="clc2000_2_name"  type="String" levelType="Regular"
            hideMemberIf="Never">
        </Level>


        <Level name="CLC2000 level 3" column="clc2000_3_id"
            nameColumn="clc2000_3_name" type="String"
uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>

    <!-- Description de la dimension CLC2006 et de ses trois niveaux -->
    <Dimension type="StandardDimension" foreignKey="clc2006_3_id" name="CLC2006">
      <Hierarchy hasAll="true">
        <Table name="clc2006" schema="public"></Table>
        <Level name="CLC2006 level 1" column="clc2006_1_id"
nameColumn="clc2006_1_name" type="String"
levelType="Regular" hideMemberIf="Never"
description="Highest level for Corinne Land Cover 2006">
        </Level>
        <Level name="CLC2006 level 2" column="clc2006_2_id"
            nameColumn="clc2006_2_name" type="String"
levelType="Regular" hideMemberIf="Never">
```

```
        </Level>
        <Level name="CLC2006 level 3" column="clc2006_3_id"
             nameColumn="clc2006_3_name" type="String"
uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>

    <!-- Description de la dimension Bassins Versants et de ses trois niveaux
-->
    <Dimension type="StandardDimension" foreignKey="rbd_3_id" name="RDB">
      <Hierarchy hasAll="true">
        <Table name="rbd" schema="public"></Table>
        <Level name="RDB level 0" column="rbd_0_id" nameColumn="rbd_0_name"
             type="String" levelType="Regular" hideMemberIf="Never">
        </Level>
        <Level name="RDB level 1" column="rbd_1_id" nameColumn="rbd_1_name"
             type="String" levelType="Regular" hideMemberIf="Never">
        </Level>
        <Level name="RDB level 2" column="rbd_2_id" nameColumn="rbd_2_name"
 type="String" levelType="Regular" hideMemberIf="Never">
        </Level>
        <Level name="RDB level 3" column="rbd_3_id" nameColumn="rbd_3_name"
type="String" uniqueMembers="true"
levelType="Regular" hideMemberIf="Never">
             <Property name="geom" column="geometry" type="Geometry" />
        </Level>
      </Hierarchy>
    </Dimension>

    <!-- Description de la mesure -->
    <Measure name="Measure" column="fact_value" datatype="String"
             formatString="Standard" aggregator="sum" visible="true">
    </Measure>
  </Cube>
</Schema>
```

# Glossaire

Ce glossaire référence les termes signalés d'un astérisque tout au long de ce mémoire et en propose une définition.

**Donnée nominale :** Type de variable décrivant un nom ou une catégorie, sans ordre naturel.

**Donnée numérique :** Donnée quantitative qui décrit une valeur mesurée de façon numérique. Elle peut être continue ou discrète.

**Europe des 28 :** Ensemble des 28 pays membres de l'Union Européenne, au 1er janvier 2013 : Allemagne, Autriche, Belgique, Bulgarie, Chypre, Croatie, Danemark, Espagne, Estonie, Finlande, France, Grèce, Hongrie, Irlande, Italie, Lettonie, Lituanie, Luxembourg, Malte, Pays-Bas, Pologne, Portugal, République tchèque, Roumanie, Royaume-Uni, Slovaquie, Slovénie et Suède.

**Géomatique :** Contraction des termes « géographie » et « informatique ». C'est l'ensemble des techniques de traitement informatique des données géographiques (Journal officiel du 14 février 1994). Celles-ci permettent de modéliser, de représenter et d'analyser le territoire pour en faire des représentations virtuelles.

**Multiscalaire :** Démarche ayant pour but la compréhension de l'organisation et de l'aménagement du territoire à plusieurs échelles.

**Sémiologie :** Science des signes et de la signification. La sémiologie graphique est l'étude des signes graphiques, de leurs propriétés et de leurs rapports, avec les éléments d'information qu'ils expriment.

**Système de référence spatiale :** Système de coordonnées utilisé pour donner à chaque point d'une image des coordonnées. Ce système peut être géographique ou projeté.

**Troisième forme normale :** Les formes normales sont appliquées dans les bases de données relationnelles. Respecter la troisième forme normale prévient la redondance des données, et permet alors une accélération conséquente des requêtes de sélection.

# Bibliographie

Tous les sites Internet cités ont été consultés entre octobre 2011 et juillet 2012.


**[Analytical-Labs, 2012]** Analytical-Labs. *Saiku - Next Generation Open Source Analysis*. 2012. [en ligne]. Disponible sur : http://analytical-labs.com/

**[Andrienko, 1999]** Andrienko G., Andrienko N. *Interactive Maps for Visual Data Exploration. International Journal of Geographical Information Science*,1999, vol. 13, no. 4, p. 355-374.

**[Becker, 2003]** Becker B. *Another Look At Degenerate Dimensions.* 2003. [en ligne]. Disponible sur : http://www.kimballgroup.com/html/designtipsPDFDesignTips2003/KimballDT46AnotherLook.pdf

**[Bédard *et al.*, 1998]** Bédard Y., Létourneau F., Moulin B. *Perspectives d'utilisation du concept d'entrepôt de données pour les géorépertoires sur Internet. Geomatica*, 1998, vol. 52, no. 2, p. 145-163.

**[Bédard *et al.*, 2005]** Bédard Y., Proulx M.J., Rivest S. *Enrichissement de OLAP pour l'analyse géographique : Exemples de réalisations et différentes possibilités technologiques. 2005.* In : *1ère Journée Francophone sur les Entrepôts de Données et l'Analyse en ligne (EDA 2005)*, Université Lyon-II, Lyon, 10 juin 2005. [en ligne]. Disponible sur : http://sirs.scg.ulaval.ca/YvanBedard/reserve/downloads/slidewhow/398.ppt

**[Bédard, 2008]** Bédard Y. *OLAP et SOLAP : notions avancées des bases de données SIG.* 2008. [en ligne]. Disponible sur : http://www.scribd.com/doc/6964842/OLAP-et-SOLAP-complet-avec-explication-ppt-univ-laval

**[Béguin & Pumain, 2003]** Béguin M., Pumain D, 2003. *La représentation des données géographiques, statistique et cartographie.* 2nde édition, Armand Colin, Paris, 192 p.

**[Bimonte *et al.*, 2005]** Bimonte S., Miquel M., Tchounikine A., 2005. T*owards a Spatial Multidimensional Model.* Publication du Laboratoire d'InfoRmatique en Images et Systèmes d'information, Université Lyon-II. [en ligne]. Disponible sur : http://www.cis.drexel.edu/faculty/song/dolap/dolap05/paper/p39-bimonte.pdf

**[Boulil, 2012]** Boulil K., 2012. *Une Approche Automatisée basée sur des Contraintes d'Intégrité définies définies en UML et OCL pour la Vérification de la Cohérence Logique dans les Systèmes SOLAP - Applications dans le domaine agri-environnemental.* Thèse en informatique, IRSTEA, Clermont-Ferrand.

**[Briggs, 2009]** *Maps*. 2009. In : Briggs *S. Sean's Website.* [en ligne]. Disponible sur : http://personal.frostburg.edu/sbriggs0/maps.htm

**[Browning & Mundy, 2001]** *Data Warehouse Design Consideration.* 2001. In: Browning D. & Mundy J., *MSDN*. [en ligne]. Disponible sur : http://msdn.microsoft.com/en-us/library/aa902672(v=sql.80).aspx

**[Chabert, 2007]** Chabert C., 2007. *HyperAtlas et HyperAdmin : des outils cartographiques pour l'analyse de phénomènes sociaux.* Mémoire ingénieur en informatique, CNAM Rhône-Alpes, 152 p.

**[Choquet & Boussaïd, 2007]** Choquet R., Boussaïd O., 2007. *Interrogation OLAP d'un entrepôt de données XML.* Publication du laboratoire ERIC, Université Lyon-II, 12 p.

**[Choquet, 2006]** Choquet R., 2006. *Interrogation OLAP sur documents XML.* Mémoire Master 2, Université Lyon-II, 27 p.

**[CNES, 2007]** Centre National d'Études Spatiales, 2007. *Comparatif des bases de données spatialisées*. [en ligne]. Disponible sur : http://cct.cnes.fr/cct05/public/2007/seminaires/etude_comp_BD_spatialisees/comparatif_bds.pdf

**[Codd *et al.*, 1993]** Codd E.F., Codd S.B., Salley C.T., 1993. *Providing OLAP (On-line Analytical Processing) to User-Analysts : An IT mandate.* [en ligne]. Disponible sur : http://www.minet.uni-jena.de/dbis/lehre/ss2005/sem_dwh/lit/Cod93.pdf

**[Desnos, 2005]** Desnos J.F., 2005. *Entrepôt de Données.* Cours Master. [en ligne]. Disponible sur : http://prevert.upmf-grenoble.fr/SpecialiteIHS/GP/coursED.pdf

**[Développez.net, 2006]** *Comment distinguer et utiliser ROLAP/MOLAP/HOLAP ?* 2006. In : *Forum des professionnels en informatique*. [en ligne]. Disponible sur : http://www.developpez.net/forums/d86373/logiciels/solutions-dentreprise/business-intelligence/rolap-molap-holap-distinguer-utiliser/

**[Dinimant, 2009]** *Qu'est-ce que la modélisation multidimensionnelle ?* 2009. In : Dinimant A., *Blogs des professionnels en informatique*. [en ligne]. Disponible sur : http://blog.developpez.com/bi/p7422/informations/modelisation-multidimensionnelle/

**[Donsez, 2006]** Donsez D., 2006. *Principes et architectures des entrepôts de données.* Cours Master. [en ligne].
Disponible sur : http://membres-liglab.imag.fr/donsez/cours/bddwdm.pdf

**[Eden-IGN, 2012]** *ISO 19115*. 2012. In : Equipe d'experts en normalisation de l'IGN (EDEN), *Eden IGN.* [en ligne].
Disponible sur : http://eden.ign.fr/std/iso_19115

**[ESPON, 2012]** *Plateforme scientifique des projets ESPON.* 2012. In : *ESPON*. [en ligne].
Disponible sur : http://www.espon.eu/main/Menu_Projects/Menu_ScientificPlatform/

**[Gouvernement, 2011]** *CORINE Land Cover*. 2011. In : Gouvernement > Ministère de l'écologie, du développement durable et de l'énergie. *Observations et statistiques*. [en ligne].
Disponible sur : http://www.statistiques.developpement-durable.gouv.fr/donnees-ligne/t/nomenclature.html

**[Hachicha *et al.*, 2007]** Hachicha M., Mahboubi H., Darmont J., 2007. *Vers une algèbre XML-OLAP : État de l'art.* Publication du laboratoire ERIC, Université Lyon-II, 16 p.

**[iCCube, 2012]** *Members, Tuples and Sets.* In : iCCube. *OLAP Server and Business Intelligence Reporting.* 2012. [en ligne].
Disponible sur : http://www.iccube.com/support/documentation/mdx_tutorial/members.html

**[IMAG, 2011]** IMAG, 2011. *HyperCarte.* [en ligne].
Disponible sur : http://hypercarte.imag.fr/

**[Inmon, 2005]** Inmon W., 2005. *Building the Data Warehouse.* 4ème édition, Hungry Minds Inc, États-Unis, 576 p.

**[ISO, 2009]** Organisation internationale de normalisation, 2009. *Information géographique - Métadonnées - Partie 2 : Extensions pour les images et les matrices.*

**[Jensen & Torben, 2001]** Jensen C, Torben P.B., 2001. *Multidimensional Database Technology. Distributed Systems Online (IEEE)*, Décembre 2001, p. 40-46.

**[Johnston, 2011]** *Life map.* 2011. In : Johnston K. *Culture and the Arts.* [en ligne].
Disponible sur : http://artsbeat.blogs.nytimes.com/2011/08/09/what-digital-maps-can-tell-us-about-the-american-way/

**[Joly, 1976]** Joly F. ,1976. *La cartographie.* Presses universitaires de France (PUF), p. 9.

**[Le Rubrus, 2009]** Le Rubrus B., 2009. *Capacité de rendu cartographique autour des technologies SOLAP.* Épreuve TEST UEENG111, CNAM Rhône-Alpes, 42 p.

**[Le Rubrus, 2011]** Le Rubrus B., 2011. *Cartographie et analyse territoriale multiscalaire. Réingénierie des logiciels HyperAtlas et HyperAdmin.* Mémoire ingénieur en informatique, CNAM Rhône-Alpes, 143 p.

**[Lebrun & Charrier, 2008]** Lebrun G. & Charrier C., 2008. *Informatique Décisionnelle.* Cours Master UENFE115.

**[Leobet & Merrien, 2011]** Leobet M., Merrien F., 2011. *La directive INSPIRE pour les néophytes (Troisième édition).* Vulgarisation de la directive INSPIRE. [en ligne]. Disponible sur : http://georezo.net/blog/inspire/files/2011/12/La_directive_Inspire_pour_les_neophytes_V3.pdf

**[LGS Group, 2000]** LGS Group, 2000. *Analysis of Health Surveillance Business Intelligence Tools and Applications : Analysis Methodology and Criteria. Project report.* 19 p.

**[Lizzi & Grasland, 2004]** Lizzi L. & Grasland C., 2004. *Third Interim Report : ESPON project 3.1. Integrated tools for European spatial development. Annex A : Multiscalar territorial analysis.*[en ligne]. Disponible sur : http://www.ums-riate.fr/documents/MTA_4th%20version-livret2.pdf

**[Loudcher, 2011]** Loudcher S., 2011. *Vers l'OLAP sémantique pour l'analyse en ligne des données complexes*. Habilitation à diriger des Recherches, Laboratoire ERIC, Université Lyon-II, 116p.

**[Lupin, 2007]** Lupin B., 2007. *Osez OLAP - Les bases de données OLAP par l'exemple.* [en ligne]. Disponible sur : http://bernard.lupin.pagesperso-orange.fr/

**[Maven (A), 2012]** Maven, 2012. *Présentation de Maven.* [en ligne]. Disponible sur : http://maven.apache.org/what-is-maven.html

**[Maven (B), 2012]** Maven, 2012. *Introduction sur le cycle de vie de Maven*. [en ligne]. Disponible sur : http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

**[Meier, 2006]** Meier A., 2006. *Introduction pratique aux bases de données relationnelles.* Seconde édition, Springer Editions, France, p. 197-203.

**[Microsoft & Hyperion, 2002]** Microsoft Corporation & Hyperion Solutions Corporation, 2002. *XML for Analysis Specification*. [en ligne]. Disponible sur : http://news.xmlforanalysis.com/docs/xmla1.1.doc

**[Milego& Ramos, 2011]** Milego R. & Ramos M.J., 2011. *Disaggregation of socioeconomic data into a regular grid and combination with other types of data.* Rapport Technique présenté dans le cadre du projet ESPON 2013 Database, 37 p.

**[Milego, 2012]** Milego R., 2012. *The ESPON OLAP Cubes : a tool for combining and analysing heterogeneous data.* In : GISCO Working Party, 8 et 9 mars 2012.

**[MSDN MDX, 2012]** Microsoft Developer Network, 2012. *Multidimensional Expressions (MDX) Reference.* [en ligne].
Disponible sur : http://msdn.microsoft.com/en-us/library/ms145506

**[Noirault, 2006]** Noirault C., 2006. *Business Intelligence avec Oracle 10g: ETL, Data warehouse, Data mining, rapports. Collection Informatique Technique, Editions Eni, France,* p. 133-138.

**[Palsky, 1996]** Palsky G., 1996. *Des chiffres et des cartes. Naissance et développement de la cartographie quantitative au XIXe siècle.* Paris, Comité des travaux historiques et scientifiques, 331p.

**[Plumejeaud, 2007]** Plumejeaud C., 2007. *Acquisition de données et cartes de potentiel pour l'analyse spatiale.* Mémoire ingénieur en informatique, CNAM Rhône-Alpes, 136 p.

**[Plumejeaud, 2011]** Plumejeaud C., 2011. *Modèles et méthodes pour l'information spatio-temporelle évolutive.* Thèse en informatique, IMAG, 308 p.

**[Poulenard, 2011]** Poulenard L., 2011. *Diffusion de l'information géographique avec les services Web : application au logiciel HyperAtlas.* Mémoire ingénieur en informatique, CNAM Rhône-Alpes, 131 p.

**[Ross, 2003]** Ross M., 2003. *De-Clutter With Junk (Dimensions).* [en ligne].
Disponible sur : http://www.kimballgroup.com/html/designtipsPDF/DesignTips2003/KimballDT48DeClutter.pdf

**[Rougé-Libourel, 2008]** Rougé-Libourel T., 2008. *Entrepôts de données spatiales OLAP et SOLAP.* Cours sur les bases de données spatiales. [en ligne].
Disponible sur : http://www.lirmm.fr/~libourel/FMIN206/cours11_BDS-OlapSolap.pdf

**[Serna Encinas, 2005]** Serna Encinas M.T., 2005. *Entrepôts de données pour l'aide à la décision médicale : conception et expérimentation.* Thèse en informatique, IMAG, 152p.

**[Shana, 2012]** Shana G., 2012. *Visualizations (CS-64/171).* Cours de l'Université d'Harvard SEAS. [en ligne].
Disponible sur : http://www.cs171.org/2012/final-projects/web/21_FoodborneIllness/

**[Smile, 2012]** Smile, 2012. *Décisionnel, le meilleur des solutions open source.* Livre Blanc disponible en ligne, sur demande. [en ligne].
Disponible sur : http://www.smile.fr/Livres-blancs/ERP-et-decisionnel/Le-decisionnel-open-source

**[Statistique Canada, 2010]** Statistique Canada, 2010. Plusieurs pages ont été consultées :
*Diagramme circulaire.* [en ligne].
Disponible sur : http://www.statcan.gc.ca/edu/power-pouvoir/ch9/pie-secteurs/5214826-fra.htm
*Diagramme à bandes.* [en ligne].
Disponible sur : http://www.statcan.gc.ca/edu/power-pouvoir/ch9/bargraph-diagrammeabarres/5214818-fra.htm
*Diagramme linéaire.* [en ligne].
Disponible sur : http://www.statcan.gc.ca/edu/power-pouvoir/ch9/line-lineaire/5214824-fra.htm

**[Techtarget, 2000]** *Hybrid OLAP definition.* In : Techtarget. *Techtarget, where serious technology buyers decide* . 2000. [en ligne].
Disponible sur : http://searchsqlserver.techtarget.com/definition/hybrid-online-analytical-processing

**[Techtarget, 2009]** *Similarities and differences between ROLAP, MOLAP and HOLAP.* In : Techtarget. *Techtarget, where serious technology buyers decide.* 2009. [en ligne].
Disponible sur : http://searchdatamanagement.techtarget.com/feature/Similarities-and-differences-between-ROLAP-MOLAP-and-HOLAP

**[Thomas, 2008]** Thomas R., 2008. *Evolutions d'outils dédiés à l'analyse territoriale et à l'analyse spatiale dans le cadre du projet HyperCarte.* Mémoire ingénieur en informatique, CNAM Rhône-Alpes, 112 p.

**[Touchart, 2004]** Touchart L., 2004. *Bassin-versant*. In : *Hypergéo, site encyclopédique consacrée à l'épistémologie de la géographie*. [en ligne].
Disponible sur : http://www.hypergeo.eu/spip.php?article408

**[Tranchant, 2011]** Tranchant M., 2011. *Capacités des outils SOLAP en termes de requêtes spatiales, temporelles et spatio-temporelles.* Épreuve TEST UEENG111, CNAM Rhône-Alpes, 40 p.

**[Visual Literacy, 2011]** *A periodic table of visualization methods.* In : *Visual Literacy: An E-Learning Tutorial on Visualization for Communication, Engineering and Business.* 2011. [en ligne].
Disponible sur : http://www.visual-literacy.org/periodic_table/periodic_table.html

**[Wikipedia, 2012]** *Thematic maps.* In : *Wikipedia, the free encyclopedia*. 2012. [en ligne].
Disponible sur : http://en.wikipedia.org/wiki/Thematic_map

# MÉMOIRE D'INGÉNIEUR C.N.A.M. en INFORMATIQUE

## Couplage d'un HyperCube SOLAP et d'un outil de visualisation

**Michaël TRANCHANT** **à Grenoble le, 22 février 2013.**

---

**Résumé**

À la croisée des mondes décisionnel et géographique, le traitement spatial et analytique en ligne, plus communément nommé *Spatial OLAP* (SOLAP), permet la manipulation et l'exploitation de la dimension spatiale des informations contenues dans l'hypercube. Les outils de visualisation, quant à eux, constituent l'interface avec laquelle interagissent les utilisateurs sur tout système d'information, leur permettant un accès aux fonctionnalités d'interrogation de l'application ainsi qu'à l'affichage des résultats. Ces deux éléments, intégrés au sein d'une chaîne décisionnelle, offrent une solution complète pour l'exploration et la représentation de la perspective spatiale tout en profitant des capacités des outils OLAP.
Dans cette version « cube », le prototype HyperAtlas³ exploite à la fois les capacités OLAP des systèmes décisionnels et permet un rendu cartographique des géométries stockées dans l'entrepôt de données.

**Mots-clés :** SOLAP, HyperCube, entrepôt de données, visualisation de l'information, carte, diagramme, tableau croisé multidimensionnel, HyperAtlas³.

---

**Abstract**

At the crossroads of Business Intelligence and geographic worlds, Spatial On-Line Analytical Processing, more commonly known as Spatial OLAP (SOLAP), allows the manipulation and the exploitation of the spatial information contained in the hypercube. In information systems, users interact with visualization tools, which compose the interface, allowing them to access the application's query functionalities and displaying capabilities. These two elements, integrated in a decision-making chain, provide a complete solution for exploring and representing the spatial perspective while enjoying the OLAP capabilities.
In this "cube" version, the HyperAtlas³ prototype operates with both decision-making systems OLAP capabilities and allows a cartographic rendering of geometries that are stored in the data warehouse.

Keywords : SOLAP, HyperCube, datawarehouse, data rendering, map, diagram, multidimensional pivot table, HyperAtlas³.