Une approche automatisée basée sur des contraintes d'intégrité définies en UML et OCL pour la vérification de la cohérence logique dans les systèmes SOLAP : applications dans le domaine agri-environnemental

Kamal Boulil

▶ **To cite this version:**

Kamal Boulil. Une approche automatisée basée sur des contraintes d'intégrité définies en UML et OCL pour la vérification de la cohérence logique dans les systèmes SOLAP : applications dans le domaine agri-environnemental. Autre [cs.OH]. Université Blaise Pascal - Clermont-Ferrand II, 2012. Français. NNT : 2012CLF22285 . tel-00923497

HAL Id: tel-00923497

https://theses.hal.science/tel-00923497v1

Submitted on 3 Jan 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° d'ordre : 2285
EDSPIC : 581

Université Blaise Pascal – Clermont-Ferrand II

École Doctorale
Sciences Pour l'Ingénieur de Clermont-Ferrand

THÈSE

Présentée par

Kamal BOULIL

Pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

Spécialité : INFORMATIQUE

Une Approche Automatisée basée sur des Contraintes d'Intégrité définies en UML et OCL pour la Vérification de la Cohérence Logique dans les Systèmes SOLAP - Applications dans le domaine agri-environnemental

---

Soutenue publiquement le Vendredi 26 octobre 2012 devant le jury composé de

Michel SCHNEIDER	Professeur (Université Blaise Pascal, Clermont-Fd)	Président du jury
Yvan BEDARD	Professeur (Université Laval, Canada)	Rapporteur
Jérôme GENSEL	Professeur (Université Pierre Mendès France, Grenoble)	Rapporteur
Ronan TOURNIER	Maître de conférence (Université Paul Sabatier, Toulouse)	Examinateur
Sandro BIMONTE	Chargé de recherche (Irstea, Centre de Clermont-Fd)	Co-directeur de thèse
François PINET	Chargé de recherche - HDR (Irstea, Centre de Clermont-Fd)	Directeur de thèse

Résumé

Les systèmes d'Entrepôts de Données et OLAP spatiaux (EDS et SOLAP) sont des technologies d'aide à la décision permettant l'analyse multidimensionnelle de gros volumes de données spatiales. Dans ces systèmes, la qualité de l'analyse dépend de trois facteurs : la qualité des données entreposées, la qualité des agrégations et la qualité de l'exploration des données.
La qualité des données entreposées dépend de critères comme la précision, l'exhaustivité et la cohérence logique. La qualité d'agrégation dépend de problèmes structurels (e.g. les hiérarchies non strictes qui peuvent engendrer le comptage en double des mesures) et de problèmes sémantiques (e.g. agréger les valeurs de température par la fonction Sum peut ne pas avoir de sens considérant une application donnée). La qualité d'exploration est essentiellement affectée par des requêtes utilisateur inconsistantes (e.g. quelles ont été les valeurs de température en URSS en 2010 ?). Ces requêtes peuvent engendrer des interprétations erronées des résultats.
Cette thèse s'attaque aux problèmes d'incohérence logique qui peuvent affecter les qualités de données, d'agrégation et d'exploration. L'incohérence logique est définie habituellement comme la présence de contradictions dans les données. Elle est typiquement contrôlée au moyen de Contraintes d'Intégrité (CI).
Dans cette thèse nous étendons d'abord la notion de CI (dans le contexte des systèmes SOLAP) afin de prendre en compte les incohérences relatives aux agrégations et requêtes utilisateur. Pour pallier les limitations des approches existantes concernant la définition des CI SOLAP, nous proposons un Framework basé sur les langages standards UML et OCL. Ce Framework permet la spécification conceptuelle et indépendante des plates-formes des CI SOLAP et leur implémentation automatisée. Il comporte trois parties :

(1) Une classification des CI SOLAP.

(2) Un profil UML implémenté dans l'AGL MagicDraw, permettant la représentation conceptuelle des modèles des systèmes SOLAP et de leurs CI.

(3) Une implémentation automatique qui est basée sur les générateurs de code Spatial OCL2SQL et UML2MDX qui permet de traduire les spécifications conceptuelles en code au niveau des couches EDS et serveur SOLAP.

Enfin, les contributions de cette thèse ont été appliquées dans le cadre de projets nationaux de développement d'applications (S)OLAP pour l'agriculture et l'environnement.

Remerciements

***J**e tiens à remercier très sincèrement M. François Pinet, Chargé de recherche – HDR à Irstea, Centre de Clermont-Ferrand, pour m'avoir proposé cette thèse, pour l'avoir dirigée ensuite,* pour sa gentillesse et son soutien dans des moments difficiles. *Je lui suis très reconnaissant pour m'avoir permis de faire mes premiers pas dans le monde de la recherche.*

Je remercie très sincèrement, MM. Emmanuel HUGO et Jean-Pierre Chanet pour m'avoir accueilli au sein de l'unité TSCF et m'avoir offert un cadre idéal pour réaliser et communiquer mes travaux.

Je remercie très sincèrement l'Irstea et le conseil régional d'auvergne qui ont financé cette thèse.

Je tiens à remercier très sincèrement M. Sandro Bimonte, Chargé de recherche à Irstea, Centre de Clermont-Ferrand, pour avoir suivi et encadré cette thèse, pour ses judicieux conseils ainsi que son aide précieuse pour l'aboutissement de cette thèse. Je voudrais lui exprimer ma reconnaissance pour les efforts inépuisables qu'il a fournis, pour sa patience et sa disponibilité à tout moment malgré ses occupations.

Je remercie sincèrement M. Yvan BEDARD, Professeur à l'université Laval, Quebec Canada, pour avoir accepté d'être rapporteur de ce mémoire, pour ses remarques pertinentes qui ont contribué à l'amélioration de la qualité de ce mémoire et pour l'honneur qu'il me fait en participant au jury.

Je remercie sincèrement M. Jérôme GENSEL, Professeur à l'Université Pierre Mendès France, Grenoble 2, pour avoir accepté d'être rapporteur de ce mémoire. Je le remercie également pour ses remarques pertinentes qui ont contribué à l'amélioration de la qualité de ce mémoire, ainsi que pour l'honneur qu'il me fait en participant au jury.

Je remercie sincèrement M. Michel SCHEINDER, Professeur à l'université Blaise Pascal, Clermont-Ferrand II, et M. Ronan TOURNIER, Maître de Conférences à l'Université Paul Sabatier, Toulouse 1, pour avoir accepté d'être examinateurs de cette thèse.

Je tiens à remercier également Michel SCHEINDER, Farouk TOUMANI, Jean-Pierre CHANET, Myoung-Ah KANG, Catherine ROUSSEY, André MIRALLES, et Frank RAVAT pour leurs participations à mes deux comités de thèse. Leurs conseils, remarques et critiques constructives m'ont beaucoup aidé.

Je tiens à remercier particulièrement M. Hadj MAHBOUBI qui m'a accueilli chez lui pendant mes premières semaines à Clermont-Ferrand. Je le remercie également pour les bons moments que nous avons partagés et pour sa précieuse aide durant les premiers mois de ma thèse.

Je tiens à remercier également mes deux stagiaires MM. Hassan NAZIH et El Mehdi LAHLOU pour leurs contributions dans la réalisation de partie implémentation de cette thèse.

Je remercie également les membres de l'équipe Pollutions agricoles diffuses de l'UR MALY de l'Irstea de Lyon et André Miralles de l'UMR TETIS de Montpellier pour leur collaboration dans le cadre des projets SIE Pesticides et Miriphyque à l'application des résultats de cette thèse dans le domaine agri-environnemental.

Mes remerciements s'adressent également à tous les membres de l'équipe COPAIN pour leur soutien, les réunions, les pauses café, et toutes les bonnes choses que nous avons partagées.

Je tiens à remercier en particulier Mme Eliane SIMON et Mme Colette CADIOU pour leur grande gentillesse et disponibilité.

Je tiens à remercier toutes les personnes qui ont contribué à l'aboutissement de cette thèse.

Je tiens à présenter toute ma reconnaissance à ma mère et mon père qui m'ont permis de continuer mes études avec tout l'amour, la tendresse, le soutien sans faille, la bonté et la fierté de cœurs parental et maternel.

Finalement, je tiens à remercier du fond du cœur mes sœurs Fadhila, Ouahiba et Dalila, mon frère Mehdi ainsi que tous les autres membres de ma famille.

Table des matières

Liste des figures

Liste des tableaux

PARTIE I : INTRODUCTION

1 Chapitre 1 : Introduction

1.1 Contexte de thèse

Les systèmes d'Entrepôts de Données (ED) et OLAP (*On-Line Analytical Processing*) sont des technologies très utilisées actuellement dans le domaine de l'informatique décisionnelle. Ces systèmes permettent l'analyse en ligne de très gros volumes de données alphanumériques issues de sources de données multiples et hétérogènes [Inmon, 2005]. Ils mettent à disposition des utilisateurs, un espace centralisé de stockage, d'intégration et d'historisation de toutes les données de l'organisation qui sont pertinentes à l'analyse décisionnelle (données décisionnelles), l'ED; et définissent des outils intuitifs pour l'exploration interactive et simplifiée de cet espace suivant une approche multidimensionnelle et à différents niveaux de granularité, les outils OLAP.
Ces systèmes reposent sur le modèle multidimensionnel qui permet une représentation multidimensionnelle de l'information décisionnelle sous la forme d'hypercubes de données qui est adaptée à la vision qu'ont les décideurs de cette information. Cette représentation est basée sur la dualité des concepts fait-dimension. Les faits représentent des sujets d'analyse et sont décrits par des mesures généralement numériques. Les dimensions définissent les critères d'analyse des faits et sont organisées en hiérarchies de plusieurs niveaux d'agrégation pour permettre la visualisation des mesures à différents niveaux de détail.
Les systèmes OLAP implémentent un ensemble d'opérateurs pour l'exploration en ligne des hypercubes de données qui se déclenchent suite à l'interaction des utilisateurs avec une interface spécifique. Ces opérateurs permettent d'avoir une information plus ou moins détaillée en agrégeant les mesures par des fonctions d'agrégation (par exemple Sum, Max, etc.). Les opérateurs habituels sont les opérateurs de forage vers le haut (Roll-up) et vers le bas (Drill-down) qui permettent respectivement de monter/descendre dans les hiérarchies de dimension, les opérateurs de coupe Slice et Dice qui sélectionnent une partie des données de l'hypercube; et l'operateur de projection qui permet de sélectionner une ou plusieurs mesures, etc.

La nécessité croissante d'intégrer l'information spatiale dans l'analyse multidimensionnelle OLAP a conduit à l'introduction des concepts d'Entrepôts de Données Spatiales (EDS) et du Spatial OLAP (SOLAP). Les Entrepôts de Données Spatiales (EDS) permettent d'intégrer et d'historiser de très gros volumes de données géoréférencées provenant de sources diverses pour supporter les processus de prise de décision [Stefanovic et al., 2000]. Les systèmes SOLAP définissent des moyens effectifs pour explorer le contenu de l'EDS [Bédard et al., 2007]. Ces systèmes enrichissent les capacités d'analyse des systèmes OLAP, en combinant des analyses multidimensionnelles avec des navigations et visualisations cartographiques. Ceci permet par exemple de comprendre la distribution géographique d'un phénomène et de comparer divers phénomènes à diverses échelles géographiques.

Les systèmes d'EDS et SOLAP se basent sur le modèle spatio-multidimensionnel qui étend le modèle multidimensionnel OLAP et permet de prendre en compte la composante spatiale de l'information géographique en la représentant en axes d'analyse (dimension spatiale) et/ou en mesures (mesure spatiale) [Malinowski et Zimányi, 2008]. En plus de la visualisation cartographique des résultats des requêtes décisionnelles, la représentation des données spatiales dans les niveaux d'agrégation permet principalement d'utiliser des prédicats spatiaux pour la sélection et le regroupement des données lors de l'application des opérateurs SOLAP comme par exemple le Slice spatial et le Roll-up spatial; la représentation des

données spatiales en mesures rend possible leur agrégation à l'aide de fonctions d'agrégation spatiale (e.g. union spatiale).

L'architecture typique d'un système SOLAP est constituée de quatre couches logicielles (cf. Figure 1.1) : ETL (*Extract, Transform and Load tool*) spatial, EDS, Serveur SOLAP et Client SOLAP [Rivest et al., 2003]. Le rôle de l'ETL spatial est l'intégration et le chargement périodiques au sein de l'EDS de toutes les données qui sont nécessaires à l'analyse SOLAP. La couche EDS historise et gère les données intégrées au moyen d'un SGBDR (Système de Gestion de Bases de Données Relationnelles) spatial (e.g. Oracle Spatial). Le serveur SOLAP (e.g. GeoMondrian) implémente les opérateurs SOLAP (e.g. Roll-up spatial) pour le calcul et la manipulation des hypercubes de données spatiales. Enfin, le client SOLAP (e.g. JRubik) met à disposition des utilisateurs une série d'interfaces graphiques intuitives qui déclenchent les opérateurs SOLAP et permettent l'exploration interactive des hypercubes de données spatiales et la visualisation des résultats de requêtes sous différents formats d'affichage interactifs : tableaux croisés dynamiques, diagrammes statistiques, cartes, etc.

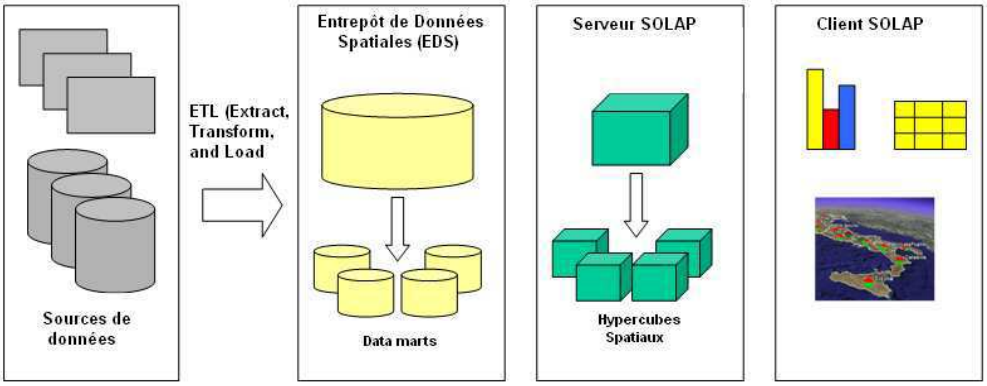


Figure 1.1. Architecture typique d'un système SOLAP.

1.2 Problématique

La qualité d'analyse (S)OLAP (i.e. l'exactitude des valeurs et la validité sémantique des indicateurs d'analyse) est une problématique de recherche très importante car ces systèmes sont utilisés pour supporter les processus de prise de décision au sein de divers types d'organisations [Rizzi et al., 2006; Salehi, 2009]. En effet, des résultats d'analyse incorrects peuvent mener à une mauvaise interprétation d'une situation qui à son tour peut mener à de mauvaises prises de décision. Les mauvaises décisions peuvent à leur tour engendrer des impacts économiques, sociétaux et environnementaux négatifs.

La qualité d'analyse (S)OLAP dépend en premier de la **qualité des données (spatiales) entreposées** qui est principalement affectée par trois types de problèmes : imprécision, non exhaustivité et incohérence logique [ISO/TC 211, 2002].
Dans cette thèse nous nous focalisons sur les problèmes d'incohérence logique. Cette notion, qui est référencée également sous le terme d'inconsistance, est définie dans le domaine des bases de données comme la présence de contradictions dans un jeu de données en rapport aux règles logiques présentes dans sa spécification [ISO/TC 211, 2002; Devillers et Jeansoulin, 2005]. Elle est typiquement contrôlée au moyen de Contraintes d'Intégrité (CI) [Duboisset, 2007; Pinet, 2010]. Les CI sont des assertions qui définissent les conditions qui doivent être respectées par les données et éventuellement les conséquences de leur violation [Goertzen et Stausberg, 2007].

Dans les systèmes d'ED(S), les problèmes d'incohérence logique (de données) se posent plus particulièrement lors de la phase d'intégration des sources de données [Salehi, 2009]. En effet, lors de cette phase importante, qui peut être très complexe à réaliser en raison des différents types d'hétérogénéité pouvant exister entre les sources de données notamment dans le cas d'applications SOLAP pour l'agriculture et l'environnement (e.g. réseaux de capteurs, systèmes de télédétection, bases de données d'enregistrement de pratiques, etc.), plusieurs problèmes d'incohérence peuvent émerger. Par exemple dans le cadre d'une application SOLAP agri-environnementale pour l'analyse du transfert de pesticides dans le sol (cf. Chapitre 8), les exemples d'incohérences peuvent être des géométries de parcelles qui se chevauchent, des géométries de parcelle qui ne sont pas dans leur domaine de définition, etc.

Un autre aspect déterminant de la qualité d'analyse SOLAP concerne **la qualité d'agrégation**, investiguée dans la littérature sous le terme Anglais de *Summarizability* que nous proposons de traduire par le terme Français **Agrégeabilité** [Lenz et Shoshani, 1997]. En effet, avoir des données entreposées de bonne qualité ne garantit pas des agrégats ou indicateurs d'analyse exacts et qui ont du sens pour l'application considérée. La qualité des indicateurs obtenus suite à l'agrégation des données, en plus de la qualité des données, dépend aussi de (i) conditions structurelles qui sont relatives aux types de hiérarchies de dimension et de relations fait-dimension[1] (par exemple les hiérarchies de dimension non strictes peuvent engendrer le comptage en double des valeurs de mesure) mais également de (ii) conditions sémantiques qui concernent essentiellement la compatibilité entre les natures des trois éléments fondamentaux d'une agrégation (S)OLAP qui sont la fonction d'agrégation, la mesure et la dimension; par exemple, l'agrégation de mesures semi-additives comme la quantité en stock des produits selon le temps par la fonction Sum est incorrecte, l'agrégation des valeurs de température (mesure non additive) en utilisant la fonction Sum n'a pas de sens dans certaines applications.

Toutefois, la qualité de l'analyse (S)OLAP ne peut se limiter aux contrôles de ces deux types de qualité que nous venons de décrire (qualité de données et qualité d'agrégation). Il faut également contrôler la qualité des requêtes (S)OLAP lors de l'exploration des données (agrégées), **qualité d'exploration,** pour éviter la mauvaise interprétation des résultats. En effet, comme les utilisateurs de ces systèmes (décideurs) sont supposés ignorer les contraintes sur les données multidimensionnelles [Inmon, 2005], ils peuvent formuler assez facilement des requêtes insensées en définissant des combinaisons inconsistantes de membres de dimension et d'indicateurs [Devillers et al., 2007; Boulil et al., 2012d]. Ces combinaisons invalides dépendent de l'application considérée. Dans une application d'analyse de ventes, un exemple de requête invalide peut être "Quel est le total des ventes en Allemagne de l'Est en 2000?". Bien que l'exécution de cette requête retourne un résultat nul car l'Allemagne de l'Est n'existe plus à partir de 1990 - en explicitant par exemple des CI de données qui interdisent le stockage des instances de fait définies par ces combinaisons, un problème d'interprétation de ce résultat se pose : l'utilisateur va probablement interpréter ce résultat nul comme absence de ventes dans les magasins de l'Allemagne de l'Est en 2000; au lieu de réaliser que sa requête est impossible. Cette interprétation erronée peut le mener à prendre de mauvaises décisions.

Dans la littérature, la **cohérence logique** a été investiguée par différents travaux de recherche dans le domaine des ED et EDS [Carpani et Ruggia, 2001; Ghozzi et al., 2003a; Malinowski et Zimányi, 2008; Pinet et Schneider, 2009; Salehi, 2009; Pinet et Schneider, 2010]. Ces travaux proposent la définition de CI pour interdire l'entreposage de données erronées.

[1] Les relations fait-dimension sont des relations qui lient les faits aux dimensions.

D'autres approches ont été proposées pour traiter les problèmes de qualité de données lors de l'exploration par exemple par la reformulation des requêtes [Pedersen et al., 2001].
La qualité d'agrégation a été largement investiguée notamment dans le domaine des ED**.** Les travaux existants proposent différentes approches : (i) la transformation des structures multidimensionnelles irrégulières (i.e. les hiérarchies de dimension et relations fait-dimension qui ne satisfont pas les conditions structurelles d'agrégeabilité introduites par [Lenz et Shoshani, 1997]) en structures agrégeables via des algorithmes [Pedersen et al., 1999; Pedersen et Tryfona, 2001; Jensen et al., 2004; Mansmann et Scholl, 2006; Mazón et al., 2006; Mazón et al., 2008], par exemple, la conversion de hiérarchies non strictes en hiérarchies strictes; (ii) la définition de CI pour contrôler la compatibilité sémantique entre la fonction d'agrégation, la mesure et la dimension [Bimonte et al., 2009; Salehi, 2009; Prat et al., 2010], etc.

A notre connaissance, la qualité d'exploration à été très peu étudiée [Levesque et al., 2007]. Ce travail ne propose ni langages pour la spécification des contraintes d'exploration, ni d'implémentation.

De manière générale, ces travaux sur la qualité (S)OLAP présentent les limites suivantes :

(1) Le manque de Framework qui traite les trois de types de qualité à la fois. A notre connaissance aucun travail ne considère les trois types de qualité qui affectent la qualité d'analyse SOLAP et qui sont les qualités de données, d'agrégation et d'exploration.

(2) Le manque d'exhaustivité des classifications de CI proposées. Les classifications de CI rendent plus facile leur identification et définition. Dans ce contexte, quelques classifications ont été proposées dans le domaine des ED et EDS [Carpani et Ruggia, 2001; Ghozzi et al., 2003a; Salehi, 2009; Prat et al., 2010]. A notre connaissance, [Salehi, 2009] est le travail qui propose la classification la plus complète et la plus détaillée. Il s'agit d'une contribution majeure dans le domaine, qui par contre peut être étendue par d'autres types de CI comme par exemple les CI de données sur plusieurs hypercubes, les CI d'exploration et les CI de métadonnées.

(3) Le manque d'un modèle spatio-multidimensionnel qui fournit un support complet pour la définition de CI concernant ces trois types de qualité. Comme nous allons le voir dans la suite de ce mémoire, un nombre important de modèles multidimensionnels conceptuels ont été proposés dans la littérature, chacun traitant un aspect particulier de la modélisation multidimensionnelle, mais aucun modèle ne présente encore tous les critères nécessaires à l'expression des différentes classes de CI qui impactent la qualité d'analyse (S)OLAP. Un nouveau modèle multidimensionnel est par conséquent une première phase pour le traitement de la qualité d'analyse (S)OLAP.

(4) Le manque d'utilisation de langages de spécification standards. L'utilisation de langages standards pour la modélisation conceptuelle des CI et du modèle multidimensionnel rend plus facile leur compréhension et leur implémentation. L'avantage des modèles basés sur des standards, comme UML et ER, est qu'ils minimisent les temps et efforts d'apprentissage et de compréhension des spécifications car les concepteurs et utilisateurs ont l'habitude de travailler avec ces langages [Torlone, 2003]. UML (*Unified Modeling Language*) en particulier est très connu pour sa grande capacité à modéliser des aspects statiques et dynamiques complexes comme ceux liés à la qualité dans les systèmes (S)OLAP; ceci est en partie du à sa facilité d'extension et à son interopérabilité avec d'autres standards notamment le langage de contraintes objet OCL (*Object Constraint Language*) [Abelló et al., 2006; Pinet et Schneider, 2010]. Ce

standard OMG pour l'expression des contraintes et de requêtes sur des diagrammes UML, constitue une solution efficace pour spécifier les CI au niveau conceptuel, de manière claire, non ambiguë et indépendante des plateformes d'implantation cibles. UML et OCL sont aussi les langages de modélisation les plus intégrés dans les Ateliers de Génie Logiciel (AGL) et les règles de leurs mappings vers différents SGBD sont bien définies, ceci permet la génération automatique des modèles physiques pour l'implémentation du modèle de données et des CI de données, ce qui optimise davantage les temps de développement. Dans ce contexte, seuls les auteurs dans [Pinet et Schneider, 2010] utilisent OCL mais pour n'exprimer que quelques classes de CI de données. La plupart des autres travaux expriment les contraintes en utilisant des langages non standards de différents types (logiques, naturels, visuels ou hybrides) qui sont difficiles à implémenter (cf. Sections 4.3.1, 4.4, 4.5 et 4.6).

(5) Le manque d'implémentations automatiques des CI. Quelques travaux proposent des implémentations manuelles pour quelques types de CI (e.g. programmes Java pour vérifier certaines conditions d'agrégeabilité dans les hiérarchies de dimension [Hurtado et al., 2005]), mais aucun travail ne propose une implémentation automatique de CI basée sur des générateurs de code (cf. Section 4.7) .

1.3 Objectifs et méthodologie de recherche

L'objectif principal de cette thèse est de proposer un cadre de conception et d'implémentation automatique pour les CI SOLAP. Ce cadre doit satisfaire les exigences suivantes :

(1) Il doit prendre en compte l'incohérence logique au niveau :
- des données,
- des agrégations,
- et des requêtes (S)OLAP.

(2) Il doit permettre une identification et définition facile des classes de problèmes d'incohérence en proposant une classification des CI SOLAP.

(3) Il doit permettre une modélisation conceptuelle basée sur les standards UML et OCL des CI SOLAP.

(4) Il doit permettre une traduction automatique des spécifications conceptuelles des CI vers des implémentations dans différents niveaux de l'architecture SOLAP.

En fin ce cadre doit être validé par une ou plusieurs applications dans le domaine agri-environnemental.

Pour réaliser cette thèse, nous avons suivi une démarche de recherche classique, constituée des étapes suivantes :

Etape 1 : Revue de littérature et formulation du problème de recherche

Cette étape a débutée par une revue des concepts fondamentaux des systèmes SOLAP et OLAP en nous focalisant sur les systèmes relationnels. Ceci a été fait dans le but de nous familiariser et d'appréhender les concepts qui étaient nécessaires pour définir et répondre à notre problématique de recherche. Le choix du système relationnel est principalement motivé par l'existence de consensus pour ces systèmes notamment concernant leur modélisation logique (i.e. adoption du schéma en étoile et ses variantes par la plupart des travaux de recherche). Cette sous étape de revue des concepts vise principalement l'étude des aspects suivants : les couches logicielles composant ces systèmes, comment ces systèmes sont modélisés au niveau conceptuel et comment ils sont implémentés. Après ça, nous avons réalisé une étude sur la qualité SOLAP dont l'objectif est de déterminer ses principales composantes et quelles approches sont proposées pour les traiter. Suite à cela, nous avons recadré la problématique de thèse en nous intéressant uniquement à la cohérence logique.

Ensuite, une étude approfondie sur la définition des CI (qui est une approche courante pour vérifier la cohérence logique) dans les systèmes SOLAP et OLAP. Ceci était réalisé dans le but de bien identifier les classes de CI proposées par les travaux existants, ainsi que de bien comprendre quels sont les langages utilisés pour modéliser les CI et comment les implémenter. Suite à cette étude, nous avons pu identifier les 5 limites présentées précédemment en Section 1.2.

Etape 2 : Définition d'une nouvelle classification des CI SOLAP
L'étude faite à l'étape 1, nous a permis d'identifier de nouvelles classes de CI (CI de métadonnées, CI d'agrégation de schéma, CI de données inter-hypercube). Nous avons également senti le besoin de réorganiser/décliner certaines classes déjà identifiées pour avoir une classification qui tienne mieux en compte des éléments principaux d'un SOLAP que sont les données, l'agrégation et les requêtes.

Etape 3 : Etat de l'art sur les modèles conceptuels
Pour permettre une expression conceptuelle aisée des classes de CI proposées à l'étape 2, nous avons vu qu'il fallait avoir un modèle spatio-multidimensionnel répondant à un ensemble d'exigences que nous avons défini. Nous avons donc réalisé un état de l'art et une évaluation (par rapport à l'ensemble des exigences définies) des principaux modèles multidimensionnels et spatio-multidimensionnels existants. Cette évaluation a montré qu'aucun modèle ne satisfait l'ensemble de nos exigences (cf. Chapitre 6).

Etape 4 : Définition du profil UML SOLAP
L'évaluation des travaux existants, réalisée à l'étape 3, a montré que les modèles existants ne satisfaisaient pas les exigences nécessaires à une expression aisée des CI SOLAP. Nous avons donc décidé de proposer un nouveau profil UML. Le choix d'UML est principalement motivé par la capacité de ce langage à modéliser des aspects complexes et par son support d'OCL.

Etape 5 : Définition des CI SOLAP
Une fois le profil défini, nous nous sommes intéressés à la définition des CI SOLAP. Pour la modélisation conceptuelle des CI de données, nous avons choisi le langage Spatial OCL, essentiellement pour répondre aux objectifs (3) et (4) précédents. L'implémentation automatique de ces CI est réalisée en utilisant le générateur de code Spatial OCL2SQL qui produit uniquement du code SQL pour Oracle Spatial, d'où le choix de cet SGBD dans notre architecture d'implémentation. Comme Spatial OCL ne permet pas d'exprimer les CI d'agrégation au niveau modèle, nous avons décidé de les définir au niveau métamodèle sous forme de contraintes OCL attachées aux stéréotypes du profil dont les caractéristiques le permettaient. Pour la modélisation des CI de requêtes, nous avons étendu notre profil par des éléments qui permettent de les exprimer au niveau conceptuel. Leur implémentation automatique au niveau serveur OLAP Mondrian est réalisée en utilisant un générateur de code que nous avons développé en Java. Le choix de Mondrian est essentiellement motivé par la réutilisation des compétences déjà acquises dans l'équipe.

Etape 6 : Expérimentation
Dans cette étape, nous avons appliqué le cadre de définition des CI SOLAP que nous avons proposé dans le contexte de projets nationaux agri-environnementaux afin de les valider. Concrètement, cette étape se faisait en parallèle des étapes 2, 3, 4 et 5. Grâce aux retours que nous avons eu des applications agri-environnementales, nous a pu réajuster et modifier nos propositions.

1.4 Contributions

Notre première contribution dans cette thèse est d**'étendre la notion de CI dans le cadre des systèmes d'ED(S) et (S)OLAP afin de prendre en compte ces trois types d'incohérences qui impactent la qualité de l'analyse S(OLAP) (cf. Section 5); voir aussi [Boulil et al., 2012d]**. Nous utilisons les CI pour effectuer trois types de contrôle de qualité (Figure 1.2) :

(A) Un contrôle de qualité de données pour garantir que les données (y compris spatiales) entreposées sont valides et cohérentes entre elles (par exemple, les géométries des villes doivent être topologiquement incluses dans les géométries de leurs pays).

(B) Un contrôle de qualité d'agrégation pour veiller à ce que l'agrégation des mesures soit correctement faite et ait bien un sens par rapport à l'application considérée. Par exemple, dans le cas d'une application d'analyse des ventes, la somme des prix unitaires des produits n'a pas de sens; la somme des quantités en stock des produits selon le temps est incorrecte puisque cette agrégation engendre un comptage en double des valeurs de cette mesure; etc.

(C) Un contrôle de qualité d'exploration (S)OLAP pour éviter les mauvaises interprétations des résultats de requêtes (S)OLAP insensées ou insatisfiables (i.e. pour lesquelles il ne peut exister de résultats dans le contexte applicatif considéré). Par exemple, la requête "quels sont les montants de ventes en URSS après le 26 Décembre 1991?" ne peut pas retourner de résultats puisque l'URSS a été dissoute à cette date.

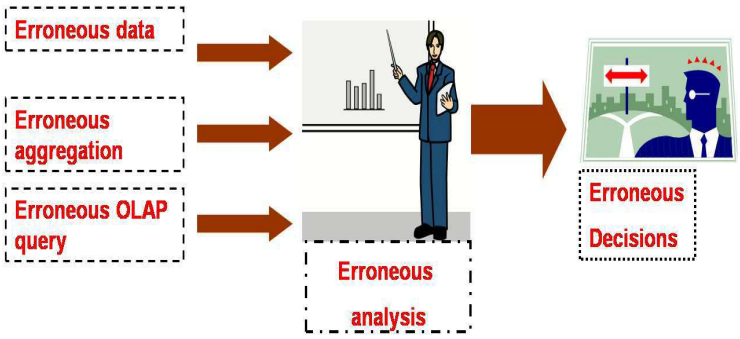


Figure 1.2. Qualité dans le processus décisionnel (S)OLAP.

Notre deuxième contribution concerne la proposition d'une classification des CI (S)OLAP (cf. Sections 5.3 et 7.5.2), voir aussi [Boulil et al., 2011]. Cette classification étend la classification proposée dans [Salehi, 2009] par l'introduction d'autres types de CI comme par exemple les CI de requêtes, les CI de métadonnées et les CI de données impliquant plusieurs hypercubes. Cette classification sert de cadre à l'identification des CI SOLAP par les concepteurs et permet une définition plus facilitée des CI. Elle définit quatre grandes familles de CI :

(A) Les CI de métadonnées qui vérifient la validité et la cohérence (compatibilité) des métadonnées des différentes sources de données intégrées;

(B) Les CI de données vérifient la validité et la cohérence des données entreposées;

(C) Les CI d'agrégation garantissent que les agrégations des mesures le long des hiérarchies de dimension sont correctes et ont du sens dans le contexte applicatif considéré;

(D) Les CI d'exploration vérifient que les combinaisons des membres de dimension et de mesures définissant les requêtes SOLAP sont consistantes (i.e. valides et satisfiables).

Ces quatre catégories sont de plus divisées en différentes sous catégories (cf. Sections 5.3 et 7.5.2). En fonction de leur catégorie, les CI peuvent impliquées différents types d'éléments et concepts (spatio)-multidimensionnels (hypercube, fait, dimension, mesure, hiérarchie de dimension, etc.).

Notre troisième contribution est la proposition d'un cadre basé sur les langages standards UML et OCL pour la spécification conceptuelle des modèles multidimensionnels et des CI (S)OLAP (cf. Section 5.4). Le choix de ces langages est principalement justifié par leur capacité à modéliser des aspects très complexes et leur facilité d'implémentation vue leur forte intégration dans les outils de modélisation existants. Ce cadre se base sur un profil UML (cf. Section 6.7) qui définit trois métamodèles interconnectés pour spécifier :

(A) les structures du modèle multidimensionnel de données et les contraintes sur les données (**métamodèle de l'EDS**),

(B) comment les mesures doivent être agrégées le long des hiérarchies de dimension pour répondre aux besoins d'analyse des utilisateurs (en d'autres termes les spécifications des indicateurs d'analyse) et les CI d'agrégation (**métamodèle d'agrégation**),

(C) les CI sur les requêtes SOLAP (**métamodèle de CI de requêtes**).

Dans notre approche, les CI sur les données sont définies au niveau modèle par les concepteurs. Elles sont exprimées sur le modèle de données en utilisant Spatial OCL. Spatial OCL est une extension spatiale d'OCL permettant l'expression des relations spatiales topologiques du modèle des 9 intersections en géométries simples et complexes [Duboisset, 2007]. Les contraintes d'agrégation sont spécifiées au niveau métamodèle (i.e. dans la définition du profil UML). Elles sont exprimées en OCL sur les stéréotypes du métamodèle d'agrégation. Les contraintes de requêtes sont définies au niveau modèle par les concepteurs. Elles sont exprimées en utilisant les stéréotypes du métamodèle de CI de requêtes et le langage Spatial OCL.

Les principales caractéristiques de notre profil, en plus de son implémentation, qui lui confère un avantage indéniable par rapport aux modèles (spatio)-multidimensionnels conceptuels existants qui sont de notre point de vue insuffisants pour la définition de tous les aspects de la qualité (S)OLAP concernent :

(i) la représentation du modèle d'agrégation des mesures grâce à de nouveaux concepts comme par exemple l'indicateur d'analyse qui permet de définir une ou plusieurs façons pour agréger une mesure le long des hiérarchies de dimension en exploitant différents types de règles d'agrégation et de fonctions d'agrégation;

(ii) la représentation explicite et le typage des éléments multidimensionnels ce qui permet d'exprimer par exemple les CI d'agrégation qui s'appliquent à tous les ED.

(iii) par rapport aux modèles et profils UML d'ED et d'EDS existants, ce profil représente explicitement d'autres concepts multidimensionnels tels que les hiérarchies de dimension et les fonctions d'agrégation, ce qui permet d'exprimer plus de CI comme par exemple les CI de requêtes définissant des combinaisons invalides de hiérarchies de dimension.

(iv) En tant qu'extension d'UML, ce profil nous permet d'exprimer des CI avec OCL et peut être facilement étendu pour traiter d'autres aspects de la qualité comme par exemple les CI de métadonnées.

Notre quatrième contribution concerne l'implémentation automatique des CI et des modèles d'EDS et d'agrégation (cf. Sections 5.5, 6.9 et 7.7)**;** voir aussi [Boulil et al., 2010b; Boulil et al., 2012d]. L'idée principale du cadre d'implémentation proposé est d'offrir un système qui à partir de la représentation conceptuelle du modèle (spatio)-multidimensionnel et des CI génère automatiquement des implémentations dans différentes couches de l'architecture SOLAP en exploitant des générateurs de code. Similairement à la modélisation conceptuelle, les langages utilisés à ce niveau (implémentation) sont typiquement des standards Spatial SQL, MDX[2] (*MultiDimensional Expressions language*) et OCL.

[2] MDX (Multidimensional Expressions language) est le langage de requête standard pour les bases de données multidimensionnelles et OLAP (cf. Sections 3.3.5.3 et 5.5).

Selon son type, chaque CI conceptuelle est automatiquement traduite dans le langage d'implémentation supporté par la couche de l'architecture SOLAP dans laquelle elle peut être implémentée.
Les CI de données sont implémentées sous forme de requêtes, triggers et vues SQL au niveau de la couche EDS (i.e. SGBD Spatial) en utilisant une extension du générateur de code Spatial OCL2SQL. Cet outil nous permet également de générer automatiquement le schéma physique de l'EDS (i.e. scripts SQL pour créer la base de données). Pour intégrer cet outil qui est à la base proposé pour les bases de données spatiales [Duboisset, 2007] dans une vraie approche de développement d'ED, nous l'avons adapté pour qu'il prenne en compte les différentes variantes de la modélisation multidimensionnelle logique notamment les schémas en étoile et en flocons.
Les CI d'agrégation sémantiques sont implémentées dans le profil UML sous forme de contraintes OCL attachées aux stéréotypes du métamodèle d'agrégation; elles sont contrôlées par l'AGL MagicDraw[3] lorsque le concepteur valide le modèle conceptuel. Le contrôle des CI au niveau d'abstraction conceptuel permet d'éviter la génération d'implémentations pour des modèles conceptuels incorrects, dans ce cas pour des modèles d'agrégation incorrects. Notons que cette approche nouvelle et intéressante (car indépendante du type d'architecture OLAP (ROLAP, MOLAP et HOLAP - cf. Section 2.) et des langages d'implémentation) pour le traitement de ce type de CI est rendue possible par le bon typage des éléments de modélisation qu'offre notre profil UML et plus particulièrement sa partie métamodèle d'agrégation.
Les CI de requêtes sont implémentées au niveau de la couche Serveur SOLAP sous forme d'expressions MDX en utilisant un générateur de code que nous avons développé (UML2MDX). En plus des CI de requêtes, UML2MDX, de façon similaire à Spatial OCL2SQL, permet de générer le schéma d'analyse OLAP Mondrian[4] (fichier XML qui définit les hypercubes, les hiérarchies de dimension et les indicateurs d'analyse) et des implémentations MDX pour les indicateurs d'analyse complexes qui ne sont pas considérées par les outils existants.
L'architecture SOLAP exemple que nous avons considérée pour la validation de ces solutions est constituée du SGBD Oracle Spatial 11g, du serveur OLAP Mondrian et du client SOLAP JRubik (cf. Section 3.3.5). Bien que cette architecture n'est pas entièrement spatiale (car non basée sur un serveur SOLAP), elle est suffisante pour l'implémentation de toutes les CI considérées (même spatiales) (cf. Chapitre 10).

Enfin, ces différentes propositions ont été appliquées à des entrepôts de données agri-environnementales dans le cadre de projets nationaux :
(A) Projets SIE Pesticides (financé par Irstea) et Miriphyque (financé par le Ministère de l'écologie) : développement d'une application d'ED pour le stockage et l'analyse des résultats de simulations de transfert de pesticides;
(B) Projet EnergeTIC (financé par le Ministère de l'agriculture) : développement d'une application d'EDS pour l'analyse SOLAP des dépendances énergétiques des exploitations agricoles.

1.5 Organisation du mémoire

La suite de ce mémoire s'organise en cinq parties, comme suit :

[3] MagicDraw est un outil de modélisation basé sur UML dans lequel nous avons choisi d'implémenter notre profil UML en raison du support intéressant qu'il offre aux contraintes OCL (cf. Sections 5.5 et 6.7).
[4] Mondrian est un serveur OLAP relationnel open source supportant le langage MDX et la connexion aux principaux SGBD existants (cf. Section 3.3.5.3).

Partie Etat de l'art. Cette partie présente principalement les concepts liés à la modélisation conceptuelle des systèmes OLAP et SOLAP, à leurs architectures et la qualité dans ces systèmes. Elle réalise trois états de l'art (les modèles multidimensionnels conceptuels, les modèles spatio-multidimensionnels conceptuels et les approches pour le traitement de la qualité dans ces systèmes) et comprend trois chapitres :

Chapitre 2. Ce chapitre introduit tout d'abord les principaux concepts des systèmes OLAP; présente ensuite leur architecture typique et le modèle multidimensionnel sur lequel ils se basent en détaillant certains aspects qui peuvent affecter la qualité d'analyse OLAP; réalise ensuite un état de l'art sur les principaux modèles conceptuels proposés dans la littérature en se focalisant sur les modèles UML; et enfin décrit brièvement des aspects concernant les modélisations logique et physique des systèmes OLAP Relationnels (ROLAP).

Chapitre 3. Dans ce chapitre, en premier les principaux concepts spécifiques aux systèmes SOLAP sont présentés; s'ensuit une description détaillée du modèle spatio-multidimensionnel sur lequel se base ces systèmes; puis la présentation de l'architecture typique d'un SOLAP pour comprendre le rôle de chacune de ses couches dans le traitement des problèmes de qualité SOLAP; enfin un état de l'art détaillant les modèles spatio-multidimensionnels conceptuels proposés dans la littérature avec un focus sur les modèles basés sur UML et ER.

Chapitre 4. Ce chapitre réalise une étude sur la qualité dans les systèmes OLAP et SOLAP : en premier les principales composantes de cette qualité sont décrites (qualités de données, d'agrégation et d'exploration); s'ensuit une présentation des CI et des différents types de langages utilisés pour leur spécification dans les bases de données spatiales et les ED(S) avec un focus sur les langages de contraintes objet OCL et Spatial OCL sur lesquels sont basées nos propositions; et enfin un état de l'art détaillant les méthodes proposées dans la littérature pour le traitement de ces trois types de qualité ainsi qu'une évaluation détaillée par rapport un ensemble de critères des propositions existantes.

Partie Contributions. Cette partie s'articule autour de trois chapitres issus de trois articles qui présentent nos contributions. Ces chapitres reprennent de manière plus succincte les notions et les états de l'art présentés de manière plus exhaustive et en détails dans la partie Etat de l'art. Les chapitres 5 et 7 reprennent brièvement l'état de l'art sur les méthodes de traitement de la qualité (S)OLAP présenté au Chapitre 4. Le chapitre 6 reprend les descriptions détaillées des modèles conceptuels classiques et spatiaux qui sont présentées dans les chapitres 2 et 3; ce chapitre fournit en plus leur évaluation par rapport à des critères existants et nouveaux.

Chapitre 5. Ce chapitre présente une vue globale du Framework que nous proposons pour traiter les problèmes de qualité d'analyse (S)OLAP.
Dans ce chapitre, nous étendons tout d'abord la définition des CI pour prendre en compte les trois principaux types de problèmes impactant la qualité d'analyse (S)OLAP; puis présentons l'architecture générale du Framework que nous proposons pour leur traitement. Ce Framework est constitué (i) d'une classification des CI SOLAP, (ii) d'un cadre conceptuel basé sur un profil UML et les langages de contraintes standards (OCL et Spatial OCL) permettant la spécification de ces problèmes de qualité sous forme de CI au niveau conceptuel d'une manière indépendante des plateformes d'implémentation et (iii) d'un cadre d'implémentation basé sur des générateurs de code permettant la génération automatique de mécanismes pour la vérification des CI dans différentes couches de l'architecture (S)OLAP. Ces propositions illustrées par quelques exemples de CI pour chaque type de qualité ont été validées au sein de l'architecture SOLAP choisie (cf. Section 1.3). Notons que ce chapitre étend la classification de CI proposée dans le chapitre 7 par l'introduction des CI

d'exploration; par conséquent les classes de CI introduites dans le chapitre 7 sont brièvement décrites. Le lecteur est donc invité à consulter le chapitre 7 pour plus de détails et d'exemples sur ces classes de CI.
Ce chapitre est issu d'une version améliorée de l'article. K. Boulil, S. Bimonte, F. Pinet (2012). ***A UML & Spatial OCL based approach for handling quality issues in SOLAP systems****. Publié dans : Proceedings of the 14th International Conference on Enterprise Information Systems (ICEIS 2012), Wroclaw, Poland, June 28 – July 1, 2012, 6 pages.*

Chapitre 6. Ce chapitre détaille plus spécifiquement notre profil UML sur lequel se basent nos propositions pour le traitement de la qualité (S)OLAP et la définition des CI d'agrégation. Le profil UML définit un ensemble de stéréotypes pour la représentation conceptuelle des systèmes (S)OLAP et de leurs différentes classes de CI. Il est implémenté dans l'AGL MagicDraw. En plus des critères de modélisation multidimensionnelle proposés dans la littérature, notre profil satisfait de nouveaux critères que nous avons identifié et qui sont essentiellement nécessaires à la définition du schéma d'analyse (S)OLAP et des CI comme par exemple la représentation des indicateurs d'analyse et règles d'agrégation complexes et le bon typage des éléments spatio-multidimensionnels. Nous proposons également dans ce chapitre une approche intéressante pour le traitement de la qualité d'agrégation : les CI d'agrégation sémantiques identifiées par la littérature et celles proposées dans ce chapitre sont définies en OCL dans le profil pour être vérifier au niveau d'abstraction conceptuel dans l'AGL. Afin de valider ces propositions, nous proposons un générateur de code (UML2MDX) qui permet la génération automatique du schéma OLAP Mondrian avec en particulier une implémentation des indicateurs d'analyse complexes sous forme d'expressions MDX.
Ce chapitre est issu d'une version améliorée de l'article. K. Boulil, S. Bimonte, F. Pinet, J.-P. Chanet (2011). ***A UML profile and OCL-based constraints for spatial data cubes****. Soumis à : Information Systems journal, Elsevier, August 11, 2011, 45 pages.*

Chapitre 7. Ce chapitre se focalise plus spécifiquement sur la définition des CI de données en fournissant plusieurs exemples. En premier la description détaillée et illustrée par des exemples d'une première version de la classification des CI SOLAP que nous proposons est présentée; ensuite la modélisation conceptuelle avec le langage Spatial OCL des CI de données ainsi que leur implémentation automatique au niveau de la couche EDS de l'architecture SOLAP au moyen du générateur de code Spatial OCL2SQL sont décrites.
Ce chapitre est issu d'une version améliorée de l'article. K. Boulil, S. Bimonte, H. Mahboubi, F. Pinet (2010). ***Towards the definition of spatial data warehouses integrity constraints with Spatial OCL.*** *Publié dans : Proceedings of the ACM 13th international workshop on Data warehousing and OLAP (DOLAP 2010), Toronto, Canada, October 26, 2010, pp. 31-36.*

Partie Expérimentations. Cette partie comprend deux chapitres issus de deux articles qui présentent l'application de nos propositions pour le traitement de la qualité (S)OLAP et la conception des ED(S) dans le cadre de projets nationaux.

Chapitre 8. Ce chapitre présente l'application de nos propositions dans le cadre les projets SIE Pesticides et Miriphyque pour la conception d'un système OLAP pour le stockage et l'analyse multidimensionnelle des résultats issus de simulations de transfert de pesticides.
Ce chapitre est issu d'une version améliorée de l'article. K. Boulil, F. Pinet, S. Bimonte, N. Carluer, C. Lauvernet, B. Cheviron, A. Miralles, J.-P. Chanet (2012). ***Using Data Warehouses for Storage and Visualization of Simulation Model Results.*** Soumis à : Environmental Modelling & Software journal, Elsevier, March 30, 2012, 20 pages.

Chapitre 9. Ce chapitre présente l'application de nos propositions dans le cadre du projet EnergeTIC pour la conception d'un système SOLAP pour l'analyse spatio-multidimensionnelle de données sur les dépendances énergétiques des exploitations agricoles.
Ce chapitre est issu d'une version améliorée de l'article. S. Bimonte, K. Boulil, M. Pradel, J.-P. Chanet (2012). ***Definition and analysis of new agricultural farm energetic indicators using spatial OLAP.*** Publié dans : Proceedings of the 17th international conference on geographical analysis, urban modeling, spatial statistics (GEOG-AN-MOD 2012), Salvador de Bahia, Brazil, June 18-20, 2012, 30 pages.

A noter que je suis le premier rédacteur de tous les articles et le contributeur de tous les travaux qui y sont présentés, à l'exception de l'article présenté au chapitre 9, dans lequel je suis le premier contributeur et rédacteur de la section 9.6; et contributeur à la section 9.5. Mon intervention sur le projet EnergeTIC concerne en effet l'utilisation de mon profil pour la conception du modèle spatio-multidimensionnel (cf. Section 9.5) et la spécification et l'implémentation de CI pour contrôler la qualité des données collectées, l'agrégation et les requêtes SOLAP (cf. Section 9.6).

Partie Conclusions et Perspectives. Cette partie dresse un bilan de nos travaux et présente les limites et les perspectives de recherche**.**

Partie Annexes. Cette partie comporte deux annexes :
Annexe A. Cette annexe fournit un tutorial sur les langages OCL et Spatial OCL.
Annexe B. Cette annexe présente les principales contraintes définies en OCL dans le profil pour éviter sa mauvaise utilisation par les concepteurs.

PARTIE II : ETAT DE L'ART

2 Chapitre 2 : Entrepôts de Données et OLAP

2.1 Introduction

Face à la concurrence grandissante et la multiplication des sources d'information dues à différents phénomènes économiques et technologiques (mondialisation, forte numérisation, etc.), les organisations dans tous les secteurs, ont un besoin important en outils d'analyse de données performants qui leur permettent d'avoir une meilleure connaissance de leur activité et de l'environnement dans lequel elles évoluent. Ces outils doivent être capables de collecter, stocker et analyser les grosses masses de données qui sont générées quotidiennement par différentes moyens et technologies [Golfarelli et Rizzi, 2009]. Ils doivent être en mesure de fournir rapidement une vision claire de l'activité de l'organisation et de son environnement, afin qu'elle puisse réguler son fonctionnement et réagir à temps aux différents stimuli et opportunités du marché.

Les systèmes d'information classiques ne satisfont pas les exigences d'une analyse de données efficace car ils sont basés sur un modèle de données non adapté à des requêtes d'analyse décisionnelle [Kimball et Ross, 2002; Malinowski et Zimányi, 2008]. C'est pourquoi, les Systèmes d'Aide à la Décision (SAD) ont été proposés dans le but d'aider les utilisateurs dans leur prise de décision. Ces systèmes d'information informatiques offrent un ensemble d'outils de collecte, de stockage et d'analyse de données permettant aux décideurs d'avoir une vision rapide et claire de l'activité [Golfarelli et Rizzi, 2009].

Parmi les différents types de SAD proposés (systèmes de fouille de données (*data mining*), tableaux de bord, etc.), les systèmes d'entrepôts de données et OLAP ont suscité, dés leur apparition aux débuts des 1990, un grand intérêt de la part des deux communautés : recherche et industrielle [Inmon, 2005]. Ces systèmes mettent à disposition des décideurs, un espace de stockage, d'intégration et d'historisation de toutes les données de l'organisation qui sont pertinentes à l'analyse décisionnelle, *entrepôt de données*; et définissent des outils intuitifs pour une exploration simple et interactive de cet espace, *outils OLAP* [Codd et al., 1993; Inmon, 2005; Golfarelli et Rizzi, 2009].

La suite de ce chapitre est organisée comme suit. La section 2.2 introduit les concepts principaux des systèmes d'ED et OLAP en explicitant leurs différences avec les systèmes OLTP. La section 2.3 présente les principaux concepts du modèle multidimensionnel sur lequel se basent ces systèmes avec une attention particulière aux facteurs qui influencent l'agrégation correcte des mesures (cf. Section 2.3.4). La section 2.4 présente l'architecture typique OLAP et ses variantes d'implémentation. Finalement, les principaux travaux existants sur la modélisation multidimensionnelle conceptuelle des systèmes d'ED et OLAP sont détaillés en Section 2.5.1; leurs représentations logique et physique sont présentées en Section 2.5.2.

2.2 Concepts principaux

La vocation originelle des premiers Systèmes d'Information, aussi dits OLTP (*On-Line Transaction Processing*), était le support des activités opérationnelles quotidiennes des organisations. Ces systèmes mélangeaient les traitements transactionnels relatifs à la gestion quotidienne de l'activité avec les traitements analytiques relatifs à l'évaluation du fonctionnement de cette activité. Cette approche présentait deux inconvénients majeurs [Inmon, 2005]. D'une part, les systèmes OLTP sont alourdis (par exemple en temps de réponse et de mise à jour) par les traitements analytiques et leurs données; et d'autre part

l'information décisionnelle n'était pas facilement accessible (par exemple en termes de facilité de formulation de requêtes et de temps de réponse) aux décideurs en raison du modèle de données normalisé sur lequel se basent ces systèmes.

Les systèmes d'entrepôts de données et OLAP ont été introduits dans le but de séparer les traitements analytiques des traitements opérationnels. Cela permet la suppression des données historiques et des traitements analytiques des systèmes OLTP et leur transfert vers un environnement dédié à l'analyse décisionnelle [Inmon, 2005]. Ceci permet d'avoir des systèmes OLTP moins complexes et plus performants car plus réduits, et des systèmes décisionnels plus efficaces et plus accessibles.
Les systèmes d'entrepôts de données et OLAP sont des systèmes d'information informatiques qui ont pour objectif d'aider les décideurs dans leur prise de décision [Kimball et Ross, 2002; Golfarelli et Rizzi, 2009]. Ces systèmes définissent des outils pour la collecte, le stockage, la consolidation et l'analyse de données afin d'assister les décideurs dans leur prise de décision.
En particulier, un Entrepôt de Données (ED) est défini comme étant « *une collection de données orientées sujet, intégrées, non volatiles, historisées disponibles pour le support du processus de prise de décision* » [Inmon, 2005]. Cette définition met l'accent sur quatre caractéristiques fondamentales des données d'un ED :
Orientées sujet : toutes les données pertinentes à l'analyse sont collectées des sources de données et organisées par sujet d'analyse suivant les besoins analytiques des décideurs.
Intégrées : les données sont généralement le résultat d'une intégration de plusieurs sources de données qui peuvent présenter différentes types d'hétérogénéité (e.g. conventions de nommage, etc.).
Non volatiles : généralement, une fois les données sont chargées dans l'ED, elles ne sont ni modifiées, ni supprimées.
Historisées : des références temporelles sont souvent associées aux données pour garder la trace des différentes versions et préciser les intervalles de validité temporelle des données.

Les systèmes OLAP sont une catégorie d'outils logiciels qui permet aux décideurs une exploration interactive suivant une approche multidimensionnelle à plusieurs niveaux d'agrégation des données des entrepôts de données [Codd et al., 1993; Kimball et Ross, 2002; Proulx et al., 2007]. L'objectif de ces systèmes est de définir des modèles analytiques simples qui transforment les données entreposées en informations stratégiques d'aide à la décision [OMG, 2003a].
Ces systèmes présentent les données sous une forme multidimensionnelle. Sur la base d'un modèle simple décrivant les besoins analytiques des décideurs, appelé modèle d'analyse OLAP ou modèle multidimensionnel (cf. Section 2.3), ces systèmes construisent des représentations multidimensionnelles sous forme d'hypercubes de données pour les données de l'entrepôt. Pour l'exploration interactive de ces hypercubes, ces systèmes implémentent un ensemble d'opérateurs de calcul et de manipulation d'hypercubes de données (cf. Section 2.3.5).
Les différences principales entre les systèmes d'ED et OLAP et les systèmes OLTP sont résumées dans le tableau 2.1 [Codd et al., 1993; Inmon, 2005].

	OLTP	ED - OLAP
Vocation (objectif)	gestion des activités opérationnelles quotidiennes	évaluation de l'activité et aide à la décision
Utilisateurs	nombreux (des milliers), agents opérationnels	moins nombreux (des centaines), analystes et décideurs
Pattern d'utilisation	régulier, prédictible, et fréquent	irrégulier, non prédictible, et moins fréquent
Modèle de données	normalisé (3FN) et optimisé pour les exigences de performance des traitements transactionnels	dénormalisé et optimisé pour les performances des traitements analytiques
Données	opérationnelles orientées transaction, détaillées, courantes (un horizon temporel moyen de 60 à 90 jours), non redondantes et moins volumineuses	orientées analyse, moins détaillées, historiques (un horizon temporel moyen de 5 à 10 ans), redondantes et volumineuses
Mode d'accès	lecture, rajout, modification et suppression	lecture et rajout
Type de traitements	transactionnel : accès à des centaines d'enregistrements	analytique : agrégation et accès à des millions d'enregistrements
Type de technologies	optimisé pour le traitement transactionnel (temps de réponse et gestion d'accès concurrents)	optimisé pour le traitement analytique (historisation et analyse en ligne)

Tableau 2.1. Différences entre systèmes OLTP et OLAP.

2.3 Modèle multidimensionnel

Les systèmes d'ED OLAP se basent sur le modèle MultiDimensionnel (MD). Ce modèle représente les données décisionnelles dans un espace à *n* dimensions, communément appelé hypercube ou cube de données [Kimball et Ross, 2002; Choong et al., 2003]. Les points de cet espace correspondent aux sujets ou faits analysés, et les axes de cet espace correspondent aux critères ou dimensions d'analyse.

2.3.1 Faits et mesures

Les **faits** modélisent les sujets, événements ou phénomènes que les décideurs de l'organisation souhaitent analyser. Chaque fait est décrit par une ou plusieurs mesures. Les **mesures** sont les attributs avec lesquels sont évalués (quantifiés ou qualifiés) les faits. Certaines mesures peuvent être calculées à partir d'autres mesures, on parle alors de mesures dérivées. Au niveau instance, on parle de **valeurs de mesure** pour une mesure et **d'instances de fait** pour un fait.

Ainsi, une analyse multidimensionnelle portant sur un fait "ventes" (cf. Figure 2.2) d'un ensemble de magasins pourra être réalisée en définissant comme mesures "la quantité des produits vendus", "le prix unitaire de vente" et la mesure dérivée "montant de vente" qui est obtenu en multipliant "le prix unitaire de vente" par "la quantité vendue", et comme dimensions "le temps", "les magasins" et "les produits".

.

2.3.2 Dimensions, hiérarchies et niveaux d'agrégation

Les **dimensions** définissent les axes de l'analyse multidimensionnelle. Elles représentent les critères ou les perspectives par rapport auxquelles les décideurs souhaitent analyser et visualiser les faits et leurs mesures. Les dimensions sont organisées en hiérarchies pour permettre l'analyse des mesures à différents niveaux de détail.

Les **hiérarchies de dimension** définissent des structures pour l'agrégation des mesures. Chaque hiérarchie dans une dimension spécifie une façon particulière de regrouper les membres de la dimension. Elle organise ses membres dans différents niveaux hiérarchiques appelés **niveaux d'agrégation** et spécifie les liens d'agrégation ou de regroupement entre ces membres en définissant **les relations d'agrégation** entre ces niveaux. En particulier, dans une hiérarchie, le niveau d'agrégation qui contient les données (membres) les moins détaillées est appelé **niveau d'agrégation racine**, celui qui contient les données les plus détaillées est appelé **niveau d'agrégation feuille.**

Au niveau instance, on parle de **membres** pour les niveaux d'agrégation, **d'instances de hiérarchie** pour les hiérarchies de dimension, d'**instances de dimension** pour les dimensions, de **liens d'agrégation** pour les relations d'agrégation, de **membres racine** pour les membres des niveaux d'agrégation racine et de **membres feuille** pour les membres des niveaux d'agrégation feuille. Une instance de dimension est définie par un ensemble de membres reliés par des liens d'agrégation [Bimonte et al., 2005]. Une instance de hiérarchie est constituée par un sous-ensemble des membres de la dimension.

Par exemple, la figure 2.1 montre un exemple de schéma (Figure 2.1(a)) et d'instance (Figure 2.1(b)) d'une hiérarchie de la dimension "Produits". Cette hiérarchie qui regroupe les produits par type est constituée de deux niveaux d'agrégation, "Produit" et "Type". "Produit" est le niveau d'agrégation feuille et "Type" est le niveau d'agrégation racine. Les membres du niveau d'agrégation "Produit" (Figure 2.1(b)) sont par exemple "Alc 23", "Alc 54", etc.

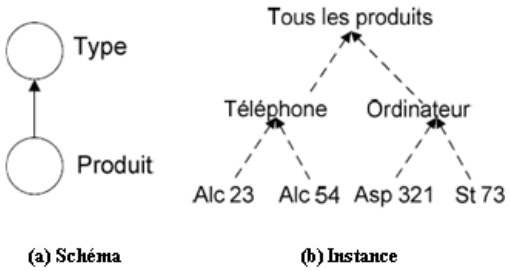


Figure 2.1. Exemple de hiérarchie de dimension - "Produits par type". Issue de [Bimonte, 2007].

En plus des structures statiques que nous venons de décrire (cf. Sections 2.3.1 et 2.3.2) et qui concernent essentiellement la couche ED (cf. Section 2.4), le modèle multidimensionnel comporte aussi des structures (ou aspects) dynamiques qui sont liées à l'agrégation des mesures et au calcul des hypercubes et indicateurs d'analyse [Pardillo et al., 2010b; Boulil et al., 2011]. Ces aspects dynamiques sont décrits dans ce qui suit (Sections 2.3.3, 2.3.4 et 2.3.5).

2.3.3Hypercubes

Le schéma d'un hypercube est composé par un fait et un ensemble de dimensions comme montré en Figure 2.2 qui représente le schéma de l'application des ventes grâce au modèle conceptuel multidimensionnel proposé dans [Malinowski et Zimányi, 2004a]. Cet hypercube présente une dimension temporelle organisée en deux niveaux d'agrégation "Mois" et "Année"; la dimension "Magasins" comportant les niveaux d'agrégation "Magasin", "Ville", "Pays" et "Continent"; et enfin la dimension "Produits" qui contient deux hiérarchies regroupant les produits ("Produit") par "Type" et par "Marque".

Ce modèle permet par exemple de visualiser les mesures (indicateurs) de vente par rapport à différentes combinaisons des niveaux d'agrégation (membres) des différentes dimensions. Par exemple, les totaux des montants et quantités des produits vendus par mois, par ville et par type de produit, etc.

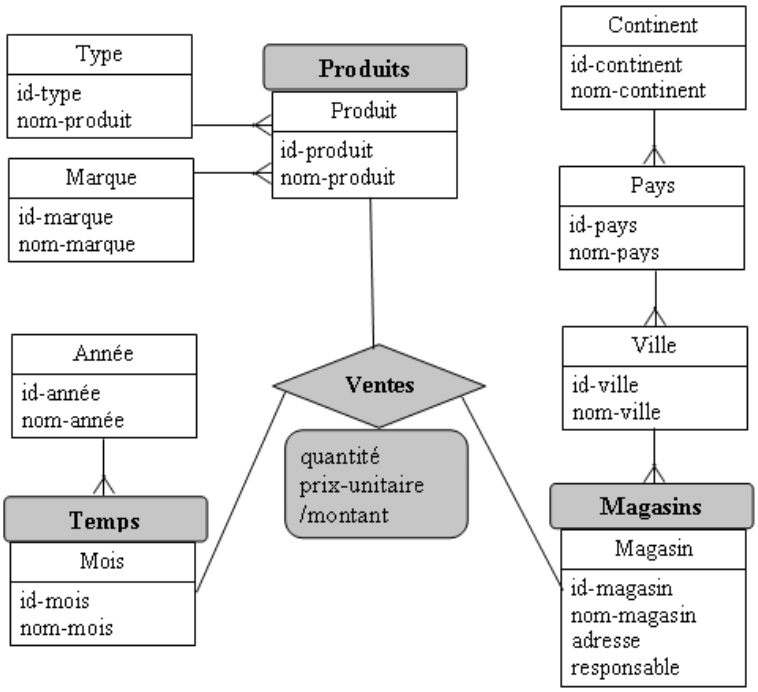


Figure 2.2. Exemple de schéma d'hypercube - Hypercube "Ventes". Adaptée de [Malinowski et Zimányi, 2008].

L'instance d'un hypercube est un ensemble de cellules ou d'instances de fait (cf. Figure 2.3). Chaque cellule contient un ensemble de valeurs de mesures (une valeur pour chaque mesure du fait) et est identifiée par une combinaison des membres des différentes dimensions (un membre par dimension). Les cellules ou instances de fait peuvent être de base ou agrégées. Une **cellule de base** est identifiée par uniquement des membres de dimension feuille, tandis qu'une **cellule agrégée** est identifiée par au moins un membre de dimension non feuille.

Par exemple comme montré en Figure 2.3, la cellule identifiée par le produit "Alc 23", le mois "mar 04" et le magasin "Standa" est une cellule de base; la cellule identifiée par le type de produit "Téléphone", le mois "mar 04" et le magasin "Standa" est une cellule agrégée.

Les valeurs de mesure des cellules agrégées (valeurs de mesure agrégées) sont obtenues par l'agrégation des valeurs de mesure des cellules de base (valeurs de mesure de base). Par exemple, les valeurs de mesure de vente ("quantité" et "montant") en "2004" pour le produit "Alca54" et le magasin "Upim" sont par exemple obtenues en sommant toutes les ventes des mois de 2004.

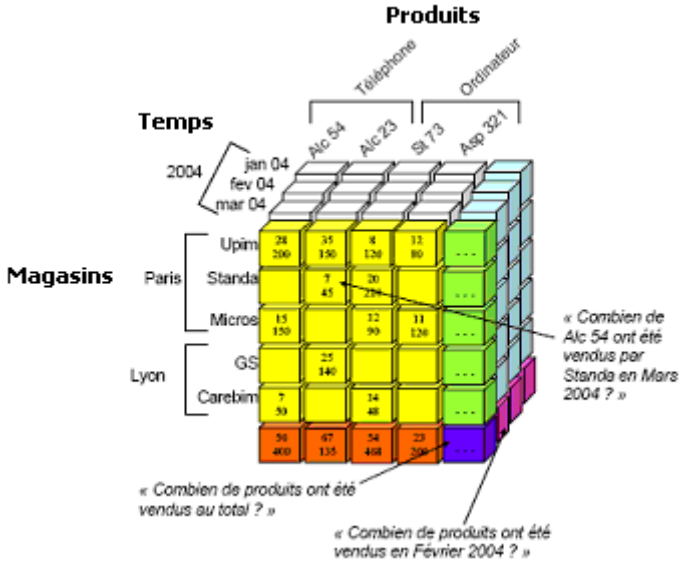


Figure 2.3. Exemple d'instance d'hypecube - hypercube "Ventes". Issue de [Bimonte, 2007].

2.3.4 Agrégation des mesures

L'agrégation des mesures est une opération fondamentale dans les systèmes d'ED. Elle permet de calculer différents indicateurs d'analyse ou agrégats à différents niveaux de détail. Par exemple, les totaux des montants de vente par produit et par année, etc.

Cette agrégation est effectuée de façon transparente aux décideurs lorsque ceux-ci naviguent dans les hiérarchies en exécutant les opérateurs OLAP, particulièrement lorsque ces opérateurs impliquent un changement dans le niveau d'agrégation auquel la mesure est visualisée.

Structurellement, une agrégation dans le contexte des ED met en jeu trois éléments clefs du modèle multidimensionnel à savoir la mesure, la fonction d'agrégation et la hiérarchie de dimension : les mesures sont agrégées le long des hiérarchies de dimension en utilisant des fonctions d'agrégation.

L'agrégation correcte des mesures dépend de certains facteurs connus dans la littérature sous l'appellation de conditions d'agrégeabilité [Lenz et Shoshani, 1997], voir Sections 4.2.2 et 4.5 pour plus de détails :

(A) Disjonction (*Disjointness*) : cette condition garantit par exemple que les ensembles de membres définis par les liens d'agrégation dans une hiérarchie sont disjoints; ceci permet d'éviter le comptage en double des valeurs de mesure.

(B) Complétude (*Completeness*) : cette condition vérifie que ces ensembles sont complets, en d'autres termes tous les membres de dimension et instances de fait sont considérés lors de l'agrégation, ceci dans le but d'éviter le problème d'agrégats incomplets.

(C) Comptabilité de type (*Type compatibility*) : cette condition vérifie que les natures des trois éléments, mesure, dimension et fonction d'agrégation, sont compatibles; en d'autres termes l'application de la fonction d'agrégation à la mesure le long de la dimension a du sens et n'engendre pas des agrégats incorrects.

En ce qui concerne ce dernier point une classification des mesures selon leur additivité[5] a été proposée dans [Lenz et Shoshani, 1997] Cette classification définit trois types de mesures : (a) **mesures de type flux** (*flow measures*), mesures qui peuvent être sommées selon toutes les dimensions (e.g. "montant de vente"); (b) **mesures de type stock** (*stock measures*),

[5] Le mot additivité est particulièrement utilisé pour désigner la possibilité d'agréger la mesure par la fonction Sum [Horner et al., 2004].

mesures qui ne peuvent pas être sommées selon le temps (e.g. "quantité en stock des produits" et "population"), et (c) **mesures de type valeur par unité** (*value per unit measures*), mesures qui ne peuvent pas être sommées selon toutes les dimensions (e.g. "prix unitaire de vente" et "température").

Vue leur importance dans l'analyse décisionnelle et leur influence sur l'agrégeabilité des mesures, les hiérarchies de dimension ont été l'objet de nombreuses recherches [Pedersen et Tryfona, 2001; Mansmann et Scholl, 2006; Malinowski et Zimányi, 2008].
Dans la littérature plusieurs typologies de hiérarchies ont été proposées [Mazón et al., 2009]. Relativement à l'agrégeabilité, les hiérarchies peuvent être classées en deux catégories [Mazón et al., 2009] :

(A) Hiérarchies régulières qui satisfont les deux premières conditions d'agrégeabilité à savoir la complétude et la disjonction. Ces hiérarchies sont strictes, onto et covering.

(a) Hiérarchies strictes : elles spécifient pour chaque membre feuille un seul chemin d'agrégation vers le membre racine de la hiérarchie. Dans les hiérarchies comportant un seul chemin conceptuel (e.g. hiérarchie des "produits par type" de la Figure 2.1) ceci se traduit par la présence d'exactement un membre père pour chaque membre non racine de la hiérarchie. Les hiérarchies strictes garantissent donc la première condition d'agrégeabilité (i.e. la disjonction).

(b) Hiérarchies onto : dans ces hiérarchies, tout membre non feuille possède au moins un membre fils et tous les membres feuille se trouvent au même niveau qui est le niveau d'agrégation feuille de la hiérarchie. Ces hiérarchies sont appelées également hiérarchies **balancées** [Malinowski et Zimányi, 2008]. Par exemple, la hiérarchie des "produits par type" (Figure 2.1) est également une hiérarchie onto

(c) Hiérarchies covering : une hiérarchie est dite covering si elle ne présente pas de raccourcis ou de sauts dans les liens d'agrégation ; en d'autres termes, il n'y a pas de membres père qui sautent des niveaux d'agrégation. Dans la littérature ces hiérarchies sont également connues sous les appellations *non ragged* et *non levels-skipped* [Mazón et al., 2009].

(B) Hiérarchies irrégulières. ces hiérarchies ne satisfont pas l'une ou les deux conditions d'agrégeabilité décrites précédemment (disjonction et complétude), induisant les problèmes de comptage en double de mesures et/ou d'agrégats incomplets. Il s'agit donc de hiérarchies : **non-strictes**, **non-covering** et **non-onto**.

Les systèmes OLAP implémentent un ensemble d'opérateurs pour l'exploration des hypercubes de données comme décrit dans la section suivante.

2.3.5Opérateurs OLAP

Les opérateurs OLAP typiques sont (un état de l'art des opérateurs OLAP est présenté dans [Ravat et al., 2010]) :

Roll-up : permet d'agréger les valeurs de mesure en montant dans une hiérarchie de dimension. Par exemple, le Roll-up du niveau "Mois" vers "Année", permet de grouper et d'agréger les ventes mensuelles par année en utilisant par exemple la fonction Sum.

Drill-down : c'est l'opération inverse à Roll-up, elle augmente le niveau de détail de la mesure en descendant dans la hiérarchie de dimension.

Slice : effectue une sélection des cellules de l'hypercube en utilisant une condition (prédicat) définie sur les membres d'une dimension; il retourne un sous hypercube de l'hypercube initial.

Dice : effectue une sélection en utilisant une condition définie sur deux dimensions ou plus.

Projection : permet de sélectionner un sous-ensemble de mesures de l'hypercube.

Pivot (*Rotate*) : effectue un pivotage des axes de l'hypercube pour obtenir différentes alternatives de présentation des données décisionnelles.

2.4 Architecture typique d'un système OLAP

L'architecture typique des systèmes d'ED et OLAP [Hackney, 2002; Malinowski et Zimányi, 2008] est constituée de quatre couches logicielles (cf. Figure 2.4). En amont de cette architecture se trouve les sources de données qui alimentent le système OLAP en données décisionnelles et qui peuvent présenter différentes hétérogénéités (e.g. plateforme technologique, encodage, conventions de nommage, unités de mesure, etc.) [Galhardas et al., 2001].

(1) Couche d'intégration (ETL) : les données décisionnelles extraites des sources subissent un ensemble de transformations pour les rendre sous un schéma unifié et homogène, qui est celui de l'ED. Cette importante phase d'intégration, qui peut être plus au moins complexe à réaliser dépendant du degré d'hétérogénéité des sources, incombe le plus souvent aux outils ETL (*Extraction, Transformation and Loading tools*). Ces outils extraient, nettoient, transforment et chargent les données décisionnelles sources dans l'ED.
*De façon générale, les outils ETL sont des outils qui permettent la transformation d'une ou plusieurs bases de données sources vers une ou plusieurs bases de données cibles***.** Dans les systèmes d'ED, ils sont généralement utilisés pour automatiser l'intégration et la réconciliation des sources de données au sein de l'ED. Dans ce cadre d'utilisation, ces outils exécutent de façon régulière un processus de quatre phases [Jarke et al., 2003] : Extraction, Nettoyage, Transformation, et Chargement des données analytiques sources dans l'ED.

Extraction : sélection et extraction régulière (par rapport à l'occurrence d'un événement temporel (e.g. fin du mois) ou intemporel (e.g. changement de données sources)) des données sources utiles à l'analyse décisionnelle;
Nettoyage : le but de cette phase est d'améliorer la qualité des données qui peut être mauvaise dans les sources en supprimant/rectifiant certaines erreurs et inconsistances (e.g. doublons, données manquantes, données incorrectes (e.g. 30/02/2011), données inconsistantes (e.g. Fr et France).
Transformation : permet la conversion des données nettoyées vers le schéma de l'ED.
Chargement : permet le chargement des données intégrées dans l'ED.

(2) Couche de stockage (ED) : cette couche comprend (i) l'ED, (ii) le référentiel de métadonnées et éventuellement (iii) un ou plusieurs magasins de données. L'ED, appelé également ED primaire, définit un lieu d'intégration et d'historisation pour toutes les données de l'organisation. Cet ED peut être lui aussi la source directe à la création de petits ED focalisés sur des besoins analytiques particuliers (par exemple, relatifs à un groupe d'utilisateurs ou répondant à certaines exigences d'optimisation), appelées magasins de données (**Data marts**). Le référentiel de métadonnées [Kelly, 2007] stocke des métadonnées (données sur les données) pour décrire toutes les phases du processus d'entreposage et d'exploration des données (sources de données, processus ETL, schémas de données, profils utilisateur, unités de mesure, etc.). Ces métadonnées peuvent être utilisées pour meilleure administration de ces processus ou servir pour l'amélioration de la compréhension des données multidimensionnelles par les décideurs. Ce référentiel doit être accessible par toutes les autres couches de l'architecture.

(3) Serveur OLAP : le serveur OLAP permet d'effectuer une analyse de données conforme au paradigme multidimensionnel, avec des temps de réponse optimisés [Codd et al., 1993]. Un serveur OLAP fournit aux utilisateurs une représentation multidimensionnelle des données sous forme d'un ensemble d'hypercubes et implémente un ensemble d'opérateurs OLAP (Roll-up, Drill-down, etc.) qui permettent d'explorer ces hypercubes.

(4) Client OLAP : cette couche définit une série d'interfaces utilisateurs intuitives pour l'exploration interactive et multidimensionnelle des données. Ces interfaces permettent de déclencher les opérateurs OLAP et présentent l'information en utilisant différents types d'affichages interactifs : tableaux croisés dynamiques, histogrammes et diagrammes statistiques.

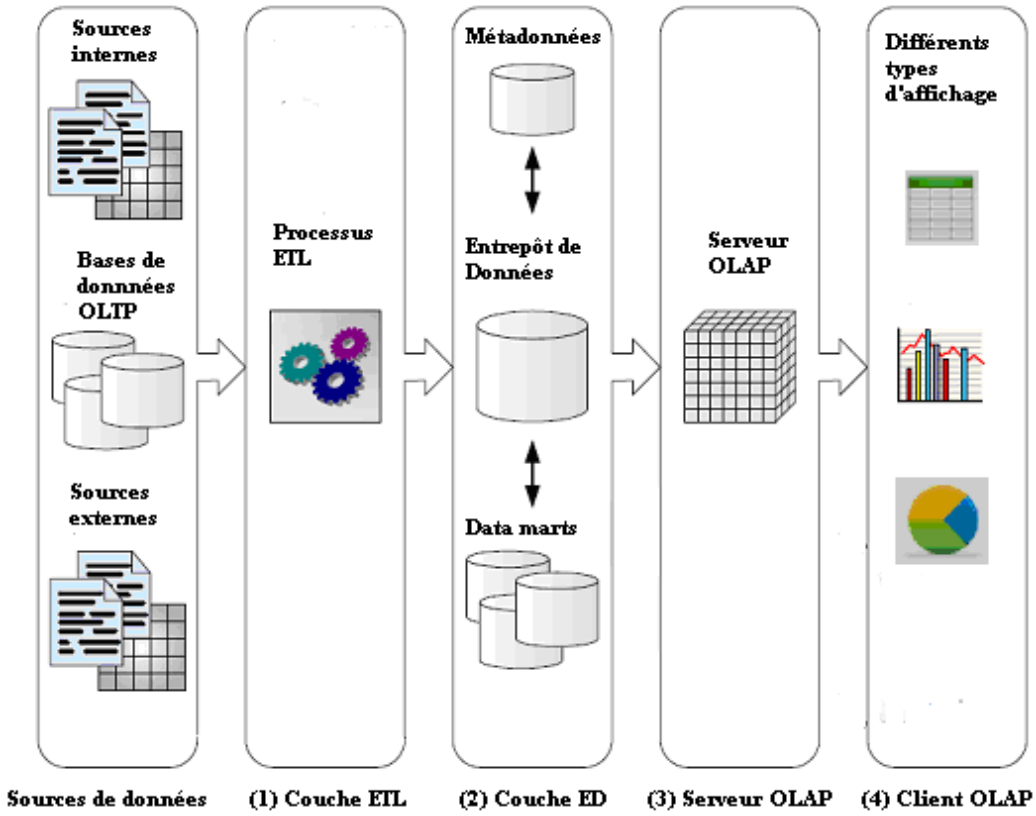


Figure 2.4. Architecture typique d'un système OLAP. Adaptée de [Malinowski et Zimányi, 2008].

2.4.1ROLAP, MOLAP et HOLAP

Il existe différentes implémentations des systèmes OLAP [Salka, 1998; Chaudhuri et al., 2001] : (a) ROLAP (*Relational OLAP*) qui se base sur une base de données relationnelle; (b) MOLAP (*Multidimensional OLAP*) qui utilise une base de données multidimensionnelle; et (c) HOLAP (*Hybrid OLAP*) qui combine les deux technologies.

2.4.1.1 Systèmes ROLAP

Dans les implémentations ROLAP, un Système de Gestion de Bases de Données Relationnelles – SGBDR (e.g. Oracle, PostgreSQL, etc.) est utilisé pour l'entreposage des données (cf. Figure 2.5). Un middleware spécialisé, appelé serveur ROLAP, permet de simuler la vue multidimensionnelle des données. Ce moteur agit comme un traducteur entre le SGBDR et les outils client OLAP. Il traduit les requêtes OLAP formulées par les utilisateurs au niveau de l'outil client OLAP vers des requêtes SQL pour être exécuter par le

SGBDR. Les résultats (SQL) retournés par le SGBR sont ensuite traduits par ce moteur ROLAP dans le langage approprié pour être visualiser au niveau du client OLAP.
Ce type d'implémentation nécessite essentiellement la définition de deux types de métadonnées [OMG, 2003a] : (a) le schéma relationnel de l'ED et (b) le schéma multidimensionnel ROLAP, cf. Figure 2.5.
Le schéma relationnel de l'ED définit les structures relationnelles (tables et leurs relations) qui sont utilisées pour le stockage de données analytiques. Ces tables relationnelles sont souvent organisées de manière à révéler la dimensionnalité intrinsèque des données en utilisant des schémas relationnels particuliers adaptés à l'analyse multidimensionnelle [Kimball, 1996; Vassiliadis et Sellis, 1999], schémas en étoile, en flocons de neige ou hybrides. Ces différents types schémas sont expliqués plus loin en Section 2.5.2.1.

Le schéma multidimensionnel ROLAP représente les structures d'analyse multidimensionnelle (hypercubes, dimensions, hiérarchies, indicateurs ou mesures, etc.), ainsi que les mappings de ces structures vers les structures relationnelles définies par le schéma relationnel de l'ED.

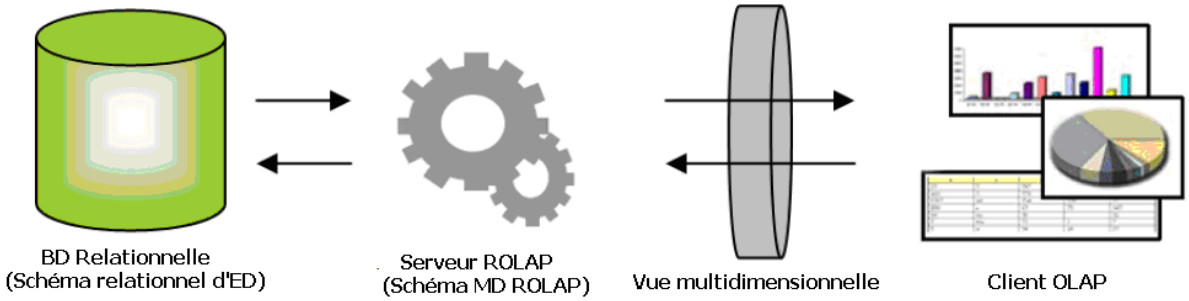


Figure 2.5. Architecture d'un système ROLAP. Adaptée de [Rivest, 2000].

2.4.1.2 Systèmes MOLAP

Les implémentations MOLAP utilisent des Systèmes de Gestion de Bases de Données Multidimensionnelles natifs (SGBDM) [Kotidis et Roussopoulos, 1998]. Ces systèmes supportent nativement le paradigme multidimensionnel : ils maintiennent les données décisionnelles dans structures multidimensionnelles ad hoc (matrices ou vecteurs à n dimensions) et définissent les opérations OLAP (e.g. Roll-up, Drill-down, etc.) pour la manipulation et l'exploration faciles et rapides de ces structures.
A part les cellules dérivées, dans ces implémentations toutes les cellules (agrégées et de base, cf. Section 2.3.3) des différents hypercubes sont précalculées à partir de l'ED ou des bases sources et stockées dans ces structures ad hoc. Les opérateurs OLAP définis par le SGBDM permettent ensuite à l'utilisateur d'explorer ces cellules d'une façon simple et rapide sans avoir à effectuer des calculs complexes. En revanche, ce précalcul implique la création d'une copie (duplication) des données de l'ED [Salka, 1998; Inmon, 2005].

2.4.1.3 Systèmes HOLAP

Les implémentations HOLAP combinent les deux technologies ROLAP et MOLAP [Salka, 1998]. Une partie des données est stockée dans des structures relationnelles gérées par un SGBDR et une autre partie est maintenue dans des structures multidimensionnelles gérées par un SGBDM. L'approche adoptée est généralement de stocker les données auxquelles les utilisateurs accèdent le plus fréquemment dans la partie multidimensionnelle et les autres dans la partie relationnelle.

2.4.2Comparaison des systèmes ROLAP, MOLAP et HOLAP

Les systèmes ROLAP permettent une grande souplesse dans la gestion de gros volumes de données (chargement, stockage et mise à jour), un meilleur passage à l'échelle et une meilleure administration des données (e.g. sécurité, etc.), cf. Tableau 2.2. Relativement aux ROLAP, les systèmes MOLAP offrent des performances moindres concernant ces aspects (volume de données et passage à l'échelle) à cause notamment du précalcul des cellules. En revanche, ils offrent de meilleures performances en termes de temps de réponse aux requêtes. Les systèmes HOLAP offrent un compromis des avantages et inconvénients des systèmes ROLAP et MOLAP.

Le tableau 2.2, adapté des travaux[Pedersen et Jensen, 2001; Malinowski et Zimányi, 2008; Golfarelli et Rizzi, 2009], résume les avantages et inconvénients de chacune de ces options d'implémentation.

	ROLAP	**HOLAP**	**MOLAP**
Requêtage	les plus lents	modérés	les plus rapides
Chargement de données	les plus efficaces	modérés	les moins efficaces
Passage à l'échelle	les meilleurs	modérés	les plus faibles
Volume de données	gros volumes	volumes moyens	petits volumes

Tableau 2.2. Comparaison des systèmes ROLAP, MOLAP et HOLAP.

2.5 Modélisation multidimensionnelle

De façon similaire aux systèmes OLTP, le processus de développement des systèmes OLAP peut être organisé en quatre grandes phases [Prat et al., 2006; Niedrite et al., 2009] : (1) définition des besoins, (2) modélisation conceptuelle, (3) modélisation logique et (4) modélisation physique.

(1) Dans la phase de définition des besoins, il s'agit d'identifier, collecter et décrire les besoins informationnels[6] des décideurs dans un document de spécification des besoins. Pour ceci n'importe quelle méthode d'analyse de systèmes peut être utilisée, comme les interviews, les formulaires, etc.

(2) Le but de la modélisation conceptuelle est l'élaboration d'une spécification abstraite, claire, concise et indépendante de l'implémentation du système permettant de répondre à ces besoins informationnels, i.e. le modèle multidimensionnel conceptuel du système OLAP cible. A ce niveau, plusieurs modèles multidimensionnels ont été proposés mais aucun modèle n'a été adopté comme standard (cf. Section 2.5.1).

(3) La modélisation logique permet de traduire le modèle conceptuel dans le paradigme d'implémentation choisi[7], i.e. définir le modèle multidimensionnel logique (cf. Section

[6] Contrairement aux systèmes OLTP où on parle d'exigences fonctionnelles, dans l'OLAP on parle de besoins informationnels ou d'analyse [Niedrite et al., 2009]

[7] Un paradigme d'implémentation représente les concepts communs à un ensemble de technologies se basant sur la même façon de voir le monde (par exemple, le Relationnel (R), l'Orienté Objet (OO) et l'Objet-Relationnel (OR)).

2.5.2.1). A ce niveau le schéma en étoile (et ses variantes) proposé par Ralph Kimball [Kimball, 1996] est largement adopté pour les implémentations ROLAP. Pour ce type d'implémentations, plusieurs travaux se sont intéressés à la définition des règles de mapping conceptuel-logique [Prat et Akoka, 2002; Malinowski et Zimányi, 2006], qui une fois implémentées dans des AGL permettent d'obtenir automatiquement le modèle logique [Mazon et Trujillo, 2008].

(4) Le modèle multidimensionnel physique (cf. 2.5.2.2) est décrit dans le langage supporté par la plateforme d'implémentation cible (e.g. Oracle ou PostgreSQL). C'est un modèle très technique qui considère les fonctionnalités propres à cette dernière (e.g. indexes). Il est à son tour obtenu à partir du schéma logique en appliquant d'autres règles de transformation [Mazon et Trujillo, 2008].

Dans la suite de cette section, nous allons présenter les principaux modèles multidimensionnels conceptuels (cf. Section 2.5.1) ainsi que les principaux modèles multidimensionnels logiques ROLAP et les techniques de modélisation multidimensionnelle physique les plus fréquentes des systèmes ROLAP (cf. Section 2.5.2).

2.5.1Modélisation multidimensionnelle conceptuelle

La capacité d'un modèle conceptuel à représenter les exigences du système d'une manière abstraite, claire et indépendante de l'implémentation est cruciale dans le cycle de vie de développement de tout système d'information. Cela est particulièrement vrai dans le contexte des ED où en raison du niveau de complexité, la difficulté, le temps et les erreurs de développement sont amplifiés [Torlone, 2003].
En dépit du nombre important de modèles multidimensionnels conceptuels proposés dans la littérature, il n'existe pas jusqu'à maintenant de modèle standard qui soit adopté [Abelló et al., 2006; Kamble, 2008]. En général, les modèles proposés dans la littérature permettent la représentation des principaux aspects statiques et quelques aspects dynamiques de la modélisation multidimensionnelle. Ils sont soit des extensions de langages de modélisation standards (UML, ER, Logique de Description, etc.) ou soit des formalismes ad hoc[8] [Abelló et al., 2006].
L'avantage des modèles basés sur des standards, comme UML et ER, est qu'ils minimisent les temps/efforts d'apprentissage et de compréhension car les concepteurs et utilisateurs ont l'habitude de travailler avec ces langages [Torlone, 2003]. UML en particulier est très connu pour sa grande capacité à modéliser des aspects statiques et dynamiques complexes; ceci est en partie du à sa facilité d'extension et à son interopérabilité avec d'autres standards comme le langage de contraintes objet OCL [Pinet et Schneider, 2009; OMG, 2011c]. UML est aussi le langage de modélisation le plus intégré dans les AGL et les règles de son mapping vers différents SGDB (e.g. Oracle, PostgreSQL, etc.) sont bien définies [Prat et Akoka, 2002], ceci permet la génération automatique des modèles logiques et physiques, ce qui optimise davantage les temps de développement.

Comme notre but est de proposer un modèle spatio-multidimensionnel conceptuel basé sur UML (cf. Section 1.3) qui permet l'expression des CI SOLAP avec (Spatial) OCL au niveau conceptuel et leur implémentation d'une façon automatique, nous allons nous focaliser, dans ce qui suit, sur les modèles UML car ces modèles supportent le langage OCL [Pinet et al., 2007; Pinet et Schneider, 2009]. D'autres travaux sont présentés en raison de leur

[8] Les modèles ad hoc sont des modèles qui ne se basent pas sur des langages de modélisation de données standards.

représentation de certains aspects de la qualité OLAP, d'autres encore parce qu'ils présentent des extensions spatiales (cf. Section 3.4.1).

Les principaux modèles UML proposés dans la littérature sont ceux présentés dans [Abelló et al., 2006; Lujan-Mora et al., 2006; Pinet et Schneider, 2009; Prat et al., 2010].
[Abelló et al., 2006] propose un modèle UML appelé YAM^2 qui définit un ensemble riche de concepts multidimensionnels et permet la représentation de plusieurs hypercubes reliés. Une caractéristique intéressante de YAM^2 est liée à son usage des paquetages UML pour organiser les éléments multidimensionnels selon trois niveaux de détail, simplifiant ainsi la lecture des modèles complexes. Ce modèle représente également certaines contraintes d'agrégation (cf. Section 4.5.2). Comme reconnu par les auteurs, les différents types de relations UML liant les éléments multidimensionnels rendent ce modèle complexe à utiliser.
[Lujan-Mora et al., 2006] simplifie le modèle YAM^2 et propose un profil UML qui définit un ensemble de stéréotypes, de valeurs marquées et de contraintes OCL. Les contraintes OCL du profil sont utilisées pour formaliser les concepts multidimensionnels du profil (i.e. stéréotypes et valeurs marquées) et éviter leurs mauvaises utilisations par les concepteurs. Ce profil est implémenté dans l'AGL Rational Rose. Une extension spatiale de ce profil est proposée dans [Glorio et Trujillo, 2008], cf. Section 3.4.1.1.1.
[Pinet et Schneider, 2009] propose un modèle UML qui unifie (i) les représentations conceptuelles des niveaux d'agrégation et des faits et (ii) celles des relations d'agrégation et des relations fait-dimension, offrant plus de flexibilité dans l'analyse et la modélisation multidimensionnelle. Ce modèle définit quelques spécialisations UML : *Classe identifiée* pour représenter les faits et les niveaux d'agrégation; *Association d'agrégation* pour représenter les relations d'agrégation et les relations fait-dimension, etc. Ce travail montre également l'utilisation d'OCL pour modéliser des CI de données qui vérifient la consistance des données intégrées de plusieurs sources. Une extension spatiale de ce modèle est proposée dans [Pinet et Schneider, 2010], cf. Section 3.4.1.1.1.
Enfin, [Prat et al., 2010] propose une extension d'UML qui organise les concepts multidimensionnels au sein de deux modèles : (i) le modèle de l'ED et (ii) le modèle de l'hypercube. Le modèle de l'ED représente les structures de la base de données (i.e. l'ED) au niveau conceptuel grâce à des concepts tels que le Fait (*Fact*), la Dimension (*Dimension*), etc. Le modèle de l'hypercube permet de définir des structures d'analyse OLAP (e.g. hypercube, cellule, hiérarchie, etc.) et est mappé vers le modèle de l'ED. Les concepts multidimensionnels définis sont formalisés comme des spécialisations de métaclasses UML, e.g. *Fact* spécialise la métaclasse UML *class.* En utilisant le standard OMG de représentation des règles de production PPR (*Production Rule Representation language*)[OMG, 2009], les auteurs expriment également quelques contraintes d'agrégation des mesures (cf. Section 4.5.2).

Les travaux de [Sapia et al., 1998; Tryfona et al., 1999; Malinowski et Zimányi, 2006; Kamble, 2008] proposent des extensions du modèle ER. En particulier, les travaux de [Malinowski et Zimányi, 2004a; Malinowski et Zimányi, 2006, 2008] présentent un modèle intéressant, appelé MultiDimER, pour la modélisation conceptuelle des ED. Ces derniers s'intéressent plus particulièrement à la modélisation des hiérarchies de dimension complexes (e.g. non strictes, etc.). Ils proposent une classification des hiérarchies de dimension ainsi que des notations graphiques pour les représenter au niveau conceptuel. MultiDimER est le seul modèle qui distingue explicitement les différentes hiérarchies composant une dimension. Une extension spatiale de ce modèle est proposée dans [Malinowski et Zimányi, 2005; Malinowski et Zimányi, 2008], cf. Section 3.4.1.1.2.

Les travaux de [Golfarelli et al., 1998; Husemann et al., 2000; Torlone, 2003] introduisent des modèles ad hoc qui définissent un support graphique pour la plupart des concepts

statiques et quelques aspects dynamiques mais qui nécessitent plus d'efforts d'apprentissage et de développement. En particulier le modèle DFM (*Dimensional-Fact Model*) proposé par [Golfarelli et al., 1998] définit une notation graphique particulière pour représenter certaines contraintes d'agrégation (i.e. les fonctions d'agrégation qui peuvent être utilisées pour agréger les mesures le long des dimensions), cf. Section 4.5.2.

D'autres travaux comme [Nguyen et al., 2000a; Franconi et Kamble, 2004] proposent des modèles ad hoc non graphiques qui formalisent les concepts multidimensionnels en utilisant par exemple la théorie des ensembles. L'utilisation de ces modèles pour la modélisation conceptuelle des ED n'est pas une tache évidente en raison du manque de notations graphiques.

En règle générale et comme nous allons le montrer en Section 4.7, tous ces modèles ne satisfont pas tous les critères nécessaires à la spécification des CI SOLAP.

2.5.2Modélisations logique et physique ROLAP

Dans cette section nous présentons les modélisations logique (Section 2.5.2.1) et physique (Section 2.5.2.2) des systèmes ROLAP.

2.5.2.1 Modélisation logique ROLAP

Dans les implémentations ROLAP, l'ED (i.e. la base de données) est organisé de façon à révéler la dimensionnalité intrinsèque des données décisionnelles suivant des schémas particuliers qui sont des extensions du schéma relationnel. Le schéma relationnel de l'ED inclut un ensemble de **tables de fait** (les tables relationnelles correspondant aux faits conceptuels) et un ensemble de tables de dimension (les tables représentant les dimensions et leurs niveaux d'agrégation) [Kimball, 1996]. Les tables de fait référencent les tables de dimension en utilisant des clés étrangères. Les tables de dimension peuvent être plus ou moins dénormalisées, ce qui donne lieu à trois types de schéma : en étoile, en flocons de neige ou hybride.

2.5.2.1.1 Schéma en étoile

Le schéma en étoile est constitué d'une **table de fait** qui référence un ensemble de **tables de dimension.** Chaque table de dimension correspond à une dimension conceptuelle. Elle contient une clé primaire et un ensemble de colonnes qui décrivent la dimension; chaque colonne correspond à un attribut de dimension conceptuel. La table de fait correspond à un fait conceptuel et inclut une colonne pour chaque mesure du fait. Cette table référence les tables de dimension en utilisant des clés étrangères; et sa clé primaire est définie par la composition de l'ensemble de ces clés étrangères.
Les tables de dimension sont totalement dénormalisées puisque tous les niveaux d'agrégation d'une dimension sont représentés par une seule table. Ceci permet d'optimiser les performances en temps de réponse aux requêtes (car le nombre de jointures est minimisé); en revanche, cette dénormalisation introduit de la redondance dans les données ce qui nécessite plus d'espace de stockage [Kimball, 1996].

Ci-dessous (cf. Figure 2.6) un exemple de schéma en étoile représentant les ventes.

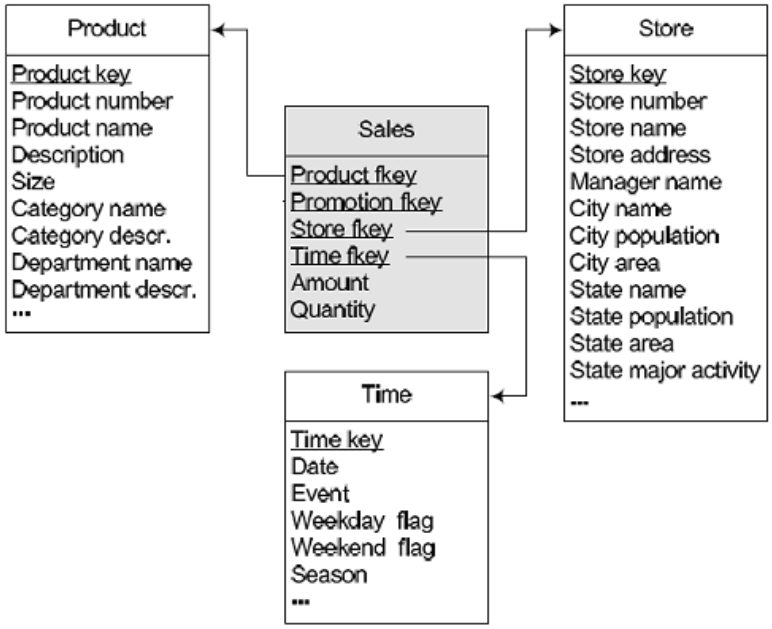


Figure 2.6. Exemple de schéma en étoile. Hypercube des "Ventes". Issue de [Malinowski et Zimányi, 2008].

2.5.2.1.2 Schéma en flocons de neige

Le schéma en flocons de neige permet de supprimer la redondance de données introduite par le schéma en étoile. Dans ce schéma, chaque niveau d'agrégation de la dimension est représenté par une table. Ces tables sont reliées par des contraintes d'intégrité référentielle (i.e. clés étrangères) qui traduisent les relations conceptuelles entre les niveaux d'agrégation. Comme pour le schéma en étoile, la table de fait est reliée aux tables de dimension correspondant aux niveaux d'agrégation les plus fins des dimensions, en utilisant des clés étrangères. Cette modélisation permet d'éviter certaines redondances de données, mais affecte négativement les performances en temps de réponse aux requêtes puisqu'elle augmente le nombre de jointures nécessaires entre les tables.

2.5.2.1.3 Schéma hybride

Le schéma hybride, appelé aussi *starflake*, est la combinaison des deux types de schéma décrits précédemment : certaines dimensions sont normalisées (dans le sens où on a une table par niveau d'agrégation), d'autres sont partiellement normalisées (i.e. certains niveaux d'agrégation sont normalisés et d'autres non), et d'autres encore sont dénormalisées.

2.5.2.1.4 Schéma en constellation

Lorsque le nombre de tables de fait est supérieur ou égal à 2, ce qui est généralement le cas[9], on parle alors souvent de constellation d'étoiles, **schéma en constellation**. Un schéma en constellation est défini comme un ensemble (constellation) d'étoiles partageant des tables de dimension. Il peut contenir des dimensions plus ou moins normalisées.
Le choix entre la normalisation ou dénormalisation des dimensions est souvent fait en considérant deux critères : le coût de stockage et les performances en temps de réponse aux requêtes attendues [Kimball et Ross, 2002].

[9] Un entrepôt de données est sensé représenté tous les sujets d'analyse d'une organisation, ce qui donne lieu généralement à plusieurs tables de fait.

2.5.2.2 Modélisation physique ROLAP

La modélisation physique des ED est une phase très importante notamment dans le contexte des systèmes ROLAP [Adamson, 2006]. Le but est de trouver des techniques pour un stockage efficace et un accès rapide à l'information. Dans cette phase, le modèle MD logique est traduit dans le langage supporté par la plateforme d'implémentation cible. Ensuite un ensemble de techniques d'optimisation sont définies pour améliorer les performances (en temps de réponse et coût de stockage) du système décisionnel. Vues matérialisées, index optimisés et fragmentation sont les techniques les plus couramment utilisées.
La matérialisation de vues consiste à précalculer et stocker physiquement (matérialiser) certaines requêtes (e.g. les requêtes les plus fréquentes ou les plus coûteuses) dans des tables [Adamson, 2006]. Ceci permet d'améliorer les performances en temps de réponse aux requêtes au détriment du coût de stockage (i.e. l'espace et le temps de mise à jour nécessaires). Les problématiques liées à cette technique notamment la sélection (quelles sont les vues à matérialiser ?), la maintenance de ces vues (leurs mises à jour) et la réécriture de requêtes utilisateurs en fonction des vues matérialisées ont été largement étudiées dans la littérature et beaucoup de solutions furent proposées.

Le but de l'indexation est d'accélérer l'exécution des requêtes en permettant l'accès direct aux données. Deux types d'indexes sont couramment utilisés dans les ED : les indexes binaires (ou Bitmap) et les indexes de jointure [Bellatreche et al., 2004]. Les indexes binaires sont adaptés aux attributs à faible cardinalité. Ces indexes nécessitent peu d'espace de stockage et peuvent être manipulés via des opérateurs logiques, ce qui les rend très efficaces. Les indexes de jointure matérialisent les jointures relationnelles entre deux tables en stockant les couples d'identifiants de tuples joints dans une table à deux colonnes.

Finalement, la fragmentation est une technique d'optimisation de performance qui consiste à diviser le contenu d'une table en plusieurs fragments de telle sorte que la combinaison de ces fragments produit l'intégralité des données source, sans perte ou ajout d'information [Bellatreche, 2000]. Cette technique nécessite aussi des méthodes de réécriture et d'exécution parallèle de requêtes.

2.6 Conclusion

Dans ce chapitre nous avons décrit les notions essentielles des systèmes d'entrepôts de données et OLAP. Ces systèmes ont été définis comme des SAD particuliers permettant l'intégration, l'historisation et l'analyse multidimensionnelle interactive de plusieurs sources données; et leurs différences par rapport aux systèmes OLTP ont été mentionnées. Il a été ensuite présenté le modèle multidimensionnel sur lequel se base ces systèmes et qui définit (i) des structures de données (*dimension*, *mesure*, *hiérarchie*, etc.) permettant de représenter les informations décisionnelles dans un espace à n dimensions, en accord avec la vision des décideurs; (ii) des fonctions (e.g. Sum) permettant d'agréger les mesures lorsque les utilisateurs naviguent dans les hiérarchies de dimension; et enfin (iii) des opérateurs (e.g. Roll-up) permettant l'exploration rapide des hypercubes de données. Nous avons souligné que l'agrégation des mesures constitue une opération fondamentale dans ces systèmes et que celle-ci dépend de trois conditions d'agrégeabilité : *disjonction*, *complétude* et *compatibilité de type*. Concernant l'additivité (applicabilité de la fonction d'agrégation Sum), il a été distingué trois types de mesures : (i) les mesures de type *flux* qui peuvent être sommées selon tout type de dimension; (ii) les mesures de type stock qui ne peuvent pas être sommées selon la dimension temporelle; et (iii) les mesures de type *valeur par unité* dont la somme le long de tout type de dimension n'a généralement pas de sens. Toujours par rapport à l'agrégeabilité, il a été rappelé que le type de la hiérarchie de dimension (régulière ou irrégulière), qui sert de support à l'agrégation, est également à considérer. Nous avons ensuite

décrit les couches logicielles composant l'architecture typique de ces systèmes avec une présentation et une comparaison des trois types d'implémentations possibles de cette architecture (ROLAP, MOLAP et HOLAP). Ce chapitre se termine par une section sur la modélisation multidimensionnelle qui présente brièvement les principaux modèles conceptuels en mettant l'accent sur les modèles basés sur UML; également la modélisation logique des systèmes ROLAP où il est souligné que le recours à un schéma en étoile, en flocons de neige ou hybride, dépend du coût de stockage et des performances attendues en termes de temps de réponse aux requêtes; et enfin les principales techniques de modélisation physique des systèmes ROLAP (vues matérialisées, index optimisés et fragmentation) permettant d'optimiser le stockage et l'accès à l'information sont brièvement décrites.

Dans ce chapitre, nous avons défini les concepts qui vont servir de base aux définitions du chapitre suivant, qui lui porte sur les systèmes d'entrepôts de données spatiales et SOLAP; ces systèmes, sur les quels portent nos travaux de thèse, représentent des extensions des systèmes OLAP pour la prise en compte de l'information spatiale dans l'analyse multidimensionnelle.

3 Chapitre 3 : Entrepôts de Données Spatiales et Spatial OLAP

3.1 Introduction

Les entrepôts de données spatiales et les systèmes SOLAP (Spatial OLAP) représentent une solution efficace pour l'analyse spatio-multidimensionnelle de données spatiales. Les Entrepôts de Données Spatiales (EDS) permettent d'intégrer, d'organiser de façon multidimensionnelle, de stocker et d'historiser de très gros volumes de données spatiales et non spatiales provenant de multiples sources pour supporter le processus de prise de décision au sein d'une organisation [Stefanovic et al., 2000].

Les systèmes SOLAP sont une catégorie d'outils logiciels qui permettent l'exploration interactive basée sur approche spatio-multidimensionnelle à plusieurs niveaux de détail des EDS [Bédard et al., 2007]. Ces systèmes étendent les systèmes OLAP par de nouveaux concepts/opérateurs (concepts/opérateurs SOLAP) qui permettent d'enrichir l'analyse OLAP par des analyses spatiales. En plus de la navigation spatio-multidimensionnelle via l'exécution des opérateurs spécifiques au SOLAP (e.g. Drill-down spatial), ces systèmes permettent également la visualisation des résultats de requêtes décisionnelles sous formes de cartes interactives, ce qui permet une meilleure compréhension du phénomène analysé.

Les EDS et les systèmes SOLAP se basent sur le modèle spatio-multidimensionnel (cf. Section 3.2). Ce modèle étend le modèle multidimensionnel classique par d'autres concepts, tels que la mesure et la dimension spatiale, permettant l'intégration de l'information spatiale dans l'analyse multidimensionnelle.

La suite de ce chapitre est organisée comme suit. Dans la section 3.2, nous allons décrire les concepts spécifiques au modèle spatio-multidimensionnel en précisant les définitions de ces concepts que nous considérons dans nos travaux et en détaillant certaines caractéristiques dynamiques qui peuvent influencer la qualité d'agrégation (cf. Section 3.2.1). La section 3.3 présente l'architecture typique et générique des systèmes SOLAP et précise les composants technologiques de l'architecture que nous avons choisie d'utiliser dans nos travaux afin de valider nos propositions. Enfin, la section 3.4 présente un état de l'art sur les principaux modèles spatio-multidimensionnels conceptuels proposés dans la littérature en détaillant les modèles basés sur UML et ER (cf. Section 3.4.1).

3.2 Modèle spatio-multidimensionnel

Les systèmes d'EDS et SOLAP se basent sur le modèle spatio-multidimensionnel. Ce modèle étend le modèle multidimensionnel des systèmes d'ED et de l'OLAP avec de nouveaux concepts spatio-multidimensionnels (e.g. mesure spatiale, dimension spatiale, etc.).

Dans la suite de cette section, nous présentons les principales définitions proposées dans la littérature pour les concepts spatio-multidimensionnels, ainsi que celles que nous utiliserons tout le long de ce travail pour définir les différentes classes de CI SOLAP.

Pour une meilleure compréhension des définitions proposées, nous reprenons l'exemple de l'application d'analyse des ventes décrit en Section 2.3 auquel nous rajoutons des attributs et relations spatiales topologiques modélisés en utilisant les notations graphiques du modèle

conceptuel spatio-multidimensionnel Spatial MultiDimER [Malinowski et Zimányi, 2008] (cf. Figure 3.1).

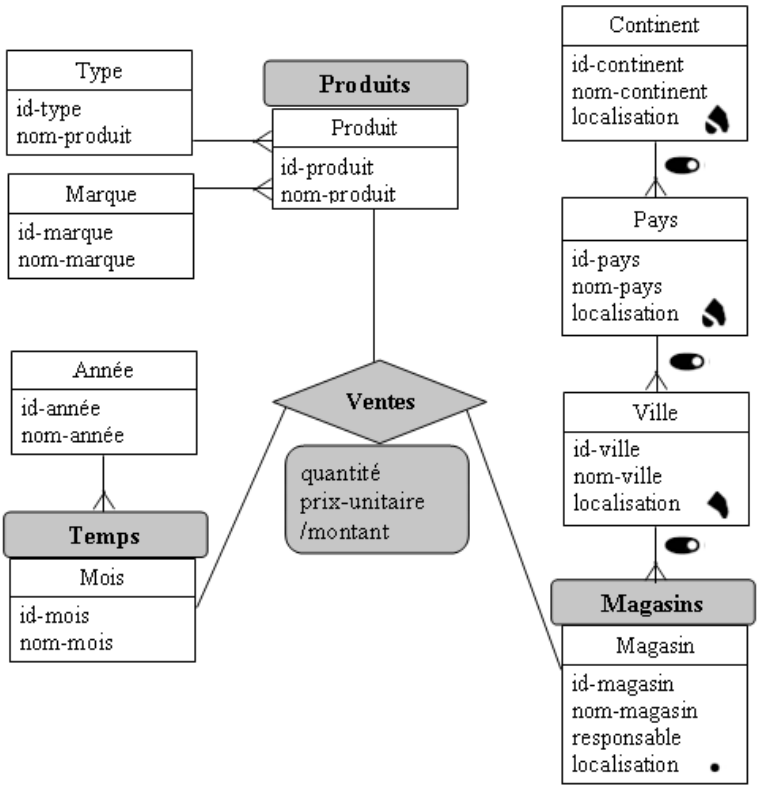


Figure 3.1. Exemple de modèle spatio-multidimensionnel -"Analyse des ventes".

3.2.1Mesure spatiale

La **mesure spatiale** permet la représentation de l'information spatiale relative aux localisations géographiques (i.e. sur la surface de la terre) et non géographiques des faits dans les cellules de l'hypercube. Dans la littérature, plusieurs définitions ont été proposées pour le concept de mesure spatiale comme par exemple :

- Un ensemble de pointeurs vers des objets spatiaux [Han et al., 1998];
- Le résultat d'opérations spatiales topologiques ou métriques [Han et al., 1998; Rivest et al., 2005]; par exemple la surface d'une région géographique, la distance entre régions spatiales, etc.
- Une mesure dont le type de données est spatial (géométrique) [Rivest et al., 2005; Damiani et Spaccapietra, 2006; Malinowski et Zimányi, 2008], par exemple, une zone agricole d'épandage.
- Une mesure complexe ayant des attributs alphanumériques descriptifs et une géométrie [Bimonte et al., 2005].

Dans cette thèse, nous considérons la définition suivante pour la mesure spatiale :

Définition (Mesure spatiale) : *Une **mesure spatiale** est une mesure dont le type de données est spatial (i.e. géométrique) et dont la représentation cartographique représente un intérêt pour l'analyse décisionnelle.*
*Les instances d'une mesure spatiale sont appelées **valeurs de mesure spatiales**.*

Exemples de mesures spatiales : des exemples de mesures spatiales dans le cas d'applications agro-environnementales peuvent être :

- La "zone géographique épandue" pour une application d'analyse des épandages agricoles [Pinet et Schneider, 2010], voir Figure 3.6;
- La "zone géographique de forêts incendiées" pour une application environnementale d'analyse des incendies [Salehi, 2009; Boulil et al., 2010a; Boulil et al., 2010b; Boulil et al., 2011].

Exemple de valeur de mesure spatiale : un exemple de valeur de mesure spatiale pour la mesure spatiale "zone géographique épandue" est montrée en Figure 3.2 (zone blanche hachurée).

Figure 3.2. Exemple de valeur de mesure spatiale - Mesure spatiale "zone géographique épandue". Issue de [Pinet, 2010].

3.2.2Niveau d'agrégation spatial

Un **niveau d'agrégation spatial** est défini comme un niveau qui contient un attribut spatial (i.e. dont le type est géométrique) permettant la représentation cartographique de ses membres de dimension [Han et al., 1998; Bimonte, 2007; Glorio et Trujillo, 2008; Malinowski et Zimányi, 2008; Pinet et Schneider, 2010]. Les niveaux d'agrégation spatiaux sont classés dans [Rivest et al., 2005; Salehi, 2009] selon le type de représentation de leur spatialité en (i) géométriques (la spatialité des membres est représentée par une géométrie) et non géométriques (représentation textuelle).

Dans la suite de ce mémoire, nous adoptons la définition suivante pour le niveau d'agrégation spatial :

Définition (Niveau d'agrégation spatial) : *Un **niveau d'agrégation** est dit spatial s'il contient un attribut spatial (géométrique) qui permet la représentation cartographique de ses membres.*
*Les instances d'un niveau d'agrégation spatial sont appelées **membres de dimension spatiaux**.*

Exemples de niveaux d'agrégation spatiaux : dans l'exemple d'analyse des ventes décrit précédemment (cf. Figure 3.1), les exemples de niveaux d'agrégation spatiaux sont "Magasin", "Ville", "Pays" et "Continent". Tous ces niveaux d'agrégation contiennent un attribut de type géométrique appelé "localisation" qui représente les localisations spatiales de leurs membres de dimension. Par exemple, les localisations spatiales des magasins sont de type *Point* (●); celles des villes sont des régions simples (◆) et celles des pays sont des régions complexes (◆).

Exemples de membres de dimension spatiaux : des exemples de membres de dimension spatiaux pour le niveau d'agrégation spatial "Pays" sont "Canada" et "USA" (cf. Figure 3.3).

3.2.3Dimension spatiale

La **dimension spatiale** permet la représentation de l'information spatiale relative aux localisations géographiques et non géographiques des faits en axes d'analyse. De façon similaire à la mesure spatiale, diverses définitions ont été proposées dans la littérature pour ce concept, comme par exemple :

- Une dimension ayant au moins un niveau d'agrégation spatial [Han et al., 1998; Stefanovic et al., 2000; Fidalgo et al., 2004; Bimonte et al., 2005; Malinowski et Zimányi, 2008];
- Une dimension contenant au moins un niveau spatial qui peut être géométrique ou non géométrique [Bédard et Han, 2009].

Dans la suite de ce mémoire, nous adoptons la définition suivante pour la dimension spatiale :

Définition (Dimension spatiale) : *Une **dimension spatiale** est une dimension qui contient au moins un niveau d'agrégation spatial.*
*Les instances d'une dimension spatiale sont appelées **instances de dimension spatiales**.*

Exemple de dimension spatiale : Un exemple de dimension spatiale dans le cas d'analyse des ventes est la dimension "Magasins" (cf. Figure 3.1)

Exemple d'instance de dimension spatiale : Un exemple d'instance pour la dimension "Magasins" est représenté en Figure 3.3.

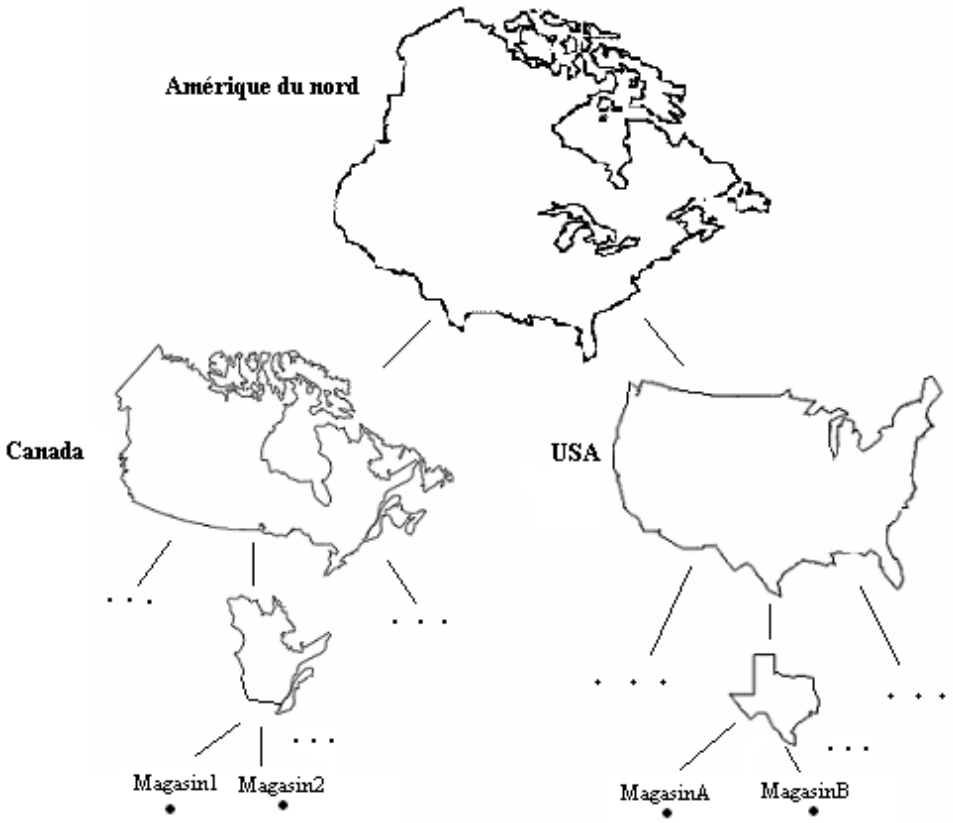


Figure 3.3. Exemple d'instance de dimension spatiale - Dimension "Magasins". Adaptée de [Salehi, 2009].

3.2.4Hiérarchie de dimension spatiale

Une **hiérarchie de dimension spatiale** est définie comme étant une hiérarchie contenant au moins un niveau d'agrégation spatial. De plus, lorsque deux niveaux d'agrégation spatiaux

successifs sont reliés dans une hiérarchie spatiale, les couples de membres impliqués dans la relation doivent vérifier une relation spatiale topologique (e.g. inclusion spatiale, etc.) [Malinowski et Zimányi, 2005]. Par exemple dans le modèle spatio-multidimensionnel des "ventes" représenté en Figure 3.1, les niveaux d'agrégation spatiaux "Ville" et "Pays" sont reliés par la relation topologique *coveredBy* (⊂) [Parent et al., 1999], pour préciser que la région spatiale d'une ville doit être couverte par la région spatiale de son pays.

Par rapport à l'agrégeabilité et d'une façon similaire aux hiérarchies de dimension classiques, les hiérarchies de dimension spatiales sont classées en deux grandes catégories [Malinowski et Zimányi, 2008] : (i) régulières (strictes, onto et covering) et ii) irrégulières (non strictes, non onto et non covering) (cf. Section 2.3.4).

Dans la suite de ce mémoire nous adoptons la définition suivante pour la hiérarchie de dimension spatiale :

Définition (Hiérarchie de dimension spatiale) : Une *hiérarchie de dimension est dite spatiale si elle contient au moins un niveau d'agrégation spatial.*
Les instances d'une hiérarchie de dimension spatiale sont dites ***instances de hiérarchie de dimension spatiales****.*

Exemple de hiérarchie de dimension spatiale : Un exemple de hiérarchie de dimension spatiale dans l'application d'analyse des ventes est la hiérarchie des "Magasins canadiens"

Exemple d'instance hiérarchie de dimension spatiale : Un exemple d'instance pour cette hiérarchie ("Magasins canadiens") est montré en Figure 3.4.

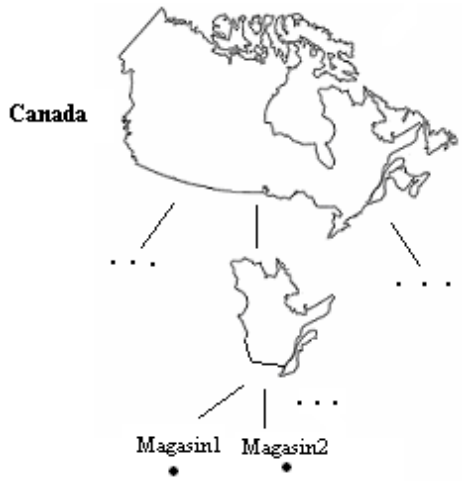


Figure 3.4. Exemple d'instance de hiérarchie spatiale - Hiérarchie "Magasins canadiens". Adaptée de [Salehi, 2009].

3.2.5Fait spatial

Un **fait spatial** est défini comme étant un fait géoréférencé à une position qui est pertinente à analyser; sa localisation est représentée par une mesure spatiale [Damiani et Spaccapietra, 2006]. Dans [Salehi et al., 2010], un fait est dit spatial s'il contient au moins une mesure spatiale ou un niveau d'agrégation spatial. Dans [Malinowski et Zimányi, 2008], un fait spatial est défini comme étant un fait qui requière une jointure spatiale entre deux dimensions spatiales ou plus; autrement dit, un fait qui spécifie une relation topologique (e.g. intersection) entre les niveaux d'agrégation spatiaux feuille de deux dimensions spatiales ou plus.

Dans la suite de cette thèse, nous proposons la définition suivante pour le fait spatial :

Définition (Fait spatial) : *Un **fait spatial** est un fait qui contient au moins une mesure spatiale.*
*Les instances d'un fait spatial sont appelées **instances de fait spatiales**.*

Exemple de fait spatial : Un exemple de fait spatial dans l'application d'analyse des épandages agricoles [Pinet et Schneider, 2010] est donné en Figure 3.5. Il s'agit du fait "Epandages" qui comporte les mesures numériques ("quantité" et "surface") et la mesure spatiale "zone-épandue". Ce modèle est décrit en utilisant la notation graphique proposée dans [Malinowski et Zimányi, 2008].
Exemple d'instance de fait spatial : Un exemple d'instance pour ce fait "Epandages" est par exemple la cellule "(catégorie = 'effluent from farm livestock', producteur = 'producteur 1', commune = 'commune 1', quantité = '33 kg', zone-épandue = '', surface = '30 hectares')"; voir Figure 3.6.

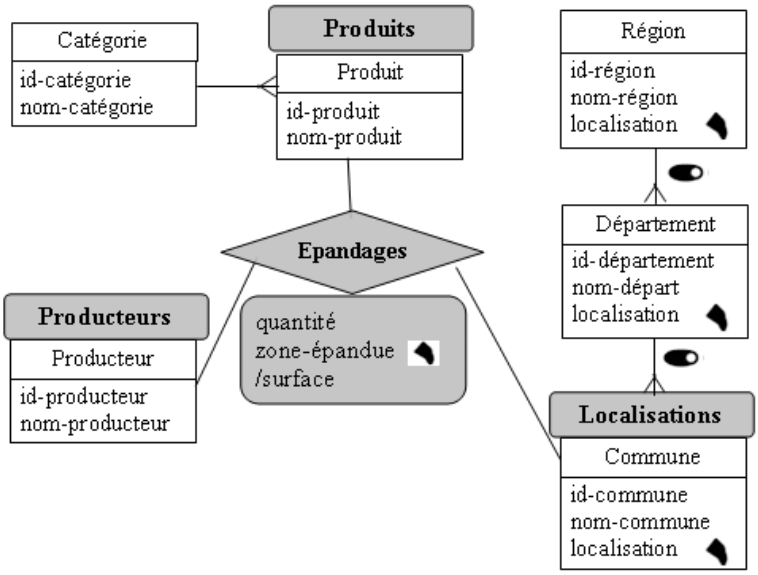


Figure 3.5. Exemple de fait spatial - "Epandages". Adaptée de [Pinet et Schneider, 2010].

3.2.6Hypercube spatial

Un **hypercube spatia**l est défini dans [Salehi et al., 2010] comme un hypercube qui a au moins une mesure ou une dimension spatiale.

Dans nos travaux nous proposons les définitions suivantes pour l'hypercube spatial et l'EDS :

Définition (Hypercube spatial) : *Un **hypercube spatial** est un hypercube qui contient au moins une mesure ou une dimension spatiale.*
*Les instances d'un hypercube spatial sont appelées **cellules spatiales** ou **instances de fait spatiales**.*

Exemples d'hypercubes spatiaux : des exemples sont les hypercubes d'analyse de ventes et d'analyse des épandages représentés précédemment en Figure 3.1 et Figure 3.5 respectivement.

Exemples d'instance d'hypercube spatiale : Un exemple est représenté en Figure 3.6.

Définition (ED Spatial) : *Un ED ou un modèle multidimensionnel pour être qualifié de spatial doit contenir au moins un hypercube spatial.*

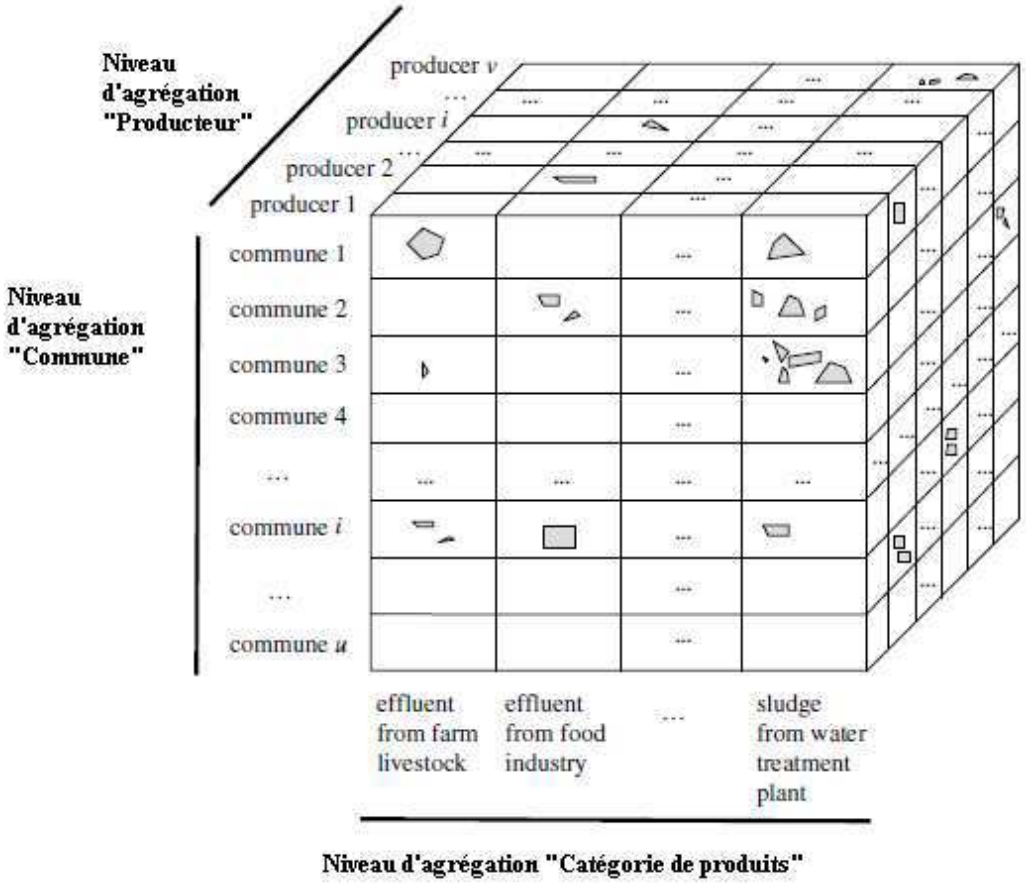


Figure 3.6. Exemple d'instance d'hypercube spatiale - Hypecube "Analyse des Epandages". Adaptée de [Pinet et Schneider, 2010].

3.2.7Opérateurs SOLAP

L'intégration de l'information spatiale en tant que dimension d'analyse mène à la définition d'opérateurs spécifiques au SOLAP pour l'exploration des hypercubes de données spatiales. Ces opérateurs SOLAP étendent/reformulent les opérateurs OLAP traditionnelles.

Les principaux opérateurs SOLAP ont été définis comme suit [Bédard et al., 2007; Ruiz et Times, 2009] :

Roll-up spatial : permet la navigation dans une hiérarchie de dimension d'un niveau d'agrégation spatial vers un autre niveau d'agrégation spatial moins détaillé. Par exemple, le Roll-up spatial du niveau "Région" vers "Département" (cf. Figure 3.5), affiche les cartes des départements et les agrégats des mesures d'épandage pour ces départements.

Drill-down spatial : permet la navigation dans une hiérarchie de dimension d'un niveau d'agrégation spatial vers un autre niveau d'agrégation spatial plus détaillé. Par exemple, l'exécution d'un Drill-down spatial du niveau spatial "Département" vers le niveau spatial "Commune" (cf. Figure 3.5) affiche les cartes des communes de chaque département ainsi que les valeurs des mesures d'épandage pour ces communes.

Slice spatial : cet opérateur permet de sélectionner un sous-ensemble des cellules de l'hypercube spatial en appliquant un prédicat spatial (métrique, d'ordre ou topologique) sur les membres d'une dimension spatiale**.**

Dice spatial : permet de sélectionner un sous-ensemble des données en appliquant des prédicats spatiaux à des membres spatiaux de deux dimensions spatiales ou plus.

3.2.8Agrégation dans le SOLAP

Les mesures spatiales sont agrégées en utilisant des fonctions d'agrégation spatiale (e.g. union, intersection, etc.). L'agrégation spatiale peut retourner un ou un ensemble d'objets spatiaux (i.e. une géométrie simple ou complexe), dépendamment de la connectivité des objets à agréger et de la fonction d'agrégation utilisée. Par exemple, si les objets spatiaux agrégés sont disjoints et la fonction d'agrégation utilisée est l'union spatiale, alors l'agrégation retournera une géométrie complexe.

En général, l'agrégation correcte et agrégeabilité des mesures dans le SOLAP dépend de plusieurs paramètres : (i) de l'additivité de la mesure (stock, flux ou valeur par unité) et (ii) de la structuration thématique de la hiérarchie – type de hiérarchie (stricte ou stricte), paramètres déjà présentés en Section 2.3.4; mais également (iii) du type de fonction d'agrégation (applicable à des données pouvant être additionnées, etc.) [Pedersen et al., 2001], (iv) de la distributivité de la fonction d'agrégation, et (v) des relations spatiales topologiques entre les membres spatiaux (pour une hiérarchie de dimension spatiale) [Pedersen et Tryfona, 2001].

Les fonctions d'agrégation numériques ont été classées en fonction des catégories de données auxquelles elles peuvent s'appliquer en trois types : applicables aux données pouvant être additionnées (Sum, Count, Avg, Min, Max), applicables aux données pouvant être moyennées (Count, Avg, Min, Max), et applicables aux données pouvant être uniquement comptées (Count) [Pedersen et al., 2001].

Par rapport aux types de données auxquels les fonctions d'agrégation SOLAP peuvent être appliquées, les catégories suivantes ont été identifiées dans la littérature :

(a) Fonctions d'agrégation numérique, s'appliquent seulement aux données de type numérique (e.g. Réel); par exemple Sum et Avg.
(b) Fonctions d'agrégation spatiale, s'appliquent seulement aux données de type géométrique (Point, Ligne, Polygone, etc.), par exemple l'union spatiale, le centroid, et l'équipartition [Shekhar et al., 2001; Silva et al., 2008; Ruiz et Times, 2009].
(c) Fonctions d'agrégation booléenne, s'appliquent seulement aux données booléennes, par exemple, Or et And [Golfarelli et al., 1998].
(d) Fonctions d'agrégation textuelle, s'appliquent seulement aux données textuelles (e.g. type String), par exemple, Topic et Top Keywords [Park et al., 2005; Ravat et al., 2007; Ravat et al., 2008].
(e) Fonctions d'agrégation temporelle, s'appliquent aux données de type temporel (e.g. Instant ou Intervalle de dates), par exemple l'union temporelle.
(f) Fonctions d'agrégation génériques : s'appliquent à plusieurs types de données, par exemple Count, Distinct Count et Mode [Ravat et al., 2008].

Suivant leur distributivité, les fonctions d'agrégation SOLAP (numérique, spatiale ou autre) sont classées de manière formelle en trois catégories [Shekhar et al., 2001] : (a) distributives (e.g. Sum, Union spatiale, etc.), (b) algébriques (e.g. Avg, Centroid, etc.) et (c) holistiques

(e.g. Mode, Equipartition, etc.). Les fonctions distributives permettent de réutiliser les agrégats d'un niveau d'agrégation pour calculer des agrégats corrects à un niveau d'agrégation plus large. Les fonctions algébriques sont des expressions algébriques finies de fonctions distributives. Ces fonctions requièrent donc une manipulation supplémentaire pour pouvoir réutiliser correctement des agrégats. Finalement, les fonctions holistiques qui ne sont ni distributives ni algébriques, nécessitent un recalcul total en utilisant les données au niveau le plus détaillé.

3.3 Architecture typique d'un système SOLAP

Les systèmes SOLAP intègrent les techniques d'analyse multidimensionnelle des systèmes OLAP et les fonctionnalités de navigation et de représentation cartographiques des Systèmes d'Information Géographique (SIG) [Rivest et al., 2003]. D'un coté, ces systèmes enrichissent les capacités d'analyse des systèmes OLAP classiques en permettant la représentation cartographique des résultats d'analyse et la navigation spatio-multidimensionnelle dans les hypercubes de données à travers la définition de nouveaux opérateurs de manipulation d'hypercubes spatiaux (e.g. Roll-up spatial, Drill-down spatial, etc.). D'un autre coté, ces systèmes enrichissent les SIG par les fonctions OLAP d'analyse multidimensionnelle de données permettant l'exploration interactive des données à différents niveaux de détail.
L'architecture typique SOLAP reformule/étend les concepts et composants de l'architecture OLAP que nous avons présentée en Section 2.4. L'architecture typique SOLAP est constituée de 4 couches logicielles (cf. Figure 3.7) : ETL spatial, EDS, Serveur SOLAP et Client SOLAP [Rivest et al., 2003]. En amont de cette architecture, on retrouve toutes les sources de données qui alimentent périodiquement le système SOLAP en données analytiques spatiales et non spatiales.

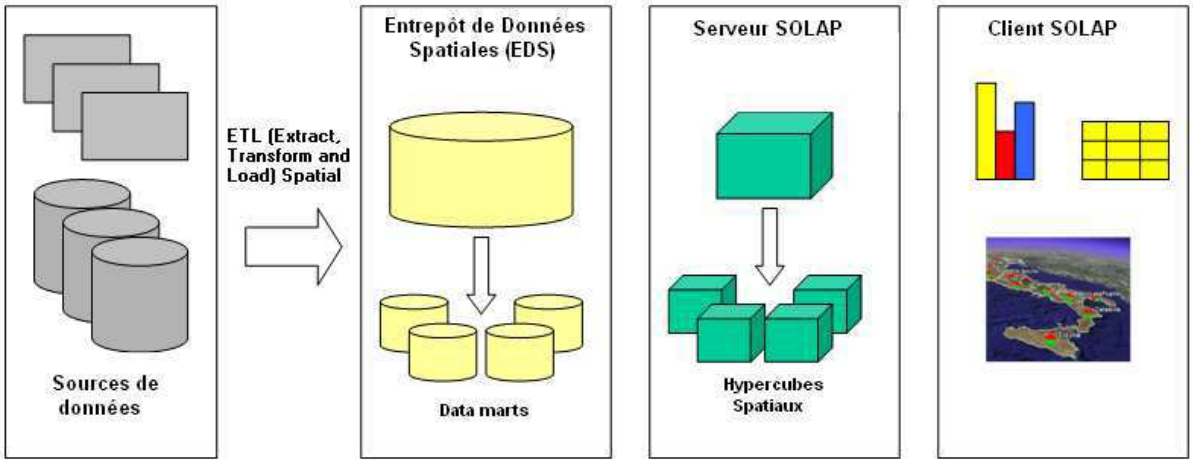


Figure 3.7. Architecture typique d'un système SOLAP.

3.3.1Couche ETL spatial

Les sources de données spatiales présentent d'autres types d'hétérogénéité par rapport aux sources de données alphanumériques (par exemple différents systèmes de référence spatiale, différentes échelles géographiques, etc.). Leur intégration dans les EDS nécessite donc des outils ETL particuliers, ETL spatiaux (par exemple Spatial Data Integrator, cf. Section 3.3.5.1), étendant les ETL classiques par d'autres fonctionnalités dédiées au traitement et à la transformation des données spatiales [Xi-Qian et al., 2004; Bédard et Han, 2009]. Les exemples d'opérations spécifiques aux ETL spatiaux sont la re-projection (i.e. changement de projection cartographique), le géocodage (i.e. conversion de références spatiales textuelles vers des cordonnées spatiales), etc.

3.3.2Couche EDS

Cette couche contient : (i) l'EDS, (ii) le référentiel de métadonnées et éventuellement (iii) plusieurs magasins de données (spatiales et non spatiales). L'EDS historise et centralise toutes les données spatiales et non spatiales nécessaires à l'analyse SOLAP. L'EDS est généralement géré au moyen d'un SGBD Spatial (SGBDS), tels qu'Oracle Spatial ou PostGIS, ce qui permet la gestion de très gros volumes de données spatiales et un bon passage à l'échelle. Cette couche se base sur un modèle spatio-multidimensionnel relationnel (schéma relationnel de l'EDS, cf. Section 2.4.1.1).

Finalement, cette couche utilise aussi des techniques d'optimisation de stockage et de requêtes adaptées aux données spatiales qui étendent les techniques utilisées pour les données alphanumériques dans les ED [Sampaio et al., 2006; Malinowski et Zimányi, 2008; da Silva et al., 2010].

3.3.3Couche Serveur SOLAP

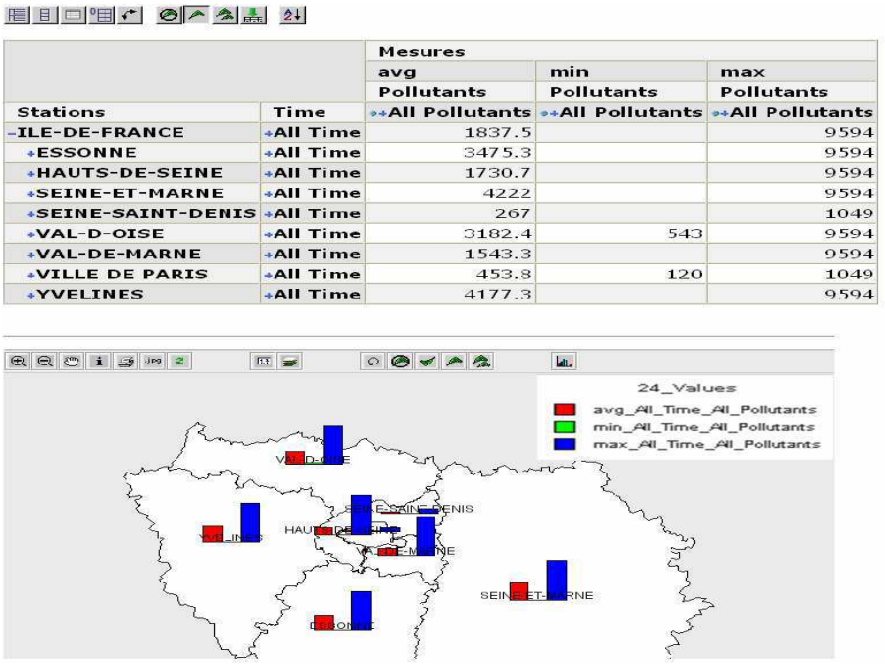
Le serveur SOLAP calcule les hypercubes spatiaux et implémente les opérateurs SOLAP pour la navigation/exploration spatio-multidimensionnelle interactive de ces hypercubes.

Cette couche exploite un fichier de métadonnées particulier, appelé schéma d'analyse SOLAP. Ce schéma définit les hypercubes (spatiaux et non spatiaux), leurs dimensions et hiérarchies, les mesures et leurs règles de calcul ou d'agrégation. Ce fichier, comme pour les serveurs OLAP, spécifie également le mapping de ces structures SOLAP vers les structures relationnelles de l'EDS (cf. Section 2.4.1.1).

3.3.4Couche Client SOLAP

Le client SOLAP permet l’exploration interactive facile et rapide des hypercubes de données spatiaux et non spatiaux en définissant une série d'interfaces utilisateur intuitives. Ces interfaces exploitent divers types d'affichages interactifs (histogrammes, tables de pivot et cartes) qui permettent aux utilisateurs de déclencher les opérateurs SOLAP par quelques clics de souris. Par exemple, les utilisateurs peuvent cliquer sur une région de la carte pour exécuter un Drill-down spatial. Les différents modes d'affichage peuvent être synchronisés lorsque désiré permettant de conserver une continuité perceptuelle nécessaire à la découverte de corrélations [Proulx et al., 2007]. Par exemple, forer dans une carte, fore automatiquement dans le tableau ou le diagramme statistique présentant les données.

Ci-dessous (cf. Figure 3.8) un exemple de visualisation dans un client SOLAP qui représente les quantités moyennes, minimales et maximales de polluants par département de France en utilisant deux types d'affichage (tableau et cartes).



		Mesures		
		avg	min	max
		Pollutants	Pollutants	Pollutants
Stations	Time	All Pollutants	All Pollutants	All Pollutants
ILE-DE-FRANCE	All Time	1837.5		9594
ESSONNE	All Time	3475.3		9594
HAUTS-DE-SEINE	All Time	1730.7		9594
SEINE-ET-MARNE	All Time	4222		9594
SEINE-SAINT-DENIS	All Time	267		1049
VAL-D-OISE	All Time	3182.4	543	9594
VAL-DE-MARNE	All Time	1543.3		9594
VILLE DE PARIS	All Time	453.8	120	1049
YVELINES	All Time	4177.3		9594

Figure 3.8. Exemple de client SOLAP. Issue de [Bimonte et al., 2007].

3.3.5Architecture SOLAP choisie

L'architecture SOLAP que nous avons choisie pour implémenter et valider nos propositions dans le cadre de cette thèse est une architecture SOLAP relationnelle qui est constituée de l'ETL Spatial Data Integrator (SDI, cf. Section 3.3.5.1), du SGBD Oracle 11g (cf. Section 3.3.5.2), du serveur OLAP Mondrian (cf. Section 3.3.5.3) et du client SOLAP JRubik (cf. Section 3.3.5.4).

3.3.5.1 Spatial Data Integrator (SDI)

SDI [Camptocamp, 2012; Talend, 2012a] est un ETL spatial développé par la société CampToCamp et est basé sur l'ETL Talend Open Studio (TOS)[10]. C'est un ETL de type « générateur de code », c'est-à-dire qu'il permet une définition graphique simple du processus ETL puis la génération automatique de l'exécutable correspondant sous forme de programmes Java ou Perl. En plus de la multitude de composants permettant de se connecter à de nombreux types de bases de données et de fichiers en mode lecture/écriture, SDI ajoute la possibilité de réaliser des traitements et transformations spatiaux (e.g. simplification).

3.3.5.2 Oracle Spatial 11g

Oracle Spatial [Oracle, 2012] représente un module/option séparée (avec une licence séparée) du SGBD Oracle. Il fournit un ensemble intégré de fonctions, de procédures et de types de données permettant le stockage, l'accès et l'analyse efficaces des données spatiales. Oracle Spatial comprend (i) un schéma (MDSYS) qui décrit le stockage, la syntaxe et la sémantique des types de données spatiales supportés, (ii) des mécanismes d'indexation spatiale; (iii) un ensemble d'opérateurs et de fonctions qui permettent des analyses spatiales; et (iv) des outils d'administration. Oracle Spatial implémente également les relations spatiales topologiques du modèle des 9 intersections (modèle 9IM) d'Egenhofer [Egenhofer et Herring, 1994] et permet l'implémentation de contraintes d'intégrité sous formes de triggers, de vues et de requêtes SQL.

[10] http://www.talend.com/products/open-studio-di.php

3.3.5.3 Mondrian

Mondrian [Pentaho, 2012a] est un serveur OLAP qui permet l'analyse en ligne de grandes masses de données en se connectant à une base de données relationnelle qui stocke les données multidimensionnelles au niveau de granularité le plus fin. Mondrian est développé par la société Pentaho sous forme d'une application Java open source, extensible et supportant la connexion aux principaux SGBDR existants (Oracle, PostgreSQL, MySQL, etc.). Mondrian supporte le langage MDX qui est utilisé dans nos travaux pour l'implémentation des CI de requêtes et des indicateurs d'analyse complexes (cf. Sections 5.5 et 6.9.2). MDX (*MultiDimensional eXpressions language*) est le langage le plus utilisé pour l'interrogation et la manipulation des bases de données multidimensionnelles et des hypercubes OLAP [Microsoft, 2012]. Ce langage permet l'expression facile de requêtes OLAP complexes, en offrant un ensemble très riche de fonctions analytiques et d'opérateurs OLAP (Drill-down, etc.). Ce langage est adopté par la plupart des fournisseurs de solutions OLAP (e.g. Oracle, Microsoft, etc.) et est devenu un de facto standard pour les systèmes OLAP.

Il est important de noter que dans ce travail de thèse, nous avons utilisé un serveur OLAP et non pas un serveur SOLAP car les fonctionnalités du serveur Mondrian sont suffisantes pour l'implémentation des CI SOLAP (cf. Chapitre 10).

3.3.5.4 JRubik

JRubik [JRubik, 2012] est un client SOLAP développé en Java en se basant sur les composants du client OLAP JPivot[11]. JRubik se connecte au serveur OLAP Mondrian et fournit l'interface graphique pour déclencher les opérateurs OLAP (Roll-up, etc.) sur les hypercubes OLAP et permet de visualiser les résultats des requêtes sous différents types d'affichage interactifs (tables de pivot, histogrammes, etc.). Cet outil permet également une représentation cartographique des résultats de requêtes spatiales en utilisant le format SVG (*Scalable Vector Graphics*).

Ci-dessous (cf. Figure 3.9), un exemple de visualisation avec JRubik représentant les nombres d'habitants par municipalité.

[11] http://jpivot.sourceforge.net

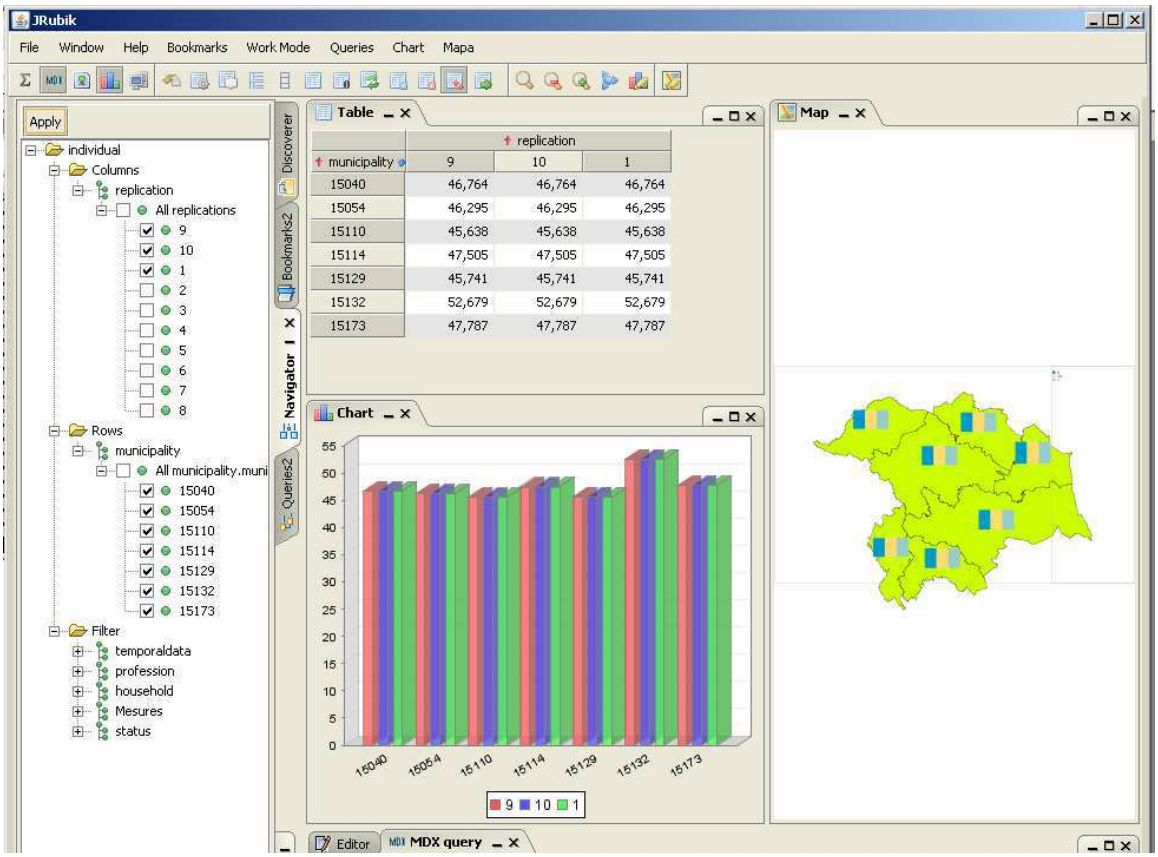


Figure 3.9. Exemple de visualisation avec le client SOLAP JRubik. Issue de [Mahboubi et al., 2011].

3.4 Modélisation spatio-multidimensionnelle

La recherche en modélisation spatio-multidimensionnelle est relativement récente. Depuis les travaux pionniers de [Bédard, 1997], quelques modèles ont été proposés dans la littérature visant à étendre le modèle multidimensionnel classique par de nouveaux concepts spatio-multidimensionnels. De façon générale, les modèles proposés au niveau conceptuel sont soit des extensions de langages de modélisation standards comme UML ou ER (cf. Section 3.4.1); ou des formalismes ad hoc (cf. Section 3.4.2). Aux niveaux logique et physique, les modèles proposés visent à produire un support spatial pour la définition des concepts spécifiques au modèle spatio-multidimensionnel (dimensions, mesures et agrégations spatiales) [Han et al., 1998; Shekhar et al., 2001; Sampaio et al., 2006; Malinowski et Zimányi, 2008; da Silva et al., 2010; do Nascimento Fidalgo et al., 2010]. Ces modèles logiques et physiques sont généralement des extensions du modèle en étoile de [Kimball et Ross, 2002]; pour plus de détails sur ces aspects veuillez vous référer à [Malinowski et Zimányi, 2008; Salehi, 2009; da Silva et al., 2010].

Dans la suite de cette section, nous présentons un état de l'art des modèles spatio-multidimensionnels conceptuels existants en détaillant les modèles UML et ER (cf. Section 3.4.1) et en décrivant brièvement les modèles ad hoc (cf. Section 3.4.2). L'évaluation suivant un ensemble de critères existants [Abelló et al., 2006; Viswanathan et Schneider, 2011] et nouveaux des modèles conceptuels UML sera présentée en Section 6.6.3. La façon dont ces modèles définissent les contraintes d'intégrité SOLAP sera aussi décrite et discutée en détail dans le chapitre 4 (cf. Sections 4.4, 4.5, 4.6 et 4.7).

3.4.1Modèles basés sur des standards

3.4.1.1 Modèles UML

Les principaux modèles UML sont proposés dans les travaux [Glorio et Trujillo, 2008; Pinet et al., 2010; Pinet et Schneider, 2010].

[Glorio et Trujillo, 2008] étend le profil UML présenté dans [Lujan-Mora et al., 2006] (cf. Section 2.5.1) par deux nouveaux stéréotypes : (i) *SpatialMeasure* pour représenter les mesures spatiales et (ii) *SpatialLevel* pour représenter les niveaux d'agrégation spatiaux. Le modèle est vu comme un ensemble d'étoiles (ou d'hypercubes) pouvant partager des dimensions, des hiérarchies et des niveaux d'agrégation. Chaque étoile (stéréotype *StarPackage*) contient un ensemble de dimensions (*DimensionPackage*) et un paquetage de faits (*FactPackage*) qui à son tour contient un fait (*Fact*) et zéro ou plusieurs faits dégénérés (*DegenerateFact*[12]). Chaque fait (*Fact*) contient un ensemble de mesures ou d'attributs de fait (*FactAttribute*) et un ensemble de dimensions dégénérées (*DegenerateDimension*[13]); et chaque niveau d'agrégation (*Base*) peut contenir différents types d'attributs : un identifiant (*OID*), un attribut particulier utilisé pour visualiser ses membres dans les outils OLAP (*Descriptor*), un attribut géométrique s'il est spatial et un certain nombre d'autres attributs descriptifs (*DimensionAttribute*). Les niveaux d'agrégation sont reliés par des relations stéréotypées *Rolls-upTo* (spécialisations d'associations UML) pour former les hiérarchies de dimension. Ces différents stéréotypes sont formalisés comme des spécialisations de métaclasses UML (e.g. le stéréotype *Base* spécialise la métaclasse UML *class*) sur lesquelles sont définies des contraintes OCL pour éviter leur mauvaise utilisation par les concepteurs.
Un exemple de modèle spatio-multidimensionnel conceptuel, défini avec ce profil et représentant l'analyse des nombres d'oliviers selon le temps et la localisation spatiale, est représenté en Figure 3.10.

[12] Le concept de fait dégénéré est introduit pour modéliser les relations fait-dimension non strictes qui possèdent des attributs.
[13] Le concept de dimension dégénérée est utilisé pour modéliser les dimensions composées d'un seul attribut.

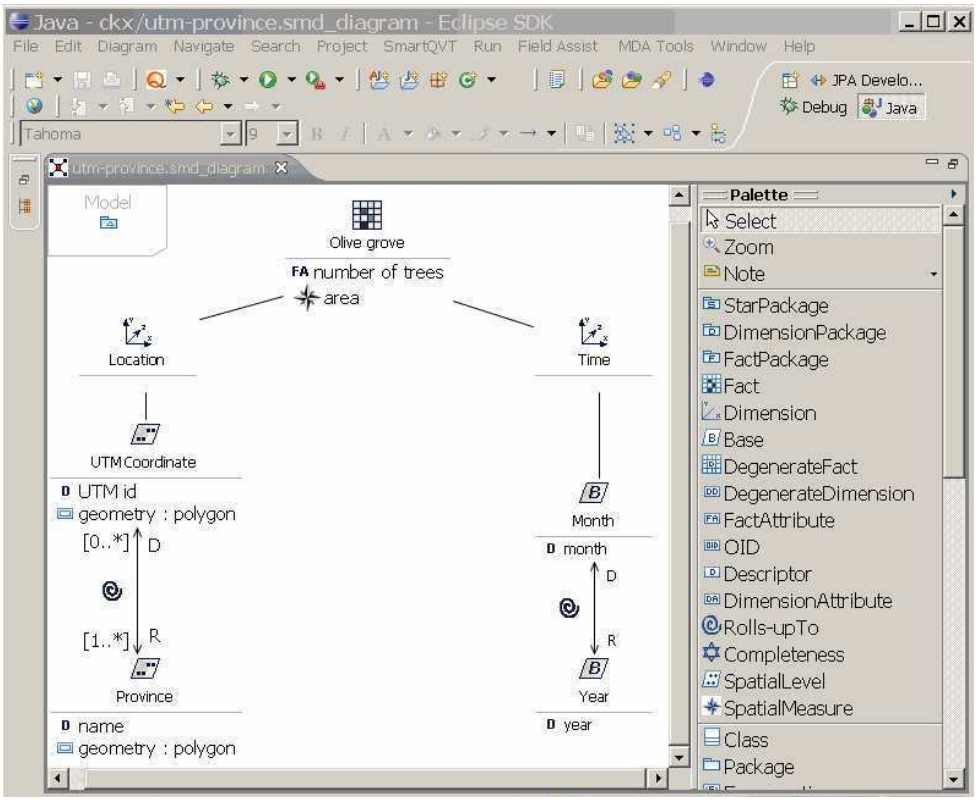


Figure 3.10. Exemple de modèle spatio-multidimensionnel conceptuel spécifié avec le profil UML de [Glorio et Trujillo, 2008]. Issue de [Glorio et Trujillo, 2008].

L'usage des paquetages UML dans ce profil permet d'organiser les éléments multidimensionnels selon trois niveaux de détail, simplifiant ainsi la lecture des modèles complexes. Ce profil ainsi qu'un ensemble de relations QVT[14] permettant la génération automatique du modèle logique sont implémentés par un module dans l'EDI (Environnement de Développement Intégré) Eclipse[15] (cf. Figure 3.10).

De la même façon, [Pinet et Schneider, 2010] propose un profil UML qui étend le modèle UML présenté dans [Pinet et Schneider, 2009] (cf. Section 2.5.1) par deux concepts spatiaux : *mesure spatiale* (attribut UML dont le type de données est géométrique) et *classe identifiée spatiale* (une classe identifiée ayant un attribut géométrique) pour représenter les niveaux d'agrégation et faits spatiaux. Ce modèle unifie les représentations conceptuelles (i) des faits et des niveaux d'agrégation sous forme de *classes identifiées (spatiales)*, ainsi que (ii) celles des relations d'agrégation et des relations fait-dimension sous forme *d'associations d'agrégation*. Ces différents stéréotypes sont formalisés également comme des spécialisations de métaclasses UML (e.g. *l'association d'agrégation* spécialise la métaclasse UML *association*).
Ce modèle est enrichi par un ensemble de contraintes définies en OCL et UML pour garantir les conditions structurelles d'agrégeabilité (i.e. disjonction et complétude – voir Sections 2.3.4, 4.2.2 et 4.5.1) dans les hiérarchies de dimension. Ces contraintes sont également utilisées pour distinguer/définir les différents types de hiérarchies (e.g. onto ou non onto, etc.).

[14] QVT (*Query/View/Transformation lanuage*) est le standard OMG pour l'expression de mappings entre modèles, c'est l'une des spécifications les plus importantes de l'approche dirigée par les modèles MDA (*Model Driven Architecture*) [OMG, 2011a].
[15] http://www.eclipse.org

A noter également que ce travail est le seul qui utilise OCL pour modéliser les contraintes d'intégrité pour vérifier la consistance des données intégrées dans les EDS.
Un exemple de modèle conceptuel représentant l'analyse spatio-multidimensionnelle des épandages agricoles, qui est spécifié avec ce profil, est montré en Figure 3.11.

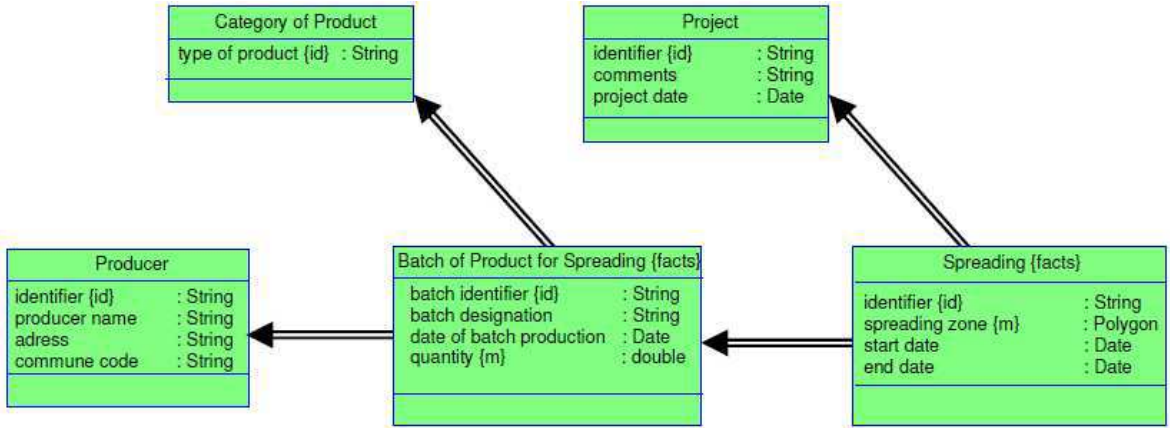


Figure 3.11. Exemple de modèle spatio-multidimensionnel conceptuel spécifié avec le profil UML de [Pinet et Schneider, 2010]. Issue de [Pinet et Schneider, 2010].

Ce profil n'a pas été implémenté.

Enfin, [Pinet et al., 2010] propose un profil UML, appelé profil SOLAP, qui représente (i) les hypercubes (spatiaux et non spatiaux) par des spécialisations de paquetages UML, (ii) les niveaux d'agrégation par des spécialisations de classes et (iii) les mesures comme des attributs UML regroupés dans une classe stéréotypée *Measure*. Les niveaux d'agrégation sont reliés par des relations d'agrégation UML pour former les hiérarchies. Les dimensions sont de plus spécialisées en trois types, (i) thématiques (ii) spatiales et (iii) temporelles. Des pictogrammes différents sont associés aux différents stéréotypes pour améliorer la lisibilité du modèle. Un exemple de modèle conceptuel spécifié avec ce profil et représentant l'analyse spatio-multidimensionnelle du transfert de pesticides dans des parcelles est représenté en Figure 3.12.

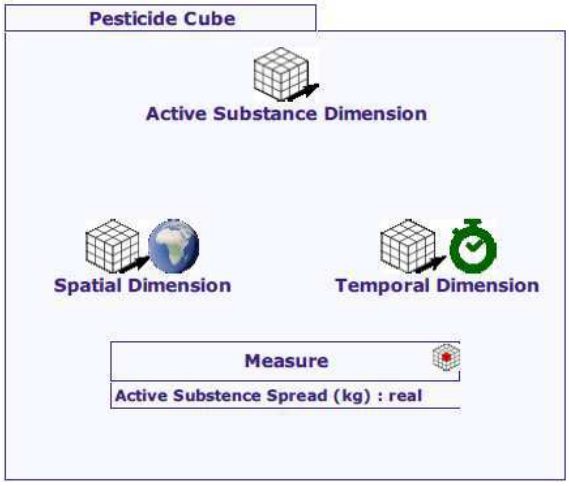


Figure 3.12. Exemple de modèle spatio-multidimensionnel conceptuel spécifié avec le profil UML de [Pinet et al., 2010]. Issue de [Pinet et al., 2010].

Ce profil été implémenté sous forme d'un module dans l'AGL Objecteering[16]. Un module permettant la transformation des instances de ce profil vers une représentation logique relationnelle en étoile a été également développé.

3.4.1.2 Modèles ER

A notre connaissance, seuls les travaux [Malinowski et Zimányi, 2004b, 2005; Malinowski et Zimányi, 2008] proposent un modèle spatio-multidimensionnel ER appelé Spatial MultiDimER. Cette extension spatiale du modèle multidimensionnel MultiDimER présenté dans [Malinowski et Zimányi, 2004a; Malinowski et Zimányi, 2006] (cf. Section 2.5.1) fournit un support spatial pour la plupart des éléments multidimensionnels (mesures, faits, niveaux d'agrégation, hiérarchies et dimensions). Le modèle (cf. Figure 3.13) est vu comme un ensemble fini de dimensions (*Dimension*) et de relations de fait (*Fact relationship*). Les dimensions sont composées d'ensembles de hiérarchies (*Hierarchy*), qui sont à leur tour composées d'ensembles de niveaux d'agrégation reliés (*Level*). Un(e) dimension/Hiérarchie/Niveau d'agrégation est dit(e) spatial(e) si elle (il) contient au moins un(e) hiérarchie spatiale/niveau d'agrégation spatial/attribut géométrique représentant les localisations spatiales de ses membres respectivement. Au sein d'une hiérarchie spatiale, deux niveaux d'agrégation spatiaux successifs sont reliés par une relation topologique (e.g. *coveredBy*).

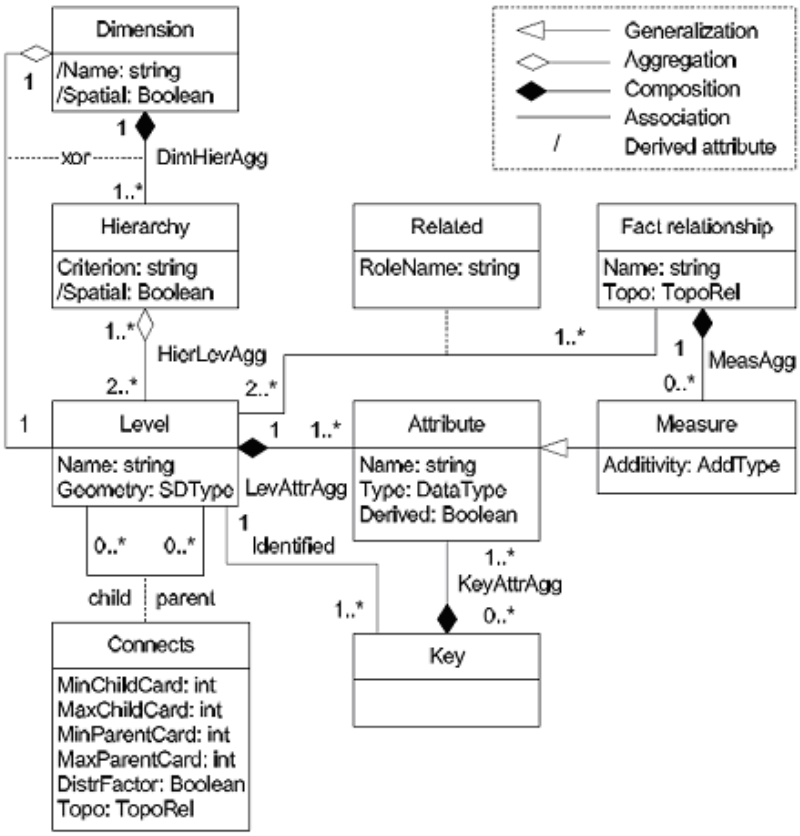


Figure 3.13. Métamodèle du modèle Spatial MultiDimER [Malinowski et Zimányi, 2008].

La mesure spatiale (*Spatial Measure*) est définie comme une mesure dont le type de données est géométrique. Le fait (*Fact relationship*) est formalisé comme une relation n-aire entre les niveaux d'agrégation feuille des dimensions, il est dit spatial s'il définit une relation topologique entre deux niveaux d'agrégation spatiaux feuille ou plus. Enfin ce modèle définit

[16] http://www.objecteering.com

la notation graphique associée à chacun des éléments proposés (cf. Figures 3.1 et 3.5). En particulier, les caractéristiques spatiales telles que les géométries de niveaux d'agrégation spatiaux et les relations topologiques sont représentées en utilisant les pictogrammes du modèle conceptuel spatio-temporel MADS[17]. Des exemples de modèles spécifiés avec ce formalisme sont représentés précédemment aux figures 3.1 et 3.5.

3.4.2Modèles ad hoc

Les modèles ad hoc nécessitent plus d'efforts de modélisation et de compréhension. Dans le contexte SOLAP, tous les modèles conceptuels ad hoc existants ne fournissent pas de notations graphiques ce qui rend leur utilisation très difficile. Les principaux modèles ad hoc sont proposés dans [Jensen et al., 2004; Ahmed et Miquel, 2005; Bimonte et al., 2005; Damiani et Spaccapietra, 2006; Salehi et al., 2010; Viswanathan et Schneider, 2010]. La plupart de ces propositions se focalisent sur des problématiques particulières de la modélisation spatio-multidimensionnelle : traitement du problème de comptage en double des valeurs de mesure dans les hiérarchies non strictes [Jensen et al., 2004]; prise en compte de la dépendance entre les agrégations numérique et spatiale [Bimonte et al., 2005], représentation de mesures spatiales à différentes échelles géographiques [Damiani et Spaccapietra, 2006], représentation des champs continus [Ahmed et Miquel, 2005].

Dans le contexte des applications de services géodépendants, [Jensen et al., 2004] étendent le modèle multidimensionnel et l'algèbre OLAP proposés dans [Pedersen et al., 2001] pour représenter et manipuler les hiérarchies non strictes; ils proposent également une méthode pour résoudre le problème du comptage en double des mesures induit par ce type de hiérarchies (cf. Section 4.5.1). La particularité de ce modèle est que les faits peuvent être reliés à des niveaux d'agrégation non feuille permettant la représentation de mesures à différents niveaux de granularité. Bien que ce travail concerne des applications géolocalisées, aucune définition et aucune notation graphique n'est proposée pour les concepts particuliers de la modélisation spatio-multidimensionnelle.

[Bimonte et al., 2005] proposent un model multidimensionnel formel, appelé *GeoCube*, représentant les mesures spatiales (géographiques) comme des entités complexes composées d'attributs descriptifs et d'une géométrie. Le modèle définit le concept de *schéma d'entité* (un tuple d'attributs) pour une modélisation symétrique des mesures et des niveaux d'agrégation; cette symétrie entre mesures et dimensions se traduit également par l'organisation des mesures en hiérarchies. En utilisant ce concept de *schéma d'entité*, les auteurs formalisent les concepts de mesure spatiale, de niveau d'agrégation spatial, de hiérarchie de dimension, de mode d'agrégation et d'hypercube. Le mode d'agrégation (ensemble de fonctions d'agrégation liant des mesures détaillées à des mesures agrégées) permet de représenter la façon dont les mesures spatiales sont agrégées.

[Damiani et Spaccapietra, 2006] proposent le modèle *MuSD* (*Multi-granular Spatial Data warehouse model*) permettant la représentation de mesures spatiales à différentes échelles géographiques. De façon similaire aux dimensions, les mesures spatiales sont organisées en plusieurs niveaux hiérarchiques représentant chacun une échelle géographique; et à chaque échelle est associée une représentation géométrique spécifique.

[Viswanathan et Schneider, 2010] propose un modèle appelé *BigCube*. Ce modèle définit un ensemble de types de données pour la modélisation de données multidimensionnelles (e.g. type de catégorie (niveau d'agrégation), type de hiérarchie, etc.). Excepté la considération

[17] MADS est un modèle conceptuel très connu dans le domaine des bases de donnés spatio-temporelles [Parent et al., 1999].

des types de données spatiales pour les mesures et les niveaux d'agrégation, *BigCube* ne propose ni définitions ni notations graphiques pour les concepts particuliers de la modélisation spatio-multidimensionnelle.

[Salehi et al., 2010] propose un modèle conceptuel pour les hypercubes de données spatiales. En plus des concepts SOLAP définis dans [Rivest et al., 2005; Bédard et al., 2006] (cf. Section 3.2), ce modèle formalise les concepts suivants et leurs extensions spatiales aux deux niveaux instance et schéma : niveau d'agrégation, hypercube, hypercellule et fait. Le niveau d'agrégation spatial est défini comme un niveau d'agrégation ayant au moins un attribut spatial. L' hypercellule est définie comme le modèle commun à un certain nombre d'instances de fait; elle est spécifiée par une combinaison de niveaux d'agrégation et de mesures; elle est dite spatiale si elle inclut au moins un niveau d'agrégation ou une mesure spatiale. Le fait spatial est définit comme l'instance d'une hypercellule spatial. Les auteurs suggèrent que ce type d'hypercube doit offrir le support géométrique nécessaire pour permettre la représentation et la navigation cartographiques; en d'autres termes, il doit contenir des données géométriques en mesures ou en dimensions.

Enfin, l'évaluation de ces modèles selon un ensemble de critères concernant leur support à la définition des CI SOLAP est montrée en Section 6.6.

3.5 Conclusion

Dans ce chapitre nous avons tenté de définir les principaux concepts des systèmes SOLAP en nous basant sur les définitions du chapitre 2. Les systèmes SOLAP sont alors définis comme des extensions des systèmes OLAP permettant la prise en compte la composante spatiale de l'information dans l'analyse multidimensionnelle. Le modèle spatio-multidimensionnel sur le quel se base ces systèmes est également défini comme l'extension du modèle OLAP. Les définitions adoptées pour les concepts de *mesure spatiale*, de *niveau d'agrégation spatial*, de *dimension spatiale*, de *hiérarchie de dimension spatiale*, de *fait spatial*, et d'*hypercube spatial*, au niveau schéma et instance, sont présentées, situées par rapport aux principales définitions existantes, et illustrées en utilisant des exemples repris de la littérature. Pour définir la spatialité de ces concepts, nous avons considéré le point de vue de la communauté informatique qui exige une représentation géométrique des données spatiales. Ce choix est fait dans le but de faciliter l'expression avec Spatial OCL (cf. Chapitres 4 et 7) et la vérification dans la base de données des contraintes d'intégrité spatiales.

L'agrégation dans le SOLAP a été ensuite décrite en soulignant la complexité de l'agrégation des mesures spatiales dont le résultat dépend de la connectivité spatiale des objets et de la distributivité des fonctions d'agrégation utilisées. Par rapport à ce dernier point (distributivité), les fonctions d'agrégation SOLAP ont été classées en trois catégories : distributives, algébriques, et holistiques. Egalement, une classification de ces fonctions par rapport aux types de données auxquels elles peuvent être appliquées est présentée. L'architecture typique des systèmes SOLAP est décrite (elle se compose de quatre couches logicielles, ETL Spatial, EDS, Serveur SOLAP et Client SOLAP), avant qu'elle ne soit présentée l'architecture effectivement choisie en termes d'outils utilisés. Ce chapitre se termine par un état de l'art décrivant les principaux modèles spatio-multidimensionnels conceptuels en détaillant les modèles basés sur les langages standards UML et ER. Le choix de détailler ces modèles est détecté par notre troisième objectif de thèse, qui consiste à proposer des solutions conceptuelles basées sur des standards (cf. Section 1.3).

Ce chapitre définit les principaux concepts liés à la modélisation, l'agrégation, l'exploration (opérateurs SOLAP) et l'architecture des systèmes SOLAP (qui constituent le contexte de nos travaux de thèse), qui sont nécessaires à la compréhension des composantes de la qualité

d'analyse SOLAP et à l'évaluation des travaux existants sur la définition des CI SOLAP décrites au chapitre suivant.

4 Chapitre 4 : Qualité d'analyse dans les systèmes Spatial OLAP

4.1 Introduction

La qualité est une notion très importante qui est recherchée dans toute production de biens ou de services et aussi dans les systèmes informatiques et logiciels [ISO, 2000; ISO/IEC, 2001; Daniel L, 2005]. Il s'agit d'un facteur capital qui conditionne la réussite et le bon fonctionnement du système ou du produit. La norme ISO 9000 [ISO, 2000] définit la qualité comme : « *l'ensemble des propriétés ou caractéristiques d'un produit ou service qui lui confère l'aptitude à satisfaire des besoins exprimés ou implicites des utilisateurs* ». La qualité est donc une notion relative qui dépend des besoins des utilisateurs : un même produit peut donner lieu à des évaluations de qualité différentes selon des besoins distincts.

La qualité d'analyse dans les systèmes (S)OLAP est une problématique de recherche très importante car ces systèmes sont utilisés pour supporter le processus de prise de décision au sein de divers types d'organisations [Rizzi et al., 2006; Salehi, 2009].

La qualité d'analyse (S)OLAP telle que nous la définissons dans [Boulil et al., 2012d, b] peut être déclinée essentiellement en trois types de qualité : (i) qualité de données, (ii) qualité d'agrégation et (ii) qualité d'exploration. Selon la norme ISO 9113:2002, la qualité de données (spatiales) dépend principalement de trois facteurs : la cohérence logique (absence de contradictions dans les données en rapport aux règles logiques définies dans leur spécification), la précision (données présentant le moins d'écarts possibles avec la réalité) et l'exhaustivité (présence de toutes les données nécessaires à la prise de décision). La qualité d'agrégation, investiguée dans la littérature sous le terme d'agrégeabilité, définit des conditions pour garantir une agrégation correcte et sensée des mesures le long des hiérarchies de dimension [Lenz et Shoshani, 1997]. La qualité d'exploration vise à éviter les problèmes d'interprétation liés à des requêtes insensées ou invalides (e.g. "les ventes en URSS en 2010") [Levesque et al., 2007; Boulil et al., 2012d, b].

Dans la suite de ce chapitre nous allons détailler les composantes de chacune de ces trois qualités (cf. Section 4.2), définir les CI et discuter les différents langages de leur spécification (cf. Section 4.3), présenter les travaux existants sur les qualités de données, d'agrégation et d'exploration dans les ED et EDS (cf. Sections 4.4, 4.5 et 4.6 respectivement); et enfin en Section 4.7, nous présentons une évaluation détaillée des travaux les plus importants en considérant un ensemble de critères.

4.2 Qualité (S)OLAP

La qualité d'analyse (S)OLAP dépend de trois types de qualité : la qualité des données (spatiales) entreposées (cf. Section 4.2.1), la façon dont les mesures (spatiales) sont agrégées (cf. Section 4.2.2) et la façon dont les décideurs explorent les données agrégées (cf. Section 4.2.3) [Levesque et al., 2007; Boulil et al., 2012d, b].

4.2.1Qualité de données

La qualité des données (spatiales) est une notion relative et très complexe qui nécessite le recours à plusieurs composantes (ou critères) pour être évaluée [ISO/TC 211, 2002; Devillers et Jeansoulin, 2005; Devillers et al., 2007]. Divers travaux de recherche ont proposé divers critères quantitatifs et qualitatifs pour déterminer la qualité d'un jeu de données (spatiales).

Ces travaux proposent deux définitions pour la qualité des données en considérant deux points de vue, celui des producteurs de données et celui des utilisateurs de données :

(i) Qualité interne : La qualité interne désigne l'absence d'erreurs/imperfections dans les données en rapport à des règles précises [ISO/IEC, 2001, 2003]. Cette qualité qualifiée de qualité intrinsèque se mesure au travers de critères quantitatifs et qualitatifs comme la précision, l'exhaustivité, la cohérence logique, la généalogie, etc.

(ii) Qualité externe : La qualité externe correspond au niveau d'adéquation existant entre les données et les besoins des utilisateurs dans un contexte donné [ISO/IEC, 2001, 2004]. Elle exprime la capacité des données à répondre à un usage particulier. C'est donc une qualité dépendante des besoins des utilisateurs qui peut être qualifiée de qualité d'usage relative.

Comme notre but est d'avoir une bonne qualité de données qui soit applicable à différentes utilisations et utilisateurs, nous allons dans ce qui suit décrire les principales composantes de la qualité interne, dans laquelle se situent nos travaux. Dans la littérature, différentes définitions, terminologies, hiérarchisations des composantes de cette qualité ont été proposées. Il y également de nombreux efforts de normalisation de ces concepts et définitions.

Dans la suite de cette section nous allons présenter la qualité interne des données (spatiales) telle que définie par la norme ISO 19113:2002 [ISO/TC 211, 2002], car cette norme fournit des définitions précises et concises pour décrire les concepts de la qualité. Dans cette norme, la qualité interne des données spatiales est décrite en utilisant deux types de critères, quantitatifs (cf. Section 4.2.1.1) et qualitatifs (cf. Section 4.2.1.2) qui sont applicables à un jeu de données.

4.2.1.1 Critères quantitatifs

Les critères quantitatifs (éléments et leurs sous éléments) sont décrits comme suit :

4.2.1.1.1 Exhaustivité (*Completeness*)

L'exhaustivité d'un jeu de données désigne la couverture avec laquelle l'univers du discours[18] (*universe of discourse*) est représenté dans le jeu de données [Batini et Scannapieca, 2006]. L'exhaustivité contrôle le manque et le surcroît d'objets, leurs attributs et relations dans un jeu de données [ISO/TC 211, 2002]. Ses sous éléments sont donc :

(a) Omission : manque ou absence de données utiles dans le jeu de données;
(b) Commission : présence de surplus de données inutiles dans le jeu de données.

Selon [Devillers et Jeansoulin, 2005], l'exhaustivité peut être de deux catégories : (i) exhaustivité des données (erreurs d'omission ou commission dans les instances de classes, d'attributs ou de relations); et (ii) exhaustivité du modèle qui mesure le degré de fidélité du modèle de données à l'univers du discours. Le contrôle de cette dernière consiste à voir si tous les objets, leurs attributs et liens de l'univers de discours sont bien représenté dans le modèle.

4.2.1.1.2 Précision de position (*Positional accuracy*)

La précision ou l'exactitude mesure les écarts entre les données mesurées et les données réelles. Elle est définie comme la proximité d'une valeur mesurée de la valeur réelle qu'elle

[18] L'univers du discours (appelé aussi terrain nominal) est défini comme une abstraction/vue du monde réel ou hypothétique qui inclut tout ce qui présente un intérêt pour l'application [ISO/TC 211, 2002].

représente [Batini et Scannapieca, 2006]. La précision de position [ISO/TC 211, 2002] décrit l'exactitude du positionnement spatial des objets spatiaux; ses sous éléments sont :

(a) Précision absolue : proximité des valeurs des coordonnées mesurées des valeurs vraies ou acceptées comme telles;

(b) Précision relative : proximité des positions relatives des objets de leurs positions relatives vraies ou acceptées comme telles;

(c) Précision matricielle : concordance des valeurs de position dans la grille avec les valeurs vraies ou acceptées comme telles.

Cette précision dépend directement des instruments de mesure et moyens d'acquisition et de traitement de l'information spatiale [Devillers et Jeansoulin, 2005]. La seule façon de la mesurer est donc de comparer le jeu de données, soit à autre jeu de données de meilleure qualité, ou soit de procéder par sondage ou échantillonnage (par exemple à l'aide de capteurs GPS) [Devillers et Jeansoulin, 2005].

4.2.1.1.3 Précision temporelle (*Temporal accuracy*)

La précision temporelle se rapporte à la précision des références temporelles des données. Elle est définie comme l'exactitude des attributs et relations temporelles des entités; ses sous-éléments sont [ISO/TC 211, 2002] :

(a) Précision des mesures temporelles : proximité des valeurs des références temporelles des valeurs vraies ou acceptées comme telles;

(b) Cohérence temporelle : exactitude de l'ordre des événements dans le temps;

(c) Validité temporelle : validité des références temporelles par rapport aux formats et système de référence temporelle.

4.2.1.1.4 Précision thématique (*Thematic accuracy*)

La précision thématique, appelée également précision sémantique ou d'attributs, se rapporte à la précision des données alphanumériques attributaires (non temporels et non spatiales), de leurs classifications et relations. Elle est définie comme (i) l'exactitude (*accuracy*) des attributs quantitatifs et (ii) la justesse (*correctness*) des attributs non quantitatifs (qualitatifs), des classifications d'entités et des relations entre entités [ISO/TC 211, 2002]. Ses sous éléments sont :

(a) Justesse de classification : comparaison des classes assignées aux entités ou leurs attributs à leurs classes dans l'univers du discours;

(b) Précision des attributs non quantitatifs : justesse des attributs non quantitatifs;

(c) Précision des attributs quantitatifs : exactitude des attributs quantitatifs.

*Nos travaux de thèse concernent **la cohérence logique** qui est décrite comme suit (cf. Section 4.2.1.1.5).*

4.2.1.1.5 Cohérence logique (*Logical consistency*)

La cohérence logique décrit le degré d'adhésion (conformité) des données aux règles logiques des structures de données, des attributs et des relations (la structure de données peut être conceptuelle, logique, physique) [ISO/TC 211, 2002]. Autrement dit, la cohérence logique désigne l'absence de contradictions dans le jeu de données en rapport aux règles logiques présentes dans sa spécification. Ses sous éléments sont :

(a) Cohérence conceptuelle : adhésion des données aux règles du schéma conceptuel;

(b) Cohérence de domaine : adhésion des valeurs à leurs domaines de valeurs;

(c) Cohérence de format : le degré auquel les données sont stockées en accord avec la structure physique du jeu de données;

(d) Cohérence topologique : justesse des caractéristiques topologiques encodées explicitement du jeu de données.

L'approche la plus répondue pour la vérification de cette qualité dans les bases de données est le mécanisme de contrôle d'intégrité ou des contraintes d'intégrité (cf. Section 4.3).

4.2.1.2 Critères qualitatifs

Les critères qualitatifs définis par la norme ISO 19113:2002 [ISO/TC 211, 2002] pour décrire la qualité d'un jeu de données sont le but, l'usage et la généalogie. Ces critères sont décrits comme suit.

4.2.1.2.1 But (*Purpose*)

Le but décrit la raison de création du jeu de données et contient des informations sur son utilisation initialement prévue.

4.2.1.2.2 Usage (*Usage*)

L'usage décrit les applications dans les quelles le jeu de données a été employé. L'usage décrit les utilisations effectives du jeu de données par le producteur ou différents utilisateurs.

4.2.1.2.3 Généalogie (*Lineage*)

La généalogie décrit les informations sur l'histoire du jeu de données depuis sa création jusqu'à sa mise à disposition de l'utilisateur. La connaissance de ces informations est primordiale pour déterminer comment exploiter les données et quelles en sont les limites [Devillers et Jeansoulin, 2005]. La généalogie peut fournir des informations par exemple sur :

- les sources des données et les systèmes d'acquisition;
- les traitements ayant permis de réaliser le jeu de données;
- etc.

4.2.2 Qualité d'agrégation

Un autre aspect fondamental de la qualité d'analyse (S)OLAP concerne la qualité d'agrégation des données. En effet, avoir des données entreposées de bonne qualité ne garantit pas des agrégats corrects (valeurs exactes) et qui ont du sens pour l'application considérée. La qualité des indicateurs obtenus suite à l'agrégation des données, en plus des données, dépend également des types de hiérarchies (e.g. strictes ou non strictes), de fonctions d'agrégation (e.g. distributives ou pas) et de mesures (e.g. additives ou pas).

Cette problématique, connue sous le terme d'agrégeabilité, a été introduite pour la première fois dans le domaine des bases de données statistiques par [Rafanelli et Shoshani, 1990] et définit comme la possibilité de calculer des agrégats corrects à un niveau de granularité plus élevé en réutilisant des agrégats plus fins. Elle a été ensuite investiguée dans l'OLAP par le travail de [Lenz et Shoshani, 1997]. Ce travail de référence définit trois conditions pour garantir l'agrégeabilité : (i) Disjonction (*Disjointness*), (ii) Complétude (*Completeness*) et (iii) Compatibilité de types (*Type compatibility*). La condition de disjonction qui permet d'éviter le comptage en double des mesures, vérifie que les ensembles de données (instances de fait ou membres de dimension) formés à partir des membres se trouvant aux mêmes niveaux de détail en suivant les liens d'agrégation et les liens fait-dimension sont disjoints. La condition de complétude vérifie que ces ensembles sont complets; en d'autres termes, toutes les données (membres de dimension et instances de fait) sont présentes et sont correctement assignées aux différents ensembles; ceci permet d'éviter le problème des agrégats incomplets. La condition de comptabilité de types vérifie que l'application de la fonction d'agrégation à la mesure le long de la dimension est valide en considérant le type de la mesure (stock, flux, ou valeur par unité; cf. Section 2.3.4), le type de dimension (temporelle ou non temporelle) et le type de fonction d'agrégation utilisée (cf. Section 3.2.8).

Une analyse détaillée des causes d'inagrégeabilité est proposée par [Horner et Song, 2005]. Cette étude distingue trois familles de problèmes : problèmes de schéma (liés aux différents types de hiérarchies de dimension irrégulières - cf. Section 2.3.4), problèmes de données (imprécisions et incohérences dues à l'emploi de différents instruments de mesure) et problèmes de calcul (agrégation des mesures avec des fonctions inappropriées (incompatibilité de types – dans le sens de [Lenz et Shoshani, 1997]) et agrégation de mesures ayant des unités incompatibles).

Pour éviter ces problèmes d'inagrégeabilité et garantir les conditions définies par [Lenz et Shoshani, 1997] différentes approches sont proposées dans la littérature. Ces approches sont présentées et discutées en Section 4.5.

4.2.3Qualité d'exploration

Les outils (S)OLAP sont une catégorie de logiciels axés sur l'exploration interactive et rapide des données selon une approche multidimensionnelle à plusieurs niveaux d'agrégation [Bédard, 2009]. Les utilisateurs de ces systèmes sont essentiellement des analystes d'affaires et des décideurs, qui sont supposés ignorer les contraintes liant les données multidimensionnelles [Inmon, 2005]. Ils peuvent donc formuler assez facilement des requêtes multidimensionnelles inconsistantes en définissant des combinaisons insensées ou invalides de membres et d'indicateurs [Devillers et al., 2007; Boulil et al., 2012d]. Ces requêtes invalides dépendent de l'application considérée. Dans une application d'analyse de ventes, un exemple de requête invalide peut être "Quel est le total des ventes en Allemagne de l'Est en 2000?". Dans une application environnementale d'analyse de transfert de pesticides dans l'environnement (cf. Chapitre 8), un exemple pourrait être "Quelles sont les quantités d'isoproturon appliquées au blé durant une certaine période?"; sachant que cette matière active (isoproturon) ne s'applique pas au blé [Boulil et al., 2012a]. Bien que l'exécution de ces requêtes retourne des résultats nuls - en explicitant par exemple des CI de données qui interdisent le stockage de données (instances de fait) définies par ces combinaisons, un problème se pose au niveau de l'interprétation de ces résultats. L'utilisateur qui a formulé la requête va probablement interpréter ce résultat nul comme une absence de données; au lieu de réaliser que sa requête est insensée [Levesque et al., 2007] - par exemple du fait d'une absence de ventes dans les magasins pour l'Allemagne de l'Est en 2000. Ces interprétations erronées peuvent le mener à prendre de mauvaises décisions.
Sachant que les outils (S)OLAP existants ne prennent pas en compte ce type d'incohérences (sémantiques) au niveau des requêtes, les approches possibles pour éviter ces problèmes d'interprétation sont de définir des méthodes pour informer l'utilisateur sur la qualité des résultats de requêtes et/ou interdire l'exécution de ces requêtes incohérentes; et/ou les reformuler en requêtes valides; etc. Toutes ces approches nécessitent de vérifier si la requête utilisateur définit ou pas une combinaison problématique. Cette vérification peut être effectuée en définissant des CI sur les requêtes de façon similaire aux données [Boulil et al., 2012b]. En d'autres termes, une contrainte de requête va vérifier si la requête utilisateur définit une combinaison valide ou pas. Comme les autres types de contraintes, ces CI de requêtes doivent être spécifiées au niveau conceptuel pour être validées par les experts du domaine.

4.3 Contraintes d'Intégrité

Dans le domaine des bases de données, les CI sont définies comme des assertions qui expriment les conditions qui sont sensées être satisfaites par les données [Goertzen et Stausberg, 2007]. Elles sont utilisées principalement pour vérifier la cohérence logique des bases de données. Elles peuvent être vérifiées/contrôlées lors de l'insertion de nouvelles

données dans la base de données; et/ou la modification; et/ou la suppression de données existantes. Le mécanisme qui permet ce contrôle est connu sous le nom de mécanisme de contrôle d'intégrité.
Dans le domaine des entrepôts de données (spatiales), les CI sont typiquement utilisées pour garantir la cohérence des données intégrées à partir de plusieurs sources (qualité de données) [Carpani et Ruggia, 2001; Ghozzi et al., 2003b; Salehi, 2009; Pinet et Schneider, 2010]; spécifier les conditions qui garantissent l'agrégation correcte des mesures le long des hiérarchies de dimension (qualité d'agrégation) [Salehi, 2009; Prat et al., 2010] et vérifier la consistance des requêtes multidimensionnelles (qualité d'exploration) [Boulil et al., 2012b].

Les CI doivent être définies au niveau conceptuel afin de permettre leur validation par les experts du domaine car ce sont les langages utilisés à ce niveau qui servent de base à la communication entre les utilisateurs et les concepteurs [Torlone, 2003].
Par ailleurs, les langages de modélisation de structures de données ne peuvent exprimer qu'un nombre très réduit de contraintes d'intégrité [Olivé, 2006; OMG, 2006] comme par exemple les cardinalités des relations entre classes d'objets et les types de géométries possibles pour les données spatiales. Il est donc nécessaire de disposer de langages de spécification dédiés aux CI qui soient interopérables avec le langage de définition des structures de données utilisé et qui permettent de réduire la complexité du modèle de données en séparant la spécification des CI de celle du modèle de données. Cette séparation est plus que nécessaire dans le cas d'applications complexes comme les SOLAP par exemple.

Dans la littérature, différents travaux proposent différents langages pour la spécification des CI au niveau conceptuel. Ces langages sont classés dans [Salehi et al., 2007] en différentes catégories (cf. Section 4.3.1). Au niveau physique, des langages sont employés à différents niveaux de l'architecture (S)OLAP (cf. Sections 2.4 et 3.3). Par exemple, les langages SQL et Spatial SQL sont utilisés pour implémenter des CI au niveau de la couche ED(S) pour vérifier la qualité des données entreposées [Malinowski et Zimányi, 2008; Boulil et al., 2010b]; le langage de requête MDX au niveau de la couche serveur (S)OLAP[Boulil et al., 2012d]. D'autres langages d'implémentation sont possibles, par exemple via des programmes Java au niveau des clients (S)OLAP.

Dans la suite de cette section nous allons présenter et discuter les limites et avantages des différents langages de modélisation conceptuelle de CI dans le domaine des bases de données spatiales et entrepôts de données (cf. Section 4.3.1); et les langages de contraintes objet OCL et Spatial OCL que nous avons choisi d'utiliser dans nos travaux (cf. Section 4.3.2).

4.3.1Classification des langages de spécification CI

Les langages de modélisation conceptuelle des CI, dans les domaines des bases de données spatiales et des EDS sont classées en quatre grandes familles [Salehi et al., 2007] : (i) langages naturels, (ii) langages visuels, (iii) langages logiques et (iv) langages hybrides. Ces catégories ainsi que leurs sous catégories sont décrites dans ce qui suit.

4.3.1.1 Langages naturels

Les langages naturels sont issus des langages de communication de tous les jours entre les humains (comme l'Anglais et le Français). Comparés à d'autres types de langages, ces langages permettent une spécification plus facile des CI et plus de lisibilité (les utilisateurs comprennent plus facilement les spécifications en langues naturelles). Ces langages sont classés en deux sous-catégories : (i) langages naturels libres et (ii) langages naturels contrôlés.

4.3.1.1.1 Langages naturels libres

Un langage naturel libre correspond exactement à un langage naturel (e.g. Français) sans aucune restriction sur le vocabulaire, la syntaxe et la sémantique du langage. La syntaxe décrit la façon dont les éléments du langage doivent être combinés pour former des expressions (ou phrases) correctes. La sémantique décrit le sens des éléments et des expressions correctes du langage. Un exemple de contrainte spatiale spécifiée en Français peut être "les géométries des bâtiments ne doivent pas chevaucher celles des routes".
Ces langages offrent des vocabulaires riches et sont les plus faciles à utiliser. En revanche, ils présentent plusieurs ambiguïtés (e.g. un même mot peut avoir différents sens selon le contexte de son utilisation) et nécessitent beaucoup d'efforts d'implémentation. Leur implémentation automatique est très difficile voire même parfois impossible.

4.3.1.1.2 Langages naturels contrôlés

Ce sont des sous-ensembles de langages naturels dont la sémantique et la syntaxe ont été réduites. Ces langages ont été proposés dans le but de réduire l'ambiguïté des langages naturels tout en gardant leur facilité d'expression. Par exemple, la contrainte exemple précédente (cf. 4.3.1.1.1) est exprimée en utilisant le langage proposé par [Ubeda et Egenhofer, 1997] de la manière suivante :

```
(Bâtiment, Cross, Route, Forbidden)
```

Suivant ce langage, une CI spatiale définit une relation spatiale topologique du modèle 9IM [Egenhofer et Herring, 1994] (e.g. *inside*) entre les objets de deux classes (Classe 1 et Classe 2), avec un qualificatif indiquant le nombre de fois que la relation doit être vraie (e.g. jamais, exactement n fois, etc.). La syntaxe générale est la suivante :

```
(Classe 1, Relation, Classe 2, Qualificatif)
```

A notre connaissance, seul [Salehi, 2009] propose un langage de ce type pour spécifier au niveau conceptuel des CI d'agrégation dans le domaine des EDS (cf. Section 4.5.2).

4.3.1.2 Langages visuels

Les langages visuels se basent uniquement sur des notations visuelles pour la spécification des CI. Les langages visuels rendent les spécifications plus lisibles par les utilisateurs. Par exemple, la contrainte spatio-temporelle suivante (cf. Figure 4.1), modélisée avec le langage visuel proposé par [Pizano et al., 1989] dans le domaine des bases de données spatio-temporelles, spécifie que les véhicules et les personnes ne doivent pas se trouver au même moment sur un passage pour piétons. Dans cette modélisation, les personnes sont représentées par des astérisques, les véhicules par des rectangles et les passages piéton par des lignes obliques.

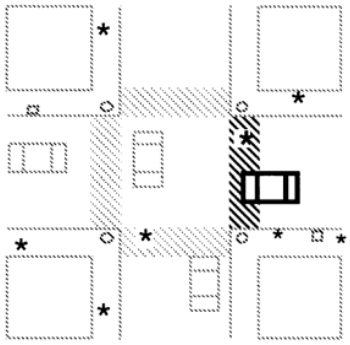


Figure 4.1. Exemple de langage visuel pour la spécification des CI. Issue de [Salehi, 2009].

Ces langages présentent en revanche d'importantes limites qui sont liées à : leur manque d'expressivité (i.e. un langage visuel ne peut exprimer qu'un nombre réduit de CI), la taille des expressions (les notations visuelles nécessitent un grand espace notamment pour exprimer des CI complexes), l'ambiguïté liée à différents contextes culturels et la difficulté d'apprentissage.
A notre connaissance, dans le domaine des ED et EDS aucun langage de ce type n'est proposé.

4.3.1.3 Langages logiques

Ces langages se basent essentiellement sur la logique du premier ordre pour la spécification des CI. Par conséquent, ces langages fournissent une sémantique et syntaxe assez précises permettant d'éviter de mauvaises interprétations et des ambiguïtés des/dans les spécifications de CI. Ces langages présentent également un bon niveau d'expressivité relativement aux autres. L'inconvénient majeur de ces langages est leur faible lisibilité et leur difficulté d'écriture puisqu'ils nécessitent un certain niveau de connaissances mathématiques de la part des concepteurs et utilisateurs qui n'ont pas forcément cette connaissance.
Par exemple dans le domaine des bases de données spatiales, la représentation conceptuelle de l'exemple de contrainte précédent (cf. Section 4.3.1.1.1) en utilisant le langage logique proposé par [Hadzilacos et Tryfona, 1992] est :

$$\forall r \in Route \, \forall b \in Batiment \, \neg Cross(r, b)$$

Dans le domaine des entrepôts de données classiques, le travail de [Carpani et Ruggia, 2001] propose un langage logique (cf. Section 4.4).

4.3.1.4 Langages hybrides

Ces langages sont des combinaisons des langages précédents (langages naturels, et/ou visuels et/ou logiques). Ces langages sont classés selon la composante dominante du langage (notations visuelles ou texte) en deux sous-catégories : langages hybrides visuels et langages hybrides naturels.

4.3.1.4.1 Langages hybrides visuels

La partie dominante de ces langages est constituée par des notations visuelles, essentiellement des symboles visuels. Ces langages combinent la lisibilité des langages visuels avec la richesse des langages naturels. Dans le domaine des bases de données spatiales, les exemples de cette catégorie sont les modèles conceptuels spatio-temporels MADS [Parent et al., 1999] et PVL (*Perceptory*) [Bédard et al., 2004]. Dans le domaine des ED et EDS, les exemples sont les langages ad hoc proposés dans [Carpani et Ruggia, 2001; Ghozzi et al., 2003a; Malinowski et Zimányi, 2008] (cf. Section 4.4).

Bien que ces langages/modèles améliorent la lisibilité des spécifications vis-à-vis des utilisateurs, ils ne permettent d'exprimer qu'un très petit nombre de contraintes. De plus dans le cas d'applications complexes comme par exemple celles du SOLAP, qui présentent beaucoup de contraintes, l'ajout des spécifications de contraintes au modèle de données dégrade sa lisibilité.

4.3.1.4.2 Langages hybrides naturels

Ces langages combinent un langage naturel qui constitue la partie dominante avec des pictogrammes ou des symboles.

4.3.1.4.2.1 Langages hybrides naturels avec pictogrammes

Un langage hybride naturel avec pictogrammes est un langage naturel enrichi avec un ensemble de pictogrammes. Les pictogrammes sont des symboles visuels intuitifs (i.e. un pictogramme se caractérise par sa ressemblance picturale à la notion du monde réel qu'il représente), concis et "standardisés" utilisés pour faciliter la modélisation des données et des CI en particulier dans le domaine des bases de données spatiales et des EDS [Parent et al., 1999; Ma et al., 2009; Salehi, 2009]. Un exemple de ce type de langages est proposé dans [Salehi, 2009] pour la spécification des CI dans les hypercubes de données spatiales (cf. Section 4.4). Les contributions amenées par ce langage sont très importantes. Néanmoins, le langage n'exprime pas certains types de contraintes et n'est pas compatible avec les standards de modélisation de données (UML et ER). Aussi, il n'a pas été implémenté.

4.3.1.4.2.2 Langages hybrides naturels avec symboles

Ces langages combinent un langage naturel contrôlé (partie dominante) avec des symboles visuels spécifiques (autres que des pictogrammes). Les exemples de ces langages sont le langage standard de contraintes objet OCL développé par l'OMG, les langages ad hoc proposés par [Ghozzi et al., 2003a] (cf. Section 4.4) et par [Hurtado et Mendelzon, 2001; Hurtado et al., 2005] (cf. Section 4.5.1). Contrairement à OCL, les inconvénients majeurs de ces langages ad hoc sont leur faible expressivité, leur absence d'implémentation et leur incompatibilité avec les langages standards de modélisation de données. Le langage OCL ainsi que son extension spatiale Spatial OCL que nous avons choisi d'utiliser dans nos travaux sont décrits dans la section suivante (cf. 4.3.2).

4.3.2 Langages OCL et Spatial OCL

OCL (*Object Constraint Language*) est un langage formel pour l'expression de contraintes et de requêtes sur des diagrammes UML (diagrammes de classes, états-transitions, etc.) [OMG, 2006]. Ce standard adopté par l'OMG présente de nombreux avantages [OMG, 2006; Chimiak-Opoka et Lenz, 2007]. C'est un langage qui combine la facilité d'expression et la lisibilité des langages naturels avec la non ambiguïté amenée par les langages formels. OCL a été conçu par IBM autour de 1995 comme un langage de modélisation métier (*Business modeling language*) à la portée des concepteurs de systèmes d'information; la syntaxe et la sémantique de ses expressions ont été formalisées afin d'éviter les ambiguïtés des langages naturels. C'est un langage déclaratif : il permet de spécifier les conditions qui doivent être satisfaites par un jeu de données sans préciser les processus ou algorithmes pour la vérification de ces conditions. C'est un langage sans effet de bord : l'évaluation d'une contrainte ne modifie pas l'état du système. C'est un langage conceptuel : il permet l'expression des contraintes au niveau conceptuel de façon générique et indépendante des plates-formes d'implémentation. OCL permet assez facilement d'exprimer un nombre beaucoup plus important de contraintes que les langages ad hoc proposés dans [Hurtado et Mendelzon, 2001; Ghozzi et al., 2003a; Malinowski et Zimányi, 2008; Salehi, 2009]. Dans [Mandel et Cengarle, 1999; Akehurst et Bordbar, 2001], l'expressivité d'OCL en tant que

langage de requête est étudiée. Les auteurs de ces travaux montrent qu'en considérant la notion de tuples dans OCL, le langage peut supporter l'algèbre relationnelle. Cette notion de tuples a d'ailleurs ensuite été intégrée à la spécification OCL à partir de la version 2.0.
En plus, de nombreux outils de modélisation UML (Eclipse, Rational Rose, MagicDraw, ArgoUML) et de générateurs de code implémentent OCL (Octopus, Bold for Delphi, OCL2SQL) et permettent l'implémentation automatique des contraintes [Klasse Objecten, 2009]; ce qui n'est pas le cas de la majorité des langages décrits précédemment. Ces générateurs de code permettent la production de mécanismes de contrôle d'intégrité dans différents langages (Java, SQL, etc.) à partir des spécifications de contraintes exprimées en OCL. Par exemple, Octopus génère du code Java; Bold for Delphi et OCL2SQL génèrent du code SQL (requêtes et triggers SQL) pour les principaux SGBD (e.g. Oracle et PosgreSQL).

OCL peut être utilisé soit comme langage de contraintes ou comme langage de requête, i.e. spécifier des expressions, qui une fois évaluées, retournent soit un résultat booléen, soit les valeurs qui satisfont les conditions définies par ces expressions. En tant que langage de contraintes, OCL permet de spécifier entre autres des invariants sur des classes et des types de données dans un modèle UML, exprimer des pré et post conditions sur des opérations et méthodes, définir des invariants sur les stéréotypes d'un profil UML; etc.
Dans nos travaux, nous utilisons essentiellement les contraintes OCL de type invariant pour exprimer des contraintes au niveau modèle sur le modèle multidimensionnel (cf. Chapitres 5 et 7, et [Boulil et al., 2010b]) et au niveau métamodèle sur les stéréotypes du profil UML SOLAP que nous proposons (cf. Chapitres 5 et 6, et [Boulil et al., 2011]).
Un invariant OCL (mot clef *inv*) se définit dans le contexte d'une classe ou d'un stéréotype et précise les conditions qui doivent être satisfaites par ses instances. Par exemple, l'invariant OCL suivant exprime le fait que tous les employés dont le poste actuel est chercheur doivent percevoir un salaire supérieur ou égal à 1500€. Le contexte de définition de cette contrainte est la classe "Employe" (mot clef *context*). Le mot clef *self* est utilisé dans OCL pour référencer l'objet courant de la classe spécifiée dans le contexte; l'opérateur *implies* correspond à l'implication logique; "poste_actuel" et "salaire" sont des attributs de la classe "Employe". Plus de détails sur les invariants OCL sont présentés en annexe A.

```
context Employe inv:
self.poste_actuel = 'chercheur' implies self.salaire >= 1500
```

Pour résumer, dans nos travaux nous avons choisi OCL pour la spécification des contraintes d'intégrité non spatiale dans le domaine des entrepôts de données spatiales essentiellement car il nous permet d'exprimer assez facilement au niveau conceptuel la plupart des contraintes que nous avons identifiées et ceci de manière non ambiguë et indépendante des plates-formes d'implantation cibles. Aussi, il permet une implémentation automatique des contraintes.
Pour l'expression des contraintes d'intégrité spatiale, nous utilisons l'extension spatiale d'OCL, Spatial OCL, proposée dans le cadre des travaux de thèse de Magali Duboisset [Duboisset, 2007]. Cette extension permet d'exprimer les relations spatiales topologiques du modèle 9IM [Egenhofer et Herring, 1994] entre géométries simples et complexes. Pour ce faire, Spatial OCL rajoute à OCL les types géométriques primitifs Point, Ligne et Region, ainsi que les relations spatiales topologiques 9IM : *disjoint*, *meet*, *overlap*, *inside*, *contains*, *covers*, *coveredBy* et *equal*. Les géométries complexes sont définies comme des ensembles de géométries simples. Ci-après un exemple de contrainte Spatial OCL qui spécifie que les géométries des villes doivent être disjointes. Les détails de cette extension sont présentés en annexe A également.

```
Context Ville inv:
Ville.allInstances -> forAll(v1, v2 | v1 <> v2 implies
v1.geo.areDisjoint (v2.geo))
```

Pour permettre l'implémentation automatique des CI, Spatial OCL a été implémenté par le générateur de code Spatial OCL2SQL [Duboisset et al., 2007]. Spatial OCL2SQL (cf. Figure 4.2) est un outil Open Source développé en Java qui étend le générateur de code OCL2SQL de l'université de Dresde[19] par les nouvelles syntaxes de Spatial OCL ainsi que leurs mappings automatiques vers le SQL spatial d'Oracle. Afin de réaliser la génération de code, les principales inputs nécessaires sont : (i) le fichier XMI[20] du modèle conceptuel de données (UML), (ii) un fichier de contraintes Spatial OCL et (iii) un fichier de métadonnées pour les attributs géométriques (incluant entre autres le système de référence spatiale). En sortie, l'outil produit (a) des scripts pour la création du schéma physique de la base de données spatiales ou de l'EDS et (b) un ensemble de requêtes, vues et de triggers SQL pour implémenter les CI. Pour chaque contrainte Spatial OCL, l'outil génère une requête, une vue et un ou plusieurs triggers. La requête et la vue sélectionnent les tuples de table (la table qui correspond au contexte de la contrainte Spatial OCL) qui ne satisfont pas la contrainte. Les triggers renvoient un message d'erreur si le nombre de tuples sélectionnés par la vue est supérieur à zéro. Plus de détails sur ce mode d'implémentation peuvent être trouvés dans [Duboisset, 2007; Pinet, 2010].

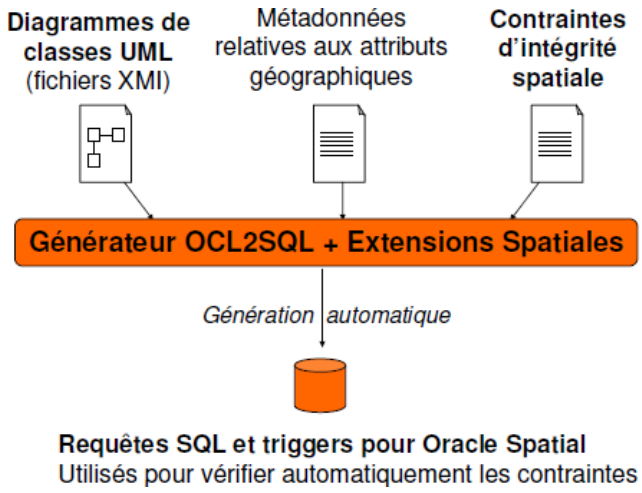


Figure 4.2. Entrées et sorties du générateur de code Spatial OCL2SQL. Issue de [Pinet, 2010].

4.4 Travaux sur la cohérence logique de données dans les ED et EDS

La problématique d'incohérence logique des données dans les systèmes d'ED(S) est due essentiellement à la phase d'intégration des sources données [Salehi, 2009]. En effet, en raison des différentes hétérogénéités qui peuvent exister entre les sources de données, de

[19] http://www.dresden-ocl.org/index.php/DresdenOCL

[20] XML Metadata Interchange (XMI) est un standard OMG basé sur XML pour l'échange de métadonnées [OMG, 2007].

nombreuses incohérences peuvent être rencontrées lors de cette phase importante du processus d'entreposage. Par exemple, dans l'application d'analyse des ventes (cf. Section 3.2), après intégration des sources de données, on peut avoir des géométries de villes qui se chevauchent; des géométries qui ne sont pas dans leurs domaines de définition, etc.

Quelques travaux traitant de l'incohérence logique ont été menés concernant la définition des contraintes d'intégrité dans les ED [Carpani et Ruggia, 2001; Ghozzi et al., 2003a; Pinet et Schneider, 2009; Turki et al., 2010] et EDS [Malinowski et Zimányi, 2008; Salehi, 2009; Pinet et Schneider, 2010].

Dans le domaine des entrepôts de données classiques, [Carpani et Ruggia, 2001] proposent le premier modèle conceptuel pour les hypercubes qui intègre des contraintes d'intégrité. Ce modèle définit les concepts de dimension, niveau d'agrégation et de relation dimensionnelle (ou fait) ainsi que leurs instances. Les contraintes peuvent être définies sur un ou plusieurs éléments du modèle. Ainsi les CI intra-membre portent sur un membre de dimension; les CI inter-membres, inter-niveaux et inter-dimensions portent respectivement sur plusieurs (deux ou plus) membres, plusieurs niveaux d'agrégation et plusieurs dimensions. Ces différentes contraintes sont spécifiées au niveau conceptuel en utilisant deux langages, un langage logique et un langage hybride visuel (cf. Section 4.3.1). Par exemple, la CI intra-membre suivante, spécifiée avec un langage logique, précise que chaque vendeur (niveau d'agrégation "Salesman″) doit avoir un âge entre 18 et 61.

$$\forall x \in Salesman.(x(age) > 18 \wedge x(age) < 61)$$

Par contre, [Carpani et Ruggia, 2001] ne considère pas les CI qui portent sur les faits, ainsi que les CI d'agrégation et les CI d'exploration. De plus aucune implémentation n'est proposée.

[Ghozzi et al., 2003a] proposent un modèle multidimensionnel en constellation qui permet de modéliser des dimensions avec plusieurs hiérarchies et instances. Ils introduisent deux types de contraintes d'intégrité : CI intra-dimension et CI inter-dimensions. Une CI intra-dimension spécifie une inclusion ou une exclusion entre les ensembles d'instances de deux hiérarchies d'une même dimension. Par exemple, si on considère la dimension "Magasins" représentant les magasins de ventes des produits dans l'hypercube d'analyse des ventes (cf. Section 3.2), avec deux hiérarchies, "Magasins_FR" et "Magasins_US" représentant respectivement les magasins Français et Américains; alors une contrainte intra-dimension entre ces deux hiérarchies peut exprimer que les membres qui appartiennent à la hiérarchie "Magasins_FR" ne doivent pas appartenir à la hiérarchie "Magasins_US" et vice versa. Une CI inter-dimensions spécifie une inclusion ou une exclusion entre les ensembles d'instances de hiérarchie de deux dimensions. Ces CI contrôlent exactement le dimensionnement des instances de fait (i.e. quelles combinaisons des membres des différentes dimensions sont valides pour identifier les instances de fait). Par exemple, en plus de la dimension "Magasins" précédente, considérons la dimension "Produits" avec deux hiérarchies, "Produits-FR" (produits Français) et "Produits_US" (produits US), alors une CI inter-dimensions d'exclusion peut être spécifiée par exemple entre les hiérarchies "Produits-FR" et "Magasins_US" pour dire que les magasins US ne vendent pas de produits Français.
Ces contraintes sont spécifiées au niveau conceptuel au moyen de deux types de langages, hybride naturel avec symboles et hybride visuel (cf. Section 4.3.1.4). Par exemple, la spécification avec le langage hybride naturel avec symboles de la CI inter-dimensions entre les hiérarchies "Produits-FR" et "Magasins_US" est la suivante :

```
Produits-FR ⊗ Magasins_US
```

En plus des CI d'agrégation et d'exploration, ce travail ne considère pas plusieurs types de CI de données, par exemple les CI impliquant plusieurs faits, un ou plusieurs membres de dimension (cf. Chapitres 5 et 7), etc. Ce travail ne propose également pas d'implémentation.

Dans le domaine des entrepôts de données spatiales, [Malinowski et Zimányi, 2008] proposent le modèle conceptuel spatio-multidimensionnel ER, Spatial MultiDimER (cf. Section 3.4.1.2). Ce modèle utilise les pictogrammes du modèle MADS pour la spécification de contraintes d'intégrité spatiale simples. Plus exactement, les pictogrammes du modèle MADS sont utilisés pour (i) représenter la géométrie d'un niveau d'agrégation spatial ou d'une mesure spatiale (CI de domaine de valeurs), (ii) exprimer une relation spatiale topologique (e.g. *coveredBy*) entre les géométries de deux niveaux d'agrégation spatiaux reliés par une relation d'agrégation dans une hiérarchie de dimension spatiale et (iii) enfin spécifier une relation spatiale topologique entre des niveaux d'agrégation spatiaux feuille de plusieurs dimensions spatiales afin de définir un fait spatial. Les relations spatiales topologiques considérées sont les relations définies par le modèle 9IM. La figure 4.3 montre des exemples de spécifications de relations spatiales topologiques (*coveredBy*) entre niveaux d'agrégation spatiaux (⬬) et des exemples de spécifications de géométries de niveaux d'agrégation spatiaux (◭).
Les auteurs proposent également un mapping de ce modèle vers le modèle logique Objet-Relationnel et une implémentation manuelle des contraintes sous forme de triggers, vues et fonctions SQL dans le SGBD spatial Oracle Spatial 10g. Cependant, en plus des CI d'agrégation sémantiques (condition de comptabilité de types [Lenz et Shoshani, 1997]) et d'exploration, beaucoup de types de CI de données ne sont pas considérés, par exemple les CI qui portent sur les attributs thématiques, les relations entre attributs thématiques et spatiaux, les instances de fait, etc. De plus, le langage visuel (cf. Section 4.3.1.4.1) proposé (ensemble de pictogrammes) ne peut spécifier que les types de CI spatiales décrites précédemment.

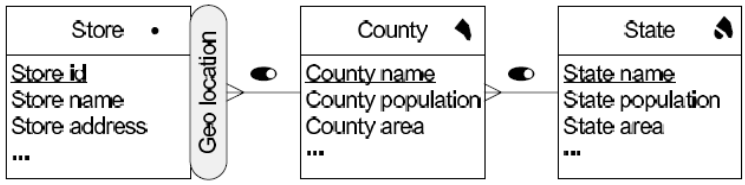


Figure 4.3. Exemples de spécifications de CI avec Spatial MultiDimER [Malinowski et Zimányi, 2008].

[Salehi, 2009] proposent un modèle conceptuel ad hoc pour les hypercubes de données spatiales (cf. Section 3.4.2), une classification détaillée des CI des hypercubes spatiaux et un langage hybride naturel avec pictogrammes pour la spécification de ces CI au niveau conceptuel. La catégorisation de CI proposée considère les CI de données et d'agrégeabilité; cette dernière catégorie est présentée en Section 4.5.2. Les contraintes de données définies peuvent contraindre les données dimensionnelles (CI traditionnelles) ou les données factuelles (CI de faits). Les CI sur les dimensions considérées peuvent porter sur un ou plusieurs attributs d'un membre de dimension (CI Inter 0 et CI Inter 1 respectivement), plusieurs membres (CI Inter 2), plusieurs niveaux d'agrégation (CI Inter 3), ou plusieurs dimensions (CI Inter 4). Les CI de faits peuvent contraindre une instance de fait (CI F-Inter 0), plusieurs instances de fait d'une seule hypercellule[21] (CI F-Inter 1), ou plusieurs instances

[21] Une hypercellule définit le schéma commun à un ensemble d'instances de fait agrégées ou de base (cf. Section 3.4.2).

de fait de plusieurs hypercellules (CI F-Inter 2). Ces différents types de CI sont modélisés au niveau conceptuel en utilisant un langage hybride naturel avec pictogrammes (cf. Section 4.3.1.4.2.1). Par exemple, la CI Inter 2 (porte sur plusieurs membres d'un niveau d'agrégation) représentée en Figure 4.4, spécifie que les géométries des états (membres du niveau d'agrégation "State") situés dans la région "West" doivent être spatialement disjointes des géométries des états situés dans la région "Northeast".

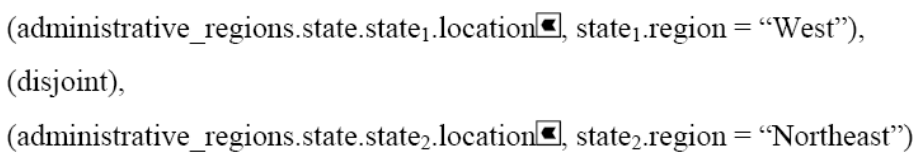


Figure 4.4. Exemple de spécification de CI Inter 2 [Salehi, 2009]

Ce travail comme les précédents ne propose pas d'implémentation pour les CI spécifiées. De plus, la catégorisation de CI proposée ne prend pas en compte les CI de données impliquant plusieurs hypercubes et les CI d'exploration. Finalement, le langage de spécification de CI proposé n'est pas basé sur des langages standards, ce qui rend l'implémentation difficile.

A travers quelques exemples et en se basant sur le modèle multidimensionnel UML qu'ils proposent (cf. Sections 2.5.1 et 3.4.1.1), [Pinet et Schneider, 2009; Pinet et Schneider, 2010] montrent comment les langages de spécification de CI standards OCL et Spatial OCL peuvent être utilisés pour exprimer des CI d'ED et d'EDS au niveau conceptuel. Ces contraintes sont définies pour garantir la cohérence logique des données intégrées dans l'entrepôt à partir de plusieurs sources hétérogènes. Par exemple, dans l'exemple d'analyse des épandages agricoles (cf. Section 3.2.5), la contrainte Spatial OCL suivante spécifie que les géométries des zones d'épandage ("zone-epandue") de chaque commune doivent être soit égales ou soit incluses dans la géométrie de la commune ("localisation") à laquelle elles appartiennent :

```
context Epandages inv :
(self.zone-epandue).{equal|inside|coveredBy} (self.Commune.localisation)
```

Les travaux de [Pinet et Schneider, 2009; Pinet et Schneider, 2010] ne proposent ni classification de CI, ni implémentation pour les contraintes spécifiées. De plus, les auteurs ne considèrent pas les CI d'agrégation et d'exploration.

Tous les travaux ci-dessus utilisent ou proposent des langages de spécification de CI en plus des langages de définition des structures de données multidimensionnelles; ceci permet comme dit précédemment (cf. Section 4.3) d'exprimer un plus grand nombre de types de contraintes et d'éviter la complexification du modèle de données en séparant la spécification des CI de la spécification des structures de données. Contrairement aux travaux ci-dessus, [Pestana et al., 2005; Glorio et Trujillo, 2008] se basent seulement sur les modèles de données pour spécifier les CI. Ces modèles n'expriment que des contraintes d'intégrité simples, i.e. des contraintes de domaine de valeurs des attributs (e.g. géométries des niveaux d'agrégation spatiaux et mesures spatiales) et des contraintes de multiplicité des relations d'agrégation et de fait-dimension.

4.5 Travaux sur la qualité d'agrégation

Les conditions d'agrégeabilité définies dans [Lehner et al., 1998] peuvent être regroupées en deux catégories : (i) des conditions structurelles (i.e. disjonction et de complétude, cf. Section 4.5.1) et des conditions sémantiques (condition de compatibilité de types, cf. Section 4.5.2).

4.5.1Conditions structurelles d'agrégeabilité

Les conditions structurelles d'agrégeabilité (disjonction et complétude) dépendent [Mazón et al., 2009] :
- des types de relations d'agrégation et de hiérarchies de dimension (e.g. les relations d'agrégation non strictes peuvent engendrer le comptage en double des mesures);
- et des types de relations fait-dimension liant les faits aux dimensions (e.g. les relations fait-dimension incomplètes[22] peuvent engendrer des agrégats incomplets).

Concernant ces conditions structurelles, les approches de modélisation multidimensionnelle conceptuelle peuvent être classées en 4 catégories :

Catégorie A : *ces approches garantissent l'agrégeabilité en interdisant la présence des structures multidimensionnelles complexes problématiques à l'agrégeabilité (cf. Section 2.3.4) dans les modèles conceptuels. L'inconvénient de ce type d'approches est que de nombreuses situations réelles (hiérarchies complexes) sont ignorées dans la modélisation.*
Dans cette catégorie, nous retrouvons par exemple le modèle ad hoc DFM [Golfarelli et al., 1998] (cf. Section 2.5.1) et les travaux qui utilisent les Dépendances Fonctionnelles (DF) pour garantir ces conditions structurelles d'agrégeabilité [Lehner et al., 1998; Niemi et al., 2001; Lechtenbörger et Vossen, 2003]. Ces travaux étendent les DF et Formes Normales (FN) des bases de données relationnelles aux bases de données multidimensionnelles.

Catégorie B : *Ces approches proposent des modèles conceptuels qui représentent toutes ou quelques-unes des structures multidimensionnelles complexes mais ne fournissent ni de solutions pour déterminer l'agrégeabilité de ces structures (i.e. déterminer si ces structures respectent ou pas les conditions structurelles d'agrégeabilité), ni des solutions pour résoudre les problèmes d'agrégeabilité qui peuvent être induits par ces structures.*
Les travaux de cette catégorie sont par exemple les modèles multidimensionnels conceptuels proposés dans [Abelló et al., 2006; Lujan-Mora et al., 2006] (cf. Section 2.5.1) et les modèles spatio-multidimensionnels conceptuels proposés dans [Glorio et Trujillo, 2008; Pinet et al., 2010; Salehi et al., 2010] (cf. Section 3.4.)

Catégorie C : *Ces travaux proposent des modèles qui permettent la représentation conceptuelle des structures multidimensionnelles complexes et proposent des algorithmes pour déterminer l'agrégeabilité de ces structures. Ces algorithmes en revanche ne permettent pas de rendre ces structures agrégeables.*
Les principaux travaux de cette catégorie sont [Hurtado et Mendelzon, 2001; Hurtado et al., 2005; Pinet et Schneider, 2009; Pinet et Schneider, 2010]; ils sont décrits ci-dessous.

Dans le domaine des ED, [Hurtado et Mendelzon, 2001; Hurtado et al., 2005] introduisent de nouvelles contraintes, appelées contraintes de dimension, pour déterminer l'agrégeabilité des

[22] Une relation fait-dimension incomplète est une relation qui présente une multiplicité minimale de zéro du coté de la dimension (au sens des diagrammes de classes UML) et qui signifie que des instances de fait peuvent ne pas être liées à aucune instance de la dimension impliquée dans la relation [Mazón et al., 2009].

hiérarchies de dimension hétérogènes[23] (ces hiérarchies correspondent à un type particulier de hiérarchies non strictes; cf. Section 2.3.4). Ces contraintes définissent les chemins d'agrégation possibles des membres le long des hiérarchies de dimension. Elles sont définies en utilisant un langage hybride naturel avec symboles (cf. Section 4.3.1.4.2.2). Par exemple, la contrainte de dimension de la figure 4.5 spécifie que tout produit (Niveau d'agrégation "Product") s'agrège soit à une branche de produits (Niveau d'agrégation "Branch"), soit à un département (Niveau d'agrégation "Department") mais pas les deux à la fois. Finalement, un prototype permettant de tester l'agrégeabilité de ces hiérarchies hétérogènes en présence de ces contraintes de dimension est développé en Java.

Dans les ED et EDS, [Pinet et Schneider, 2009; Pinet et Schneider, 2010] expriment avec OCL et UML différents types de contraintes d'agrégeabilité structurelles. Les contraintes exprimées peuvent porter sur une ou plusieurs relations d'agrégation et même sur plusieurs chemins d'agrégation. Ces contraintes sont aussi utilisées pour caractériser différents types de hiérarchies. Ce travail propose un algorithme pour tester l'agrégeabilité d'une hiérarchie de dimension en présence de ces contraintes.

$$\langle Product, Branch \rangle \oplus \langle Product, Department \rangle$$

Figure 4.5. Exemple spécification d'une contrainte de dimension [Hurtado et al., 2005].

Catégorie D : *Les travaux de cette catégorie proposent des modèles qui tolèrent la présence des structures multidimensionnelles irrégulières (cf. Section 2.3.4) et des algorithmes et mappings pour transformer ces structures en structures agrégeables.*

Les principaux travaux dans cette catégorie sont décrits comme suit.

Dans le domaine des ED, [Pedersen et al., 1999] proposent trois algorithmes pour transformer les instances de hiérarchie de dimension irrégulières (non onto, non covering et non strictes – cf. Section 2.3.4) en instances de hiérarchies régulières (i.e. respectant les conditions structurelles d'agrégeabilité). [Mazón et al., 2006] définissent un ensemble de relations QVT (*Query/View/Transformation language*) basées sur les formes normales multidimensionnelles de [Lechtenbörger et Vossen, 2003] pour vérifier l'agrégeabilité du modèle multidimensionnel conceptuel et sa validité par rapport au modèle des sources. Ces relations QVT modifient (transforment) le schéma multidimensionnel conceptuel pour satisfaire certaines des formes normales multidimensionnelles de [Lechtenbörger et Vossen, 2003] garantissant en partie l'agrégeabilité structurelle.

Dans les ED et EDS, [Malinowski et Zimányi, 2008] propose une classification qui définit différents types de hiérarchies de dimension complexes (cf. Section 2.3.4). Ces hiérarchies sont représentées au niveau conceptuel en utilisant le modèle (Spatial) MultiDimER (cf. Sections 2.5.1 et 3.4.1.2). Les auteurs définissent également de manière informelle un ensemble de mappings de ces hiérarchies conceptuelles vers les modèles logiques Relationnel et Objet-Relationnel permettant leur implémentation dans les SGBD existants en garantissant les conditions structurelles d'agrégeabilité Contrairement à [Pedersen et al., 1999], ce travail en revanche ne définit pas l'ordre d'application de ces mappings dans le cas où la hiérarchie de dimension présente plusieurs irrégularités structurelles (e.g. non stricte et non onto).

[23] Une hiérarchie de dimension hétérogène est une hiérarchie de dimension où deux membres d'un même niveau d'agrégation ont des ancêtres dans deux niveaux d'agrégation différents [Hurtado et al., 2005].

Dans le domaine des EDS, [Pedersen et Tryfona, 2001] analysent l'influence des relations spatiales topologiques du modèle 9IM sur l'additivité des mesures métriques (e.g. "surface" de la zone épandue (cf. Section 3.2.5) et montrent que celle-ci dépend de la disjonction des objets spatiaux. Cette étude propose une approche pour transformer l'EDS et garantir l'agrégeabilité dans le cas où cette condition n'est pas vérifiée (les objets ne sont pas disjoints). Cette approche consiste à calculer les parties disjointes avec les valeurs de mesure associées et à normaliser les hiérarchies de dimension en appliquant les transformations présentées [Pedersen et al., 1999]. [Jensen et al., 2004] proposent une méthode pour évaluer l'imprécision des chemins d'agrégation engendrée par les relations d'inclusion partielle entre membres de dimension spatiale. Cette méthode permet de choisir le chemin le plus précis parmi plusieurs alternatives lors de l'exécution des requêtes. Les auteurs adaptent également les algorithmes de transformation des hiérarchies irrégulières (non strictes, non onto et non covering) proposés dans [Pedersen et Tryfona, 2001] pour supporter ces relations d'inclusion partielle.

Tous les travaux ci-dessus dans cette catégorie se focalisent sur les hiérarchies de dimension irrégulières (cf. Section 2.3.4). Pour résoudre les problèmes d'agrégeabilité dus aux relations fait-dimension irrégulières (non strictes et incomplètes), [Mazón et al., 2008] propose des transformations pour convertir un modèle multidimensionnel conceptuel non agrégeable (i.e. qui contient ces relations fait-dimension irrégulières) en un modèle agrégeable. Ces transformations sont exprimées en utilisant le langage QVT et implémentées par un module sous Eclipse.
Enfin, tous ces algorithmes et techniques de transformation de structures multidimensionnelles qui sont proposées par les travaux de cette catégorie engendrent une modification et complexification de l'analyse (O)LAP [Mazón et al., 2009] notamment par l'insertion de valeurs artificielles qui n'ont pas de sens pour les décideurs.

Dans cette thèse, nous exprimons les conditions structurelles d'agrégeabilité sous la forme de contraintes de multiplicité UML et de contraintes OCL [Boulil et al., 2011] d'une façon similaire à [Mazón et al., 2009; Pinet et Schneider, 2010], mais nous ne nous intéressons par à leur implémentation. Il ne s'agit pas d'une limitation à notre travail car comme présenté dans cette section plusieurs approches ont été proposées pour transformer toutes les structures multidimensionnelles non agrégeables en structures agrégeables.

4.5.2Conditions sémantiques d'agrégeabilité

Les conditions sémantiques d'agrégeabilité (condition de compatibilité de types [Lenz et Shoshani, 1997]) dépendent essentiellement de la comptabilité entre les natures sémantiques et types de la mesure, de la fonction d'agrégation et de la dimension.

Concernant ces conditions sémantiques d'agrégeabilité, certains travaux identifient seulement différentes typologies de mesures, de fonctions d'agrégation et de dimensions et les incompatibilités entre ces typologies sans proposer de solutions pour les représenter au niveau conceptuel ou les implémenter [Lenz et Shoshani, 1997; Shekhar et al., 2001]

D'autres travaux proposent des solutions pour exprimer ces incompatibilités au moyen de contraintes dans les modèles conceptuels. Ces travaux utilisent différents langages. Mais en règle générale, ces travaux ne fournissent pas d'implémentations et ne donnent aucune indication sur la façon dont ces contraintes peuvent être intégrées dans les outils OLAP/SOLAP pour contrôler l'agrégation. Ces travaux sont décrits comme suit.

Certains modèles conceptuels d'ED et d'EDS [Tryfona et al., 1999; Malinowski et Zimányi, 2008] ne représentent que le type additif de la mesure. Dans le modèle multidimensionnel ER

de [Tryfona et al., 1999], le type additif de la mesure (Flux, Stock ou Valeur par unité) est indiqué par un langage naturel contrôlé (cf. Section 4.3.1.1.2) : une lettre F, S ou V respectivement. Le modèle Spatial MultiDimER de [Malinowski et Zimányi, 2008] représente le type de mesure par des symboles visuels (langage visuel).

Les modèles proposés par [Nguyen et al., 2000b; Ravat et al., 2008] permettent d'exprimer seulement des incompatibilités entre les mesures et les fonctions d'agrégation sans considérer les dimensions. Pour ce faire ces modèles intègrent la fonction d'agrégation dans la définition de la mesure. Par exemple, [Nguyen et al., 2000b] propose un modèle conceptuel UML où la mesure est définie par son nom et sa fonction d'agrégation.

Les travaux [Golfarelli et al., 1998; Abelló et al., 2006; Lujan-Mora et al., 2006] spécifient seulement des contraintes d'agrégation qui définissent la ou les fonctions d'agrégation qui peuvent être appliquées à une mesure ou un ensemble de mesures le long d'une ou plusieurs dimensions sans considérer les hiérarchies de dimension. Dans les ED, le modèle DFM proposé par [Golfarelli et al., 1998] (cf. Section 2.5) représente les fonctions d'agrégation qui peuvent être appliquées à une mesure le long d'une dimension en utilisant un langage hybride visuel. Le modèle YAM^2 [Abelló et al., 2006] (cf. Section 2.5) modélise les règles d'agrégation de mesures le long des dimensions sous forme d'opérations UML stéréotypées *Summarization*. Exactement chaque règle précise une fonction d'agrégation pour une mesure et un ensemble de dimensions. YAM^2 définit également des contraintes qui spécifient qu'un niveau d'agrégation est une source invalide pour l'agrégation (les agrégats à ce niveau ne peuvent être réutilisés pour calculer les agrégats des niveaux supérieurs dans la hiérarchie de dimension). [Lujan-Mora et al., 2006] (cf. Section 2.5) modélise des contraintes sur l'additivité des mesures par des notes UML. Ces contraintes spécifient seulement les dimensions le long desquelles les mesures ne peuvent être sommées (fonction Sum).

[Bimonte et al., 2009] proposent un modèle spatio-multidimensionnel (cf. Section 3.4.2) qui prend en compte les dépendances entre fonctions d'agrégation spatiales et alphanumériques lors de l'agrégation des mesures spatiales. Pour contrôler la qualité de l'agrégation alphanumérique, ce modèle formalise deux contraintes qui spécifient pour chaque mesure alphanumérique le type de fonctions d'agrégation possible : la *contrainte d'agrégation verticale* considère le type de la mesure (e.g. additive ou pas) et de la dimension utilisée (temporelle ou pas); et la *contrainte d'agrégation horizontale* considère le type de mesure et la fonction d'agrégation spatiale utilisée pour agréger la géométrie. Ces contraintes sont spécifiées par des fonctions au niveau métamodèle.

Les travaux de [Prat et al., 2010] et [Salehi, 2009] proposés dans le contexte des ED et EDS respectivement sont les travaux qui représentent le plus de contraintes d'agrégation en relation avec la condition de compatibilité de types. En se basant sur un modèle multidimensionnel UML, [Prat et al., 2010] spécifient différentes contraintes d'agrégation sémantiques en utilisant le standard OMG de représentation des règles de production PPR (*Production Rule Representation language* – cf. Section 2.5.1) pour garantir des agrégations correctes des mesures. Ces règles expriment entre autres les fonctions d'agrégation numériques applicables aux mesures le long des dimensions, l'ordre d'application des fonctions et leur distributivité. Ce travail ne propose pas d'implémentation à ces contraintes.
[Salehi, 2009] définit six catégories de CI d'agrégeabilité (cf. Figure 4.6). Chaque catégorie est utilisée pour spécifier une interdiction d'un type de combinaisons lors d'une agrégation. Par exemple, les CI de catégorie *M-D-AF IC* interdisent l'agrégation d'une mesure avec une fonction d'agrégation le long d'une dimension. Ces contraintes sont spécifiées au moyen d'un langage naturel contrôlé (cf. Section 4.3.1.1.2), comme montré en Figure 4.7. Les auteurs ne proposent pas d'implémentation. Certaines catégories de CI sont insensées dans le contexte

du (S)OLAP, comme la catégorie *M IC* qui interdit l'agrégation d'une mesure. De notre point de vue, une mesure qui ne peut pas être agrégée ne peut être considérée comme utile dans l'analyse (S)OLAP.

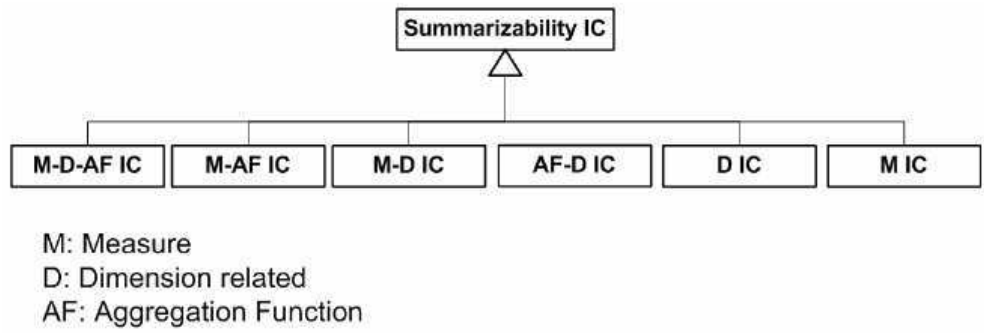


Figure 4.6. Classification des CI d'agrégeabilité [Salehi, 2009].

(measure: "fire_surface", *aggregation function:* "SUM", *dimension*: "time"*) is not summarizable.*

Figure 4.7. Exemple de CI d'agrégeabilité [Salehi, 2009].

4.6 Travaux sur la qualité d'exploration

La qualité d'exploration a été principalement investiguée dans le domaine de l'OLAP par les travaux de [Sapia, 1999; Böhnlein et al., 2002] et dans le domaine du SOLAP par [Levesque et al., 2007]. De manière générale ces travaux ne proposent ni langages pour la spécification conceptuelle des contraintes d'exploration, ni de langages pour leur implémentation.

Les travaux de [Sapia, 1999; Böhnlein et al., 2002] s'intéressent principalement à la modélisation conceptuelle d'une session d'analyse OLAP (séquence de requêtes). Ces travaux proposent des modèles et des notations graphiques (Figure 4.8) pour représenter la requête OLAP au niveau conceptuel. Ils considèrent des définitions très similaires de la requête OLAP. La requête OLAP est définie par deux parties : le template (clause SELECT) qui définit le sous-ensemble de mesures et la combinaison des niveaux d'agrégation qui doivent apparaitre dans le résultat; et le filtre (clause WHERE) qui spécifie des conditions pour la sélection des membres de dimension à considérer dans la requête. Cette définition est enrichie par quelques contraintes structurelles pour garantir la validité structurelle de la requête OLAP. Le modèle proposé par [Sapia, 1999] considère la contrainte structurelle suivante : "exactement un niveau d'agrégation par dimension doit apparaitre dans la requête soit dans le résultat ou dans le filtre". Dans le modèle de [Böhnlein et al., 2002], les contraintes considérées vérifient que la mesure et le niveau d'agrégation appartiennent au même hypercube et empêchent la combinaison de niveaux d'agrégation appartenant à différentes hiérarchies d'une même dimension.
Les contraintes d'exploration structurelles définies par ces travaux sont implémentées par les serveurs OLAP actuels (e.g. Mondrian) et ne nécessitent pas donc une spécification.

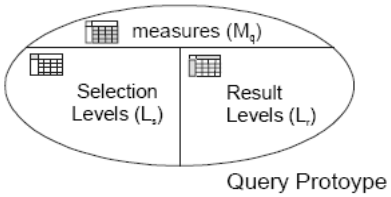


Figure 4.8. Représentation graphique d'un patron de requêtes OLAP [Sapia, 1999].

Dans le domaine du SOLAP, [Levesque et al., 2007] propose une approche pour aider les producteurs de données à identifier et gérer les risques potentiels de mauvaise utilisation des hypercubes de données spatiales; ceci dans le but d'éviter par exemple les conséquences des décisions basées sur une interprétation incorrecte des données. L'approche proposée s'inspire du management de risque du domaine de gestion de projet et de quelques standards ISO et comporte donc les phases classiques suivantes : identification, analyse et évaluation du risque, réponse au risque et enfin la documentation du processus de management de risque. Une fois le risque connu, différentes solutions de réponse au risque sont possibles, par exemple éliminer les données présentant un risque inacceptable; tolérer la présence de ces données et informer l'utilisateur via des warnings sur le risque encouru, etc. Ce travail ne propose pas de langages pour la spécification de ces risques au niveau conceptuel et physique.
Enfin, bien que le travail de [Salehi, 2009] ne précise pas à quelle composante SOLAP (données, agrégation, ou requêtes) se rapportent les contraintes d'hypercellabilité, qui définissent des combinaisons interdites de mesures et de niveaux d'agrégation, ces contraintes pourront être considérées comme un type de contraintes de requêtes.

4.7 Evaluation des travaux

Dans cette section, nous proposons une évaluation des travaux sur les CI d'ED et d'EDS (cf. Tableau 4.1) en considérant les critères suivants :

Les types de qualité considérés (C1). Comme nous l'avons montré précédemment la qualité d'analyse (S)OLAP dépend de trois types de qualité : la qualité des données entreposées; la qualité d'agrégation et la qualité d'exploration. La considération de ces trois qualités est nécessaire pour avoir des analyses correctes. Dans le tableau 4.1, les valeurs de ce critère sont respectivement QD, QA et QE.

Proposition d'une classification des CI (C2). Les classifications facilitent l'identification des CI par les analystes. Elles peuvent aussi servir de base à la définition de patrons de conception spécifiques à chaque classe de CI. Ces patrons intégrés dans des AGL vont faciliter la modélisation conceptuelle et la validation des contraintes et permettre leur implémentation automatique dans différents niveaux de l'architecture SOLAP (pour chaque patron, il est possible de définir les règles de transformation du niveau conceptuel vers des représentations physiques dans différentes couches de l'architecture SOLAP).

Nombre de types de CI considérés (C3). La qualité de l'analyse SOLAP dépend forcément de toutes les CI pouvant être définies à différents niveaux de l'architecture SOLAP. La valeur (élevé, moyen, réduit) de ce critère concerne le nombre de classes de CI considérées

Niveau d'abstraction (C4). La plupart des travaux sur les CI préconisent leur définition au niveau conceptuel. Ceci facilite leur validation par les experts du domaine.

Utilisation de standards pour spécifier les CI et le modèle multidimensionnel (C5). L'utilisation des langages standards comme UML et OCL (qui sont intégrés dans de nombreux AGL) pour la spécification conceptuelle des CI (S)OLAP et du modèle (spatio)-multidimensionnel rend plus facile leur spécification et permet leur implémentation automatique.

Utilisation de notations visuelles pour exprimer certaines CI (C6). Dans certains cas, les notations visuelles améliorent la lisibilité des modèles conceptuels et facilitent l'interaction avec les décideurs et spécialistes du domaine [Papajorgji et al., 2010]

Implémentation automatisée des CI (C7). La formalisation basée sur des langages standards de règles de mapping entre le modèle conceptuel et les représentations physiques possibles dans différentes plateformes d'implémentation et leur implémentation au sein de générateurs de code ou bien l'utilisation de générateurs de code existants (comme préconisé par l'approche MDA [OMG, 2003b] par exemple) est nécessaire pour une implémentation automatique des CI (S)OLAP. L'évaluation d'un travail selon ce critère est considérée satisfaisante si le travail formalise des mappings avec des standards ou utilise des générateurs de code pour une implémentation automatique des CI.

Comme le montre la deuxième colonne du tableau 4.1 (critère C1), aucun travail ne considère tous les trois types de qualité : qualité de données (QD), qualité d'agrégation (QA) et qualité d'exploration (QE). L'évaluation du troisième critère (C3) montre que [Salehi, 2009] est le travail qui considère le plus grand nombre de types de contraintes concernant les qualités de données et d'agrégation. La catégorisation de CI proposée néanmoins nécessite d'être étendue par d'autres types de CI comme les CI de données sur plusieurs hypercubes, les CI d'exploration et les CI de métadonnées. A l'exception de [Sapia, 1999; Böhnlein et al., 2002; Bimonte et al., 2009; Prat et al., 2010], tous les travaux définissent les CI au niveau conceptuel (C4), i.e. sur des modèles conceptuels. La plupart des travaux expriment les contraintes en utilisant des langages non standards (logiques, naturels, visuels ou hybrides) qui sont difficiles à implémenter (C5/CI). Dans le domaine des EDS, seul [Pinet et Schneider, 2010] utilise un langage standard de spécification de CI au niveau conceptuel (OCL) mais n'exprime que quelques types de CI de données (C3). Dans les ED, seulement [Prat et al., 2010] utilisent un langage standard, mais pour exprimer que certains types de CI au niveau métamodèle (C3 et C4). Les travaux de [Abelló et al., 2006; Glorio et Trujillo, 2008] se basent sur des profils UML et n'exploitent pas des langages dédiés à la spécification des CI; par conséquent ils n'expriment que très peu de contraintes (C3) : [Abelló et al., 2006] ne modélise qu'un type de CI d'agrégation (les fonctions d'agrégation applicables aux mesures le long des dimension – cf. Section 4.5.2) et [Glorio et Trujillo, 2008] spécifie des CI de données très simples (CI de domaine de valeurs des attributs et de multiplicité – cf. Section 4.4). Enfin, aucun travail ne propose une implémentation automatisée des CI (S)OLAP (C7).

Référence	C1	C2	C3	C4	C5		C6	C7
					CI	Modèle		
[Sapia, 1999]	QE	Non	Réduit	Méta	Non (LN)	Non	Non	Non
[Carpani et Ruggia, 2001]	QD	Oui	Moyen	Conc	Non (LHV, LL)	Non	Oui	Non
[Böhnlein et al., 2002]	QE	Non	Réduit	Méta	Non -	Non	Non	Non -
[Ghozzi et al., 2003a]	QD	Oui	Moyen	Conc	Non (LHN, LHV)	Non	Oui	Non
[Abelló et al., 2006]	QA	Non	Réduit	Conc	Oui (UML)	Oui (UML)	Non	Non
[Prat et al., 2010]	QA	Oui	Moyen	Méta	Oui (PRR)	Oui (UML)	Non	Non
[Glorio et Trujillo, 2008]	QD	Non	Réduit	Conc	Oui (UML)	Oui (UML)	Oui	Non
[Malinowski et Zimányi, 2008]	QD	Non	Réduit	Conc	Non (LHV)	Oui (E/R)	Oui	Non
[Bimonte et al., 2009]	QA	Non	Réduit	Méta	Non (LN)	Non	Non	Non
[Pinet et Schneider, 2010]	QD	Non	Moyen	Conc	Oui (OCL)	Oui (UML)	Non	Non
[Salehi, 2009]	QD + QA	Oui	Elevé	Conc	Non (LHN, LNC)	Non	Oui	Non

Tableau 4.1. Evaluation des travaux existants sur les CI dans les ED et les EDS.
QD : Qualité de Données; QA : Qualité d'Agrégation; QE : Qualité d'Exploration; Conc : Conceptuel; Méta : Métamodèle; LN : Langage Naturel; LNC : Langage Naturel Controlé; LHN : Langage Hybride Naturel; LL : Langage Logique; LHV : Langage Hybride Visuel.

Finalement, ces travaux peuvent être améliorés en termes de : (1) de classification de CI, (2) d'utilisation de standards pour exprimer les CI au niveau conceptuel et (3) d'implémentation automatique des CI.

4.8 Conclusion

Dans ce chapitre nous avons abordé la qualité d'analyse dans les systèmes SOLAP, *qui a été définie comme l'exactitude et la validité sémantique des résultats d'analyse*. Il a été démontré que cette qualité dépend de trois types de qualité : la qualité des données entreposées, la qualité d'agrégation et la qualité d'exploration. S'agissant du premier type, une distinction entre la qualité interne des données spatiales, et leur qualité externe qui dépend des attentes des utilisateurs, est faite dans le but de recadrer la problématique sur la qualité interne. Les définitions *ISO 19113:2002* des critères quantitatifs (e.g. exhaustivité, précision spatiale, cohérence logique, etc.) et qualitatifs (e.g. généalogie) de cette qualité ont été ensuite présentées. La cohérence logique sur laquelle portent nos travaux est définie comme *l'absence de contradictions dans le jeu de données en rapport aux règles logiques présentes dans sa spécificatio*n. Il a été ensuite rappelé que la qualité d'agrégation dépend de conditions structurelles liées aux types de relations fait-dimension et hiérarchies (e.g. strictes ou non strictes) qui supportent l'agrégation, mais également de conditions sémantiques, auxquelles nous intéressons dans cette thèse, et qui vérifient essentiellement la cohérence sémantique entre la mesure, la fonction d'agrégation et la hiérarchie de dimension. De la même manière, nous avons défini la qualité d'exploration en termes de cohérences structurelle et sémantique

des requêtes SOLAP; en soulignant que cette dernière (cohérence sémantique) représente un problème non encore résolu.
Après cela, les Contraintes d'Intégrité ont été définies comme étant une a*pproche courante pour vérifier la cohérence logique*, en soulignant l'intérêt de les définir au niveau conceptuel. Une classification des langages de spécification conceptuelle des CI est détaillée qui donne un net avantage aux langages standards que sont OCL et Spatial OCL, langages formels, déclaratifs et faciles à implémenter.
Ce chapitre se termine par un état de l'art et une évaluation des travaux existants sur la définition des contraintes d'intégrité dans les systèmes OLAP et SOLAP pour vérifier les trois types de qualité. Suite à cette évaluation, il a été démontré essentiellement, le manque d'exhaustivité des classifications de CI proposées (les classifications existantes peuvent être étendues par d'autres classes), le manque d'utilisation de langages de spécification conceptuelle de CI standards et surtout le manque d'implémentations effectives et automatiques des CI.

Dans la suite de ce mémoire, nous allons tenter de répondre à ces limites, premièrement en proposant un modèle spatio-multidimensionnel basé sur UML qui permet l'expression facile des classes de CI que nous avons identifié (cf. Chapitre 6), ensuite la proposition d'un cadre conceptuel basé sur UML et Spatial OCL pour la spécification conceptuelle des CI et un cadre de traduction automatique (cf. Chapitres 5 et 7) des CI conceptuelles vers des implémentations effectives dans l'architecture choisie (cf. Section 3.3.5). Le choix de présenter le cadre générale de nos propositions (Chapitre 5) avant le profil UML (Chapitre 6) est fait dans le souci d'une meilleure compréhension de nos contributions.

L'expérimentation de ces contributions dans le cadre de projets agri-environnementaux sera décrite plus loin dans les chapitres 8 et 9.

PARTIE II : CONTRIBUTIONS

5 Chapitre 5 : A UML and Spatial OCL based approach for handling quality issues in SOLAP systems

5.1 Preamble

Spatial Data warehouses and Spatial OLAP systems are Business Intelligence technologies allowing efficient and interactive analysis of large geo-referenced datasets. In such a kind of systems the goodness of analysis depends on: the warehoused data quality, how aggregations are performed, and how warehoused data are explored. In this chapter, we propose a framework based on a UML profile and OCL-defined integrity constraints to grant quality in the whole SOLAP system. We also propose an automatic implementation in a classical ROLAP architecture to validate our proposal.

5.2 Introduction and motivation

The heterogeneity of data sources in Spatial Data Warehouse (SDW) and Spatial OLAP (SOLAP) systems may lead to several data quality problems [Boulil et al., 2011]. In order to grant data quality in SDW, some approaches have been proposed to “repair” data by means of statistical techniques, data mining techniques, etc. [Ribeiro et al., 2011]. At the same time, Integrity Constraints (IC) have been recognized as effective methods to express rules that control the consistency and completeness of warehoused spatial data [Salehi, 2009]. Moreover, the goodness of spatio-multidimensional analysis also depends on the correct aggregation of measures in respect to summarizability conditions (or aggregation constraints), which check for example that the measure and aggregate function types are compatible [Lenz et Shoshani, 1997]. However, in SOLAP systems the goodness of the analysis also requires another control when exploring (aggregated) data in order to avoid misinterpretation of meaningless SOLAP query results [Levesque et al., 2007], e.g. the query "Sales per country after December 26, 1991" returns empty results for USSR that could be interpreted by users as an absence of sales instead of realizing that a result is impossible for this period. On the other hand, conceptual design of complex systems such as data warehouses has been widely recognized as being necessary for successful BI projects [Malinowski et Zimányi, 2008] since it allows designers defining schemas that are easy to understand by decision makers. In this context, UML (Unified Modeling Language) is widely accepted as the Object-Oriented standard for modelling various aspects of software systems, and also SDW systems [Pinet et Schneider, 2010]. Indeed, any approach using UML minimizes the efforts of designers and decision-makers in developing and implementing the data schema. It can be also interpreted by CASE tools. In the same way, defining IC at a conceptual level allows handling quality issues at the early stages of development [Boulil et al., 2011], minimizing implementation efforts. In this context, [Ghozzi et al., 2003a] propose ad-hoc conceptual multidimensional models allowing the expression of some data IC by means of logical predicates. [Malinowski et Zimányi, 2008] propose an extension of the ER model for the design of spatio-temporal data warehouses. They define a set of ad-hoc pictograms to express spatial data IC (i.e. spatial topological relationships between spatial members). [Glorio et Trujillo, 2008] propose a UML profile for SDW, but they consider a very small number of data IC. A survey on aggregation issues is presented in [Mazón et al., 2009]. They express simple structural constraints (e.g. facts should be linked to dimensions with one-to-many associations) with UML multiplicities. In

[Pinet et Schneider, 2010], complex structural aggregation constraints are expressed with Object Constraint Language (OCL). OCL represents an effective solution to define data IC at the conceptual level in a clear, non-ambiguous and platform-independent way. Indeed, [Boulil et al., 2011] present the definition, on the top of a UML-based SDW conceptual model, of a large number of data IC on warehoused spatial data by means of Spatial OCL, which is an extension of OCL for spatial data [Pinet et al., 2007]. They also propose an automatic implementation in the Spatial DBMS Oracle Spatial 11g. [Levesque et al., 2007] propose a framework for identifying quality risks in ETL, and SOLAP systems. They define 3 types of quality problems (data sources, OLAP data cubes and GIS functionalities) and define them by means of paper forms. They also propose an implementation in the JMap SOLAP system.

Finally, to best of our knowledge, no work proposes a unique framework to express at the conceptual abstraction level IC on spatial warehoused data, aggregation, and spatio-multidimensional queries, and their automatic implementation in a classical ROLAP architecture.

Thus, in this chapter we present three main contributions.

For first, we extend/reformulate the definition of (S)DW IC for handling quality issues in SOLAP systems; we use IC to perform three quality control types:

(a) Data quality control ensures that warehoused spatial data are valid (e.g. geometries of cities must be topologically included in the geometries of their states);

(b) Aggregation quality control ensures that aggregations of measures are correct and meaningful (e.g. the sum of the unit prices does not make sense) [Lenz et Shoshani, 1997];

(c) SOLAP exploration control avoids problems of interpretation induced by meaningless SOLAP query results (e.g. sales in USSR after 26 December 1991) (misuse data cube risks as defined by [Levesque et al., 2007]).

Secondly, motivated by a lack of a unique conceptual framework to define SOLAP IC, we propose a UML-OCL based conceptual framework. Finally, we propose an automatic implementation of such framework in a classical Relational SOLAP architecture.

5.3 SOLAP IC classification

In this section, we present an extension of our previous SDW IC classification [Boulil et al., 2011] by introducing a new class, Query IC class. This classification (Figure 5.1) serves as a reference guide for the process of handling the three types of quality issues in a SOLAP system.

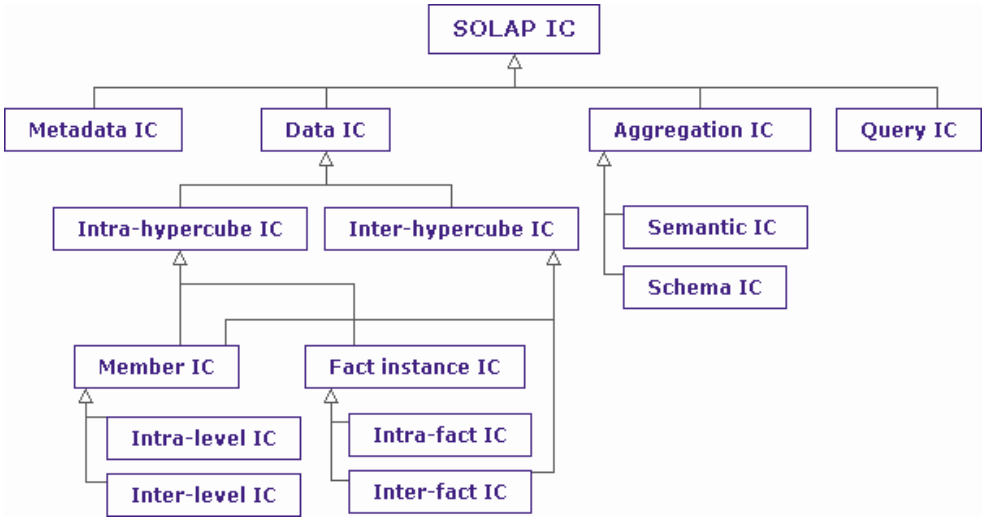


Figure 5.1. A SOLAP IC classification.

Before detailing the classification, we present the case study which will be used all along the chapter to describe our proposal. It concerns an environmental SDW, with a *temporal dimension* that groups days into months and months into years, and a *spatial dimension* representing cities with their regions and countries. The measure is the *temperature value*. Using this SDW, decision-makers can answer to SOLAP queries like these: *"What is the minimal temperature per year and country?*" or "*What is average temperature per month and country?*". In order to answer these queries, decision-makers use the min and average aggregate functions to aggregate the temperature values.

Let us now provide explanations and some examples of these IC classes using the previously described case study.

As shown in [Boulil et al., 2011], *Metadata IC* verify the consistency of metadata of different integrated data sources (e.g. spatial members and measures must be defined with the same geographic scale).

Data IC ensure the logical consistency and completeness of warehoused spatial data, for example:

Example 1: "*the geometry of each city must be topologically included in the geometry of its region*" or

Example 2: *"no facts (e.g. temperature values) should exist for USSR after 26 December 1991".*

These constraints can be defined on all elements of the SDW such as facts, members, etc.

Aggregation IC guarantee correct and meaningful aggregations of measures. In particular, *semantic constraints* address the problem of the applicability of aggregate functions to measures according to the semantic nature and the type of measures, aggregate functions and dimensions. For example:

Example 3: "*Sum of temperature values does not make sense in some applications*"

Schema constraints are conditions that must be satisfied by dimension hierarchies and dimension-fact relationships to avoid double counting and incomplete aggregates. For example, dimensions and facts should be linked by one-to-many relationships [Mazón et al., 2009].

Query IC refer to conditions that guarantee that SOLAP queries are valid in the sense that their results are not always empty in order to avoid problems of misinterpretation. For example, the SOLAP query "*What are the average temperatures in USSR in 2010?*" returns an empty result since no temperature value is stored for USSR after 26 December 1991 (the previous data IC of Example 2). Even if this IC is implemented as data IC, classical SOLAP tools allow decision-makers to formulate this query by combining these two members (USSR

and 2010) returning an empty value. This leads to a problem of interpretation: this empty value may be perceived as if there were no temperature values registered for USSR during 2010, instead of realizing that this combination of members (USSR and 2010) is invalid. Consequently, to avoid this misinterpretation we define the following query constraint:
Example 4: *"It is incorrect to combine USSR with days after 26 December 1991 in a SOLAP query"*.
Although, this query example could be resolved by using particular spatio-multidimensional data structures such as DW versioning structures, Query IC allow designers to model any invalid query which can be independent of time-versioning aspects (for example, some products cannot be sold in certain stores).

5.4 The conceptual framework

Before describing our conceptual framework for defining SOLAP IC, we present main concepts of a UML profile and Spatial OCL.
The UML profiles are a way to customize UML for particular domains or platforms by extending its metaclasses (class, property, etc.). A profile is defined using three extension mechanisms: stereotypes, tagged values and constraints. A stereotype is an extension of a UML metaclass. Tagged values are properties of stereotypes. Finally, a set of OCL constraints precise each stereotype's application semantics. OCL provides a platform-independent method to model constraints. It can be interpreted by code generators to generate code automatically. OCL constraints can be defined at the meta-model level (e.g. UML profile) and also at the model level (the profile instance). Spatial OCL is an extension of OCL that supports spatial topological relationships (inside, intersect, etc.) [Pinet et al., 2007].
In order to define SOLAP data, aggregation and query IC at a conceptual level, we propose a framework based on a UML profile and Spatial OCL (Figure 5.2).
The main idea is to a have a unique UML profile that defines 3 interconnected models to conceptually represent:

a) SDW data structures (*SDW model*),
b) how measures are aggregated to meet the analysis requirements (*Aggregation model*), and
c) *Query IC model*.

And then define IC with Spatial OCL using these models. In particular Data IC are defined by designers using Spatial OCL on the top of the instance of SDW model, Aggregation IC are defined as Aggregation model's stereotypes constraints using OCL, and Query IC are defined using the Query IC model and Spatial OCL. Due to space reasons we do not detail the proposed profile, but we provide some examples. Details on the SDW and aggregation models can be found in [Boulil et al., 2011]. It is important to note that we have chosen to define a UML profile instead of a metamodel since the UML metamodel's elements are sufficient to capture all the SOLAP applications' semantics including all the multidimensional data structures [Glorio et Trujillo, 2008] and all the identified IC types.

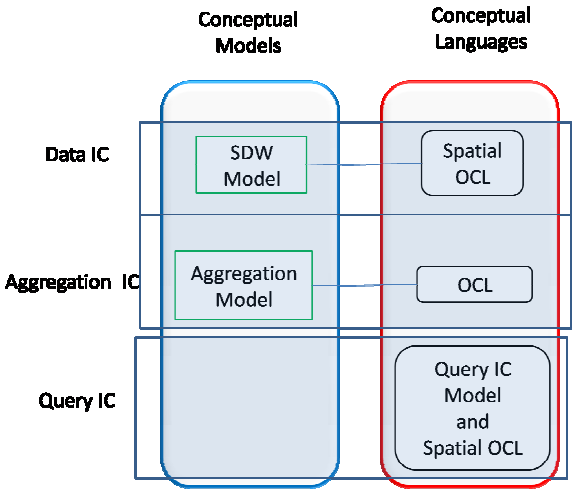


Figure 5.2. The proposed UML-OCL based conceptual framework.

The *SDW model* allows the definition of SDW data structures and the expression of Data IC on the top of these structures using Spatial OCL [Boulil et al., 2011].

The SDW case study represented using the *SDW model* is shown on Figure 5.3. This *SDW model* instance contains two dimensions: (i) a spatial dimension composed of 3 spatial levels (stereotyped as <<SpatialAggLevel>>), *City*, *Region* and *Country*; and (ii) a temporal dimension composed of three temporal levels *Day*, *Month* and *Year*. The numerical measure *temperature* (<<NumericalMeasure>> stereotype) is defined as an attributed of the fact class *Temperature* (<<Fact>> stereotype).

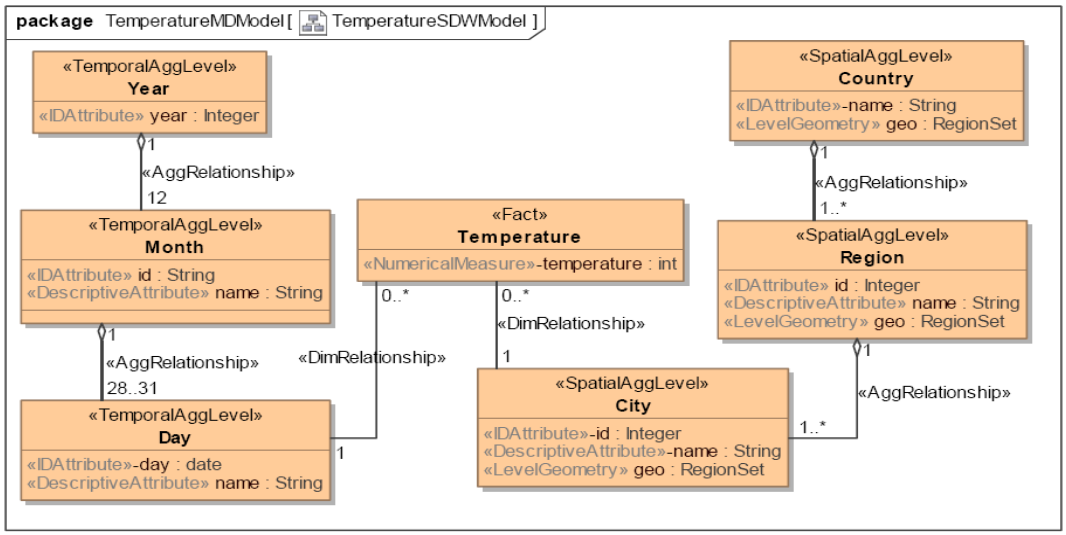


Figure 5.3. A SDW model instance.

Once the SDW model instance has been defined, data integrity constraints can be expressed using Spatial OCL. For example, the Data IC of Example 1 is expressed as follows:

```
context Region inv DataIC1:
self.geo.isInside(country.geo) or
self.geo.coveredBy(country.geo)
```

The Data IC of Example 2 is expressed using OCL in the following way:

```
context Temperature inv DataIC2:
not (
 self.day.day >= '1991-12-26' and
 self.city.region.country.name = 'URSS'
)
```

The *Aggregation model* represents how measures are aggregated along dimensions according to decision-makers' analysis needs. The instance of Aggregation model for our case study, which represents that the *temperature* measure (*aggregatedAttribute* tagged value) is aggregated along all the dimensions using the average aggregate function (*aggregator=Avg* tagged value), is depicted in Figure 5.4.

In [Boulil et al., 2011] we have identified a set of aggregation constraints that grant meaningful aggregations of measures. These constraints are valid for all SOLAP applications. Thus, we have implemented them as OCL constraints in the Aggregation Model package of the profile. They are checked by the CASE tool at the design stage when validating the conceptual model.

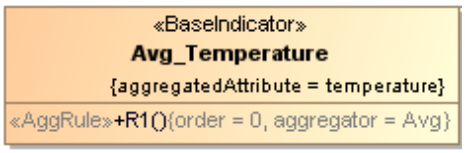


Figure 5.4. An example of an aggregation model instance

For example, in order to force the user to not aggregate non-additive (or value per unit) measures (for example the temperature; Example 3) using the sum aggregate function, the following OCL statement is defined in the profile:

```
context AggRule inv notSumValuePerUnitMeasure:
if(
 baseIndicator.aggregatedAttribute.OclIsKindOf(Measure)
 and baseIndicator.aggregatedAttribute.addType =
  'ValuePerUnit'
) then aggregator.name <> 'Sum'
```

Finally, designers can express IC on SOLAP queries using the *Query IC model*. Typically, a SOLAP query is a combination of measures and members from different dimensions. Thus, the Query IC model can be used for example to define invalid combinations of member sets. These member sets are specified as attributes with the <<MemberSet>> stereotype. The value domain of a <<MemberSet>> attribute is a subset of members of a dimension level, whose definition is specified with the *condition* tagged value, which is an OCL statement defined on the context of the dimension level to select a subset of its members.

An example of an instance of the Query IC model is depicted on Figure 5.5, where the user states that combining days (<<MemberSet>> *day*) after 26 December 1991 (*condition= After1991-12-26*, whose OCL expression is shown in Figure 5.6) with the USSR (<<MemberSet>> *country*) is meaningless in any SOLAP query.

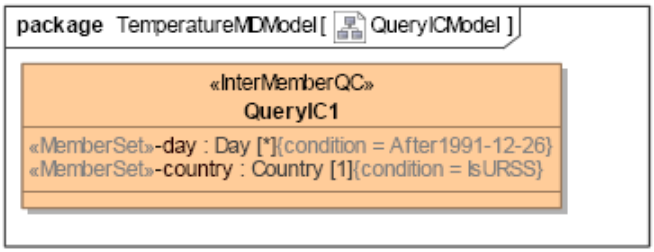


Figure 5.5. A query IC model instance.

```
context Day inv After1991-12-26:
self.day >= '1991-12-26'
```

Figure 5.6. The OCL used by the Query IC of Figure 5.5.

5.5 The implementation framework

In this section, we present our architecture to automatically implement SOLAP IC (Figure 5.7). The main idea is to automatically implement each kind of IC in a different tier of the SOLAP architecture. The conceptual definition of each IC is automatically translated into the implementation language used by each tier. In particular, Data IC are translated using SpatialOCL2SQL and implemented in the SDW tier; Query IC are translated by our automatic code generator (called UML2MDX) and implemented in the OLAP server and the SOLAP client, and finally Aggregation IC are implemented in our UML profile using OCL and controlled during the design stage by the MagicDraw CASE tool.

Our SOLAP architecture (Figure 5.7) is based on: the Spatial DBMS Oracle Spatial 11g, the ROLAP Server Mondrian and a SOLAP client JRubik. Mondrian connects to a relational database and enables the execution of OLAP queries expressed using MDX (MultiDimensional eXpressions language) that is a standard language for querying multidimensional databases. JRubik provides a graphical presentation layer on top of Mondrian and allows cartographic representations of OLAP queries using the SVG format.

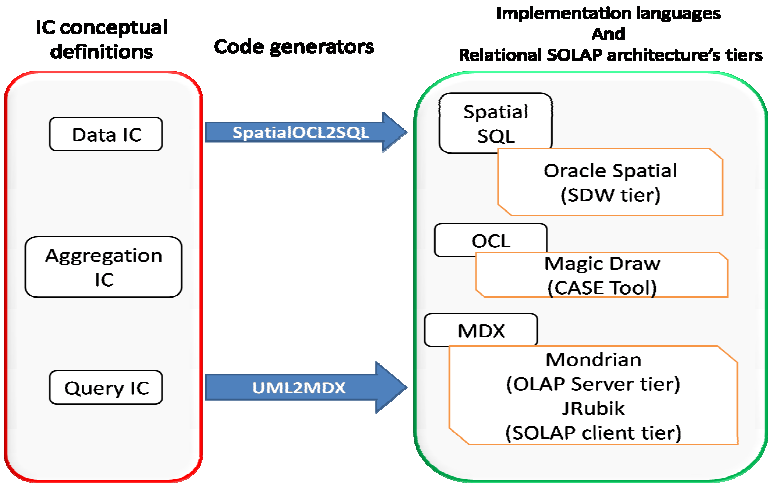


Figure 5.7. The proposed implementation framework for SOLAP IC.

In order to automatically implement data IC in the Oracle Spatial 11g, we have used the code generator Spatial OCL2SQL. Spatial OCL2SQL is a Java open source tool which integrates the spatial extensions of OCL called OCL 9IM and OCL ADV [Pinet et al., 2007]. It automatically generates SQL scripts for Oracle Spatial from Spatial OCL conceptual constraints.

In our case study, the previously defined OCL data IC of Example 2 is transformed in the following SQL query:

```
select * from TEMPERATURE SELF where not (
not (
 (select DAY from DAYS where DAY_PK in
  (select DAY_FK from TEMPERATURE where
   TEMPERATURE_PK = SELF.TEMPERATURE_PK)
 ) >= 19911226 and
 (select COUNTRY_NAME from CITIES where CITY_PK in
  (select CITY_FK from TEMPERATURE where
   TEMPERATURE_PK = SELF.TEMPERATURE_PK)
 ) = 'URSS'
));
```

This query selects the facts (TEMPERATURE table's tuples) that do not satisfy the constraint of Example 2.

The Aggregation IC are implemented as OCL profile inherent constraints in the MagicDraw CASE tool. MagicDraw supports OCL at the meta-model level (UML profile). In other terms, MagicDraw is able to check OCL constraints defined on UML stereotypes. This allows checking Aggregation IC at design stage independently of the specific SOLAP architecture used and without providing any implementation efforts. For example, if the designer defines an instance of the Aggregation model by using the Sum for the temperature measure, MagicDraw checks the OCL Aggregation IC of Example 3 and informs him that the constraint is violated.

In order to implement Query IC, we use MDX, which is the defacto standard of OLAP Servers and Clients. Thus, the choice of Mondrian as OLAP server is not a limitation for our generic architecture. The main idea is to translate the Query IC into MDX formula, which are stored in the OLAP Server and then visualized in the SOLAP client. These formulas, when executed, inform user about the quality of query results. For each Query IC type we have defined an MDX template. The templates are fulfilled using a Java method (*UML2MDX*) that parses the XMI files associated to the Query IC. Different visual policies are associated with different combinations of members from these sets to be displayed in the SOLAP client tier: green colour for valid cells, yellow colour for aggregated cells that include valid and invalid cells and red colour for invalid cells. Figure 5.8 shows an example of OLAP query where these visual policies are applied according the MDX formula implementing the Query IC of Figure 5.5: valid cells such as those combining USSR with dates before 1991-12-26 (e.g. 1991-12-01) are displayed with green colour; invalid cells that involve for example USSR and dates after 1991-12-26 (e.g. 1991-12-27, 2010-1, 2010) are displayed with red colour, other cells are displayed with yellow colour, such as 1991-12 with USSR because it is the aggregation of valid (e.g. 1991-12-01 with USSR) and invalid cells (e.g. 1991-12-27 with USSR).

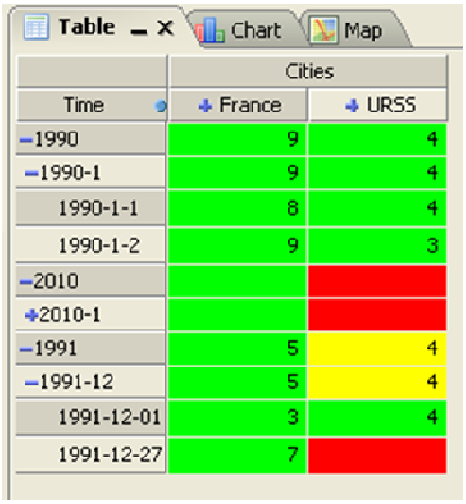


Figure 5.8. An example of a query IC verification of Example 4.

5.6 Conclusion

In this chapter, we first show that the SOLAP analysis goodness depends on 3 quality types: data, aggregation and query qualities. Thus, we (i) extend the concept of integrity constraints to consider all these quality types; (ii) propose a framework based on a UML profile and Spatial OCL to express these SOLAP IC at the conceptual level; and (iii) show their automated implementations in a typical ROLAP architecture. Our current work is on improving the UML2MDX tool by integrating Spatial MDX expressions and defining cartographic-related visualization policies in order to implement spatial query IC.

As in our current automatic implementation only considers the snowflake schema SDW implementations, we are working on the consideration of the star-schema implementations. Finally, we will work on the formal validation of the completeness of our classification, and the expressiveness of our conceptual framework.

6 Chapitre 6 : A UML profile and OCL-based constraints for spatial data cubes

6.1 Preamble

The conceptual modeling of Data Warehouse (DW) decisional systems requires the definition of two types of metadata: (i) warehouse metadata that models data structures that maintain integrated data from multiple data sources and (ii) aggregation metadata that specify how the warehoused data should be aggregated to meet the analysis goals of decision makers. In the literature, many MultiDimensional (MD) models have been proposed for the conceptual design of these systems; however, all of the models focus on warehouse modeling and ignore the data aggregation aspects that are fundamental to these systems. These models present also some limitations regarding the handling of some of the critical issues, such as data and aggregation qualities and multi-granular measures.

In this chapter, motivated by the lack of a standards-based formalism for the conceptual design of (spatial) data cubes that takes into account both the warehouse and aggregation MD model parts, we propose a new UML 2 (Unified Modeling Language) profile. In additional to classical modeling needs, our profile satisfies new requirements: (a) modeling of indicators and their aggregation rules, (b) aggregate functions' typing, (c) support for multi-granular measures and (d) support for data and aggregation quality issues. We also propose an interesting approach that addresses some aspects of the data aggregation quality at the conceptual abstraction level by defining the main semantic aggregation constraints identified in the literature using the Object Constraint Language (OCL). We finally propose an automatic implementation of the aggregation part in terms of a Mondrian OLAP schema using a Java-based tool. Our profile and its OCL-based constraints have been fully validated by a complete implementation in the MagicDraw UML-based tool.

6.2 Introduction

At the conceptual abstraction level, DW and SDW systems are modelled using the MultiDimensional (MD) model [Nguyen et al., 2000a] and the spatial MD model [Stefanovic et al., 2000], respectively. These models organise (in terms of *measures* and *dimensions*) decisional data multidimensionally according to the analytical needs of decision makers. Therefore, they represent an important means to exchange and to collaborate between decision makers and designers for attaining a high achievement with the decisional project, which depends on meeting the users' analysis goals [Torlone, 2003].

As stated in many studies [Lenz et Shoshani, 1997; Ghozzi et al., 2003a; Salehi, 2009; Boulil et al., 2010a; Pinet et Schneider, 2010], data quality and aggregation are very important issues in (S) DW applications because the goodness of the analysis depends on both the validity of the warehoused data and the way that these data are aggregated. For addressing the issues, Integrity Constraints (ICs) have been recognised by these works to be an excellent approach. ICs define conditions that should be satisfied by data sets (from the database field); they are used to improve the accuracy, consistency, and completeness of databases. In this chapter, we consider an extended definition of IC for (S)DW systems; we use ICs to control both the integrity of warehoused data (e.g. cities must be topologically included in their states) and the way that decision makers aggregate data to avoid incorrect and meaningless aggregates (e.g. the sum of the unit prices does not make sense) [Salehi, 2009; Boulil et al., 2010a]. ICs are often defined in conceptual models to allow handling these critical issues starting at the early stages of development.

In the literature, many (spatial) MD models have been proposed for DW [Abelló et al., 2006; Kamble, 2008] and SDW [Bimonte et Pinet, 2010] applications. However, until now, no standard model has emerged for these applications.
Several authors proposed Unified Modeling Language (UML)-based models for the conceptual modeling of such applications [Abelló et al., 2006; Lujan-Mora et al., 2006; Pinet et Schneider, 2010]. There are many reasons for this. First, UML provides designers with powerful constructs to better represent static and dynamic aspects (such as aggregation and data quality) of these applications. UML can be easily extended using profiles to handle specific complex aspects of these applications (e.g. security and multi-versioning). After its adoption by the software designer community, a growing number of UML-based tools emerged, allowing for easier and quicker development. Finally, by supporting the Object Constraint Language (OCL), UML-based models easily handle the data quality issues of the DW and SDW systems [Boulil et al., 2010a].

In general, all of the existing conceptual MD models whether spatial or not, standards-based or not, focus on the design of the warehouse structures and ignore the aspects related to the aggregation of data that are central in (S)DW applications. In addition to some limitations concerning the modeling of the warehouse part (e.g. multi-granular measures and data quality), these models do not provide the appropriate support for the conceptual representation of dynamic aspects related to the aggregation of measures (indicators and their aggregation rules).
Moreover, some models cannot be used to design these systems because they do not provide designers with visual notations [Torlone, 2003]. Others are ad hoc models and require more learning and modeling efforts.

6.3 Contributions and organization of the chapter

In this chapter, motivated by the lack of a standard framework for representing static (warehouse) and dynamic (aggregation) aspects of (spatial) data cubes, we propose a new UML profile. This profile and its OCL-based constraints have been fully validated by a complete implementation in the UML-based tool called MagicDraw.
Our profile meets the main classical modeling requirements that are identified in the literature (e.g. explicit complex hierarchies, measure sets and aggregate functions); see [Abelló et al., 2006; Viswanathan et Schneider, 2011]. Additionally, it meets some new important mandatory requirements: *(a) modeling of indicators and their aggregation rules, (b) aggregate functions' typing (additivity and applicability types), (c) support for multi-granular measures (measure ETL semantics and disaggregation functions), and (d) support for data quality and aggregation quality issues.*
Moreover, exploiting the aggregation support and the precise typing of MD elements provided by our profile, *we propose an approach that addresses some aspects of the data aggregation quality at the conceptual abstraction level.* The main semantic aggregation constraints identified in the literature are defined using OCL as part of our profile. This definition enables semantic correctness for the aggregation models that are defined by designers to be controlled at the conceptual level before generating their implementations.
Finally, we propose the automatic implementation of the aggregation MD model part in terms of a Mondrian OLAP server schema using a Java-based tool developed in the Eclipse Integrated Development Environment (IDE). *The novelty of our implementation proposal is the implementation of complex indicators and their aggregation rules in terms of MultiDimensional eXpressions language (MDX) based formulas.*

The remainder of this chapter is organised as follows. In Section 6.4, the retail case study used for illustration throughout the chapter is described. In Section 6.5, classical and new modeling requirements for spatial data cubes are presented. Section 6.6 surveys and discusses related work, and Section 6.7 presents our UML profile for spatial data cubes. In Section 6.8, we show how data and aggregation qualities are handled in our proposal; a focus is on the definition of semantic aggregation constraints that avoid incorrect and meaningless aggregates. In Section 6.9, the automatic implementation of the aggregation MD model part (indicators and aggregation rules) on the top of the Relational OLAP (ROLAP) architecture is detailed. Finally, conclusions and future work are outlined in Section 6.10.

6.4 Case study

To illustrate SOLAP/OLAP concepts and modeling requirements, we introduce a retail case study adapted from [Malinowski et Zimányi, 2008]. In this model, decision makers are interested in the analysis of two aspects of their activity: (1) the product supplies and (2) the product sales. In terms of OLAP, these aspects can be defined using two hypercubes (i.e. two data cubes): the "Supplying" and the "Sales" hypercubes. The Sales multidimensional model is depicted in Figure 6.1 using a graph notation. As shown in this figure, sales facts are described using five numerical measures: "price" (sales amounts), "quantity-sold" (quantities of sold products), "quantity-stock" (stock quantities of products), "costs" (sales costs) and "unit-price" (unit sales prices). The sales are analysed according to five dimensions, describing three different types of extents:

- two thematic dimensions, "Orders" and "Products", containing only thematic information;
- two spatial dimensions, "Customers" and "Stores", containing geographic and thematic information;
- one temporal dimension, "Time", representing the time frames of facts (sales).

The dimension "Orders" records the order identifiers. The dimension "Products" records thematic information about the products and their categories. The spatial dimension "Stores" records the stores and their locations. The dimension "Customers" stores the customers and their addresses. Dimensions are described using different types of attributes, called dimensional attributes. For example, the spatial dimension "Stores" (Figure 6.2) contains thematic attributes such as "store-name" and "store-manager" and spatial attributes such as "country-geometry", which describes the locations of countries. The "Time" dimension is defined by temporal attributes (e.g. "month-temporality") and thematic attributes such as "month-name".

The aggregation levels are formed by grouping dimensional attributes into disjoint subsets according to the analysis requirements. For example, the spatial dimension "Stores" (Figure 6.2) consists of one hierarchy ("StoreLocation") that is composed of four aggregation levels: "Store", "City", "State" and "Country".

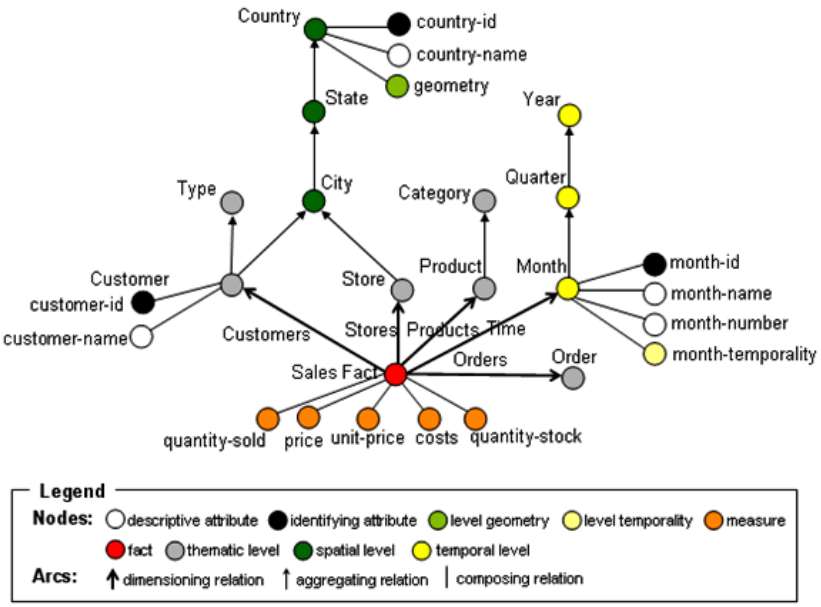


Figure 6.1. The sales hypercube – Schema level.

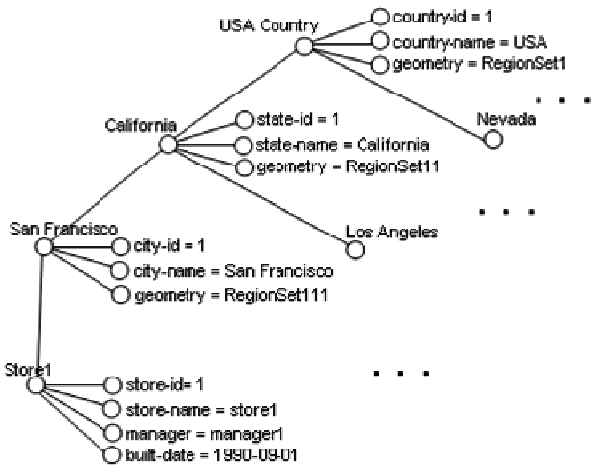


Figure 6.2. The Stores dimension – Example of instance.

Once the warehouse MD model (i.e. dimensions and measures) is defined, the decision makers decide how to aggregate data to compute different indicators of analysis in the SOLAP tier. In our case study, decision makers may want, for example, to compute the indicators, the total of sold quantities by summing the "quantity-sold" measure values or the turnover by summing the "price" values.

Finally, this spatial data cube allows users to answer queries such as "What are the turnovers per store or per product between 2000 and 2011?" and "What are the total costs per state?". These queries are visualised in the SOLAP client tier using different types of displays (e.g. tabular and maps). Specifically, a map representing states is used to show the results of the second query.

6.5 Modeling requirements

In this section, we present the main spatial data cube conceptual modeling requirements that are identified in the literature (Section 6.5.1) and propose new ones (Section 6.5.2). We illustrate these requirements using the retail case study previously described.

6.5.1Classical modeling requirements

Modeling requirements for conventional and spatial data warehouses have been investigated in many studies [Abelló et al., 2006; Viswanathan et Schneider, 2011]. These studies suggest that each (S)DW model must conform to the user's view of the decisional data by organising the data into facts and dimensions. This *support for multidimensionality* must be clear, non-ambiguous and independent from the implementation to facilitate understanding of the model and validation of the model by the decision makers.

6.5.1.1 Modeling of measures

In the literature, support for measure sets (for example, sales are described by a set of measures: "quantity-sold" and "price") and multi-granular measures (measures that are acquired at different levels of detail from data sources; for example, in our application one can suppose that sales facts are collected at two different temporal granularities, e.g. month and quarter) are well-known requirements concerning the conceptual modeling of measures.

6.5.1.2 Modeling of dimensions

The main modeling requirements regarding dimensions are support for descriptive attributes, identifying attributes and multiple hierarchies [Abelló et al., 2006; Malinowski et Zimányi, 2008; Pinet et Schneider, 2010]. The identifying attributes identify dimension members and are used for grouping purposes in Roll-up and Drill-down operations. An example of these attributes is "customer-id" (Figure 6.1). The descriptive attributes are used to select and to filter results in Slice and Dice operations (e.g. "country-name").
Dimension hierarchies are widely used structures that support data aggregations [Prat et al., 2010]. Consequently, *representing hierarchies explicitly* is stated as a mandatory requirement. Moreover, as highlighted in [Malinowski et Zimanyi, 2008], designers may want to model different hierarchies per dimension. In our example, the dimension "Customers" could be organised into two hierarchies, "CustomerLocation" and "CustomerType". This requirement is known as *multiple hierarchies per dimension*.

6.5.1.3 Modeling of aggregate functions

Several studies have highlighted the importance of aggregate functions for representing MD queries (data cubes) at the conceptual level [Cabot et al., 2010], thereby allowing validation of the model with respect to the analytical requirements of decision makers at an early stage.

6.5.1.4 Typing of multidimensional elements

A very important requirement for SDWs is the *precise typing of multidimensional elements*. Indeed, according to [Malinowski et Zimányi, 2008; Salehi, 2009], SDW elements that either represent factual or dimensional data can be spatial, temporal and thematic (non-spatial and non-temporal). This requirement allows us to perform the following: (a) to improve readability by associating adequate graphical notations for each type, (b) to ease models' validation (e.g. spatial measures have to be aggregated using only spatial aggregate functions), and (c) to facilitate MDA implementations by defining specific QVT transformations for each type (e.g. a spatial measure must be implemented using a geometric data type into the SDW tier).

6.5.1.4.1 Typing of measures

Measures can be of different categories according to their purposes of use in multidimensional analysis: Numerical, Spatial, Boolean, Textual, and Temporal. Numerical measures are attributes that are used to quantify facts; for example, "quantity-sold" quantifies sales (Figure 6.1). Spatial measures are geometric attributes that should be visualised on

maps to better understand the geographic distribution of facts. Boolean measures are used to record the occurrence/non-occurrence of events [Golfarelli et al., 1998]. Textual measures are qualitative attributes used to analyse textual data [Park et al., 2005; Ravat et al., 2007; Ravat et al., 2008]. Temporal measures hold information of temporal data types, such as time instant and time interval. We refer to this concept as *the measure using type*.
Regarding their additivity, measures are classified into flow, stock and value per unit measures [Lenz et Shoshani, 1997]. Flow measures (e.g. "quantity-sold") record the commutative effect (value changes) over a given time period. Stock measures record the state at specific points in time ("quantity-stock"). Value per unit measures record values per unit in a specific period ("unit-price"). The additivity type of a measure influences its aggregation. Flow measures can be summed (additive) on all dimensions; stock measures are not additive on the time dimension (semi-additive); and value per unit measures are not additive at all. We refer to this concept as *the measure additivity type.*

6.5.1.4.2 Typing of dimensions

The SDW model should *explicitly support thematic, spatial, and temporal dimensions, hierarchies and aggregation levels*. These different types of MD elements influence differently aggregations; for example, stock measures (e.g. "quantity-stock") cannot be summed (using Sum) along *temporal dimensions* ("Time"). Moreover, these MD elements verify different types of ICs [Boulil et al., 2010a]. For example, spatial topological relationships (e.g. geometrical inclusion) can be defined only between spatial levels (e.g. "City" and "State"). Finally, these different types of elements are used differently in SOLAP analysis. For example, only spatial hierarchies support Spatial Roll-up/Drill-down operations.

6.5.1.5 Modeling of complex data structures

Complex data structures are very frequent in real-world situations. Among these structures, we can cite irregular hierarchies (non-strict, non-onto and non-covering) and irregular fact-dimension relationships (non-strict and non-complete); for more details, see [Malinowski et Zimányi, 2008; Mazón et al., 2009].
Other important classical requirements are the separation between structure and content, the representation of multiple hypercubes that share dimensions, and the sharing of levels between hierarchies; see [Abelló et al., 2006; Viswanathan et Schneider, 2011] for more details.

6.5.2 New modeling requirements

In this section, we present new important modeling requirements for spatial data cubes.

6.5.2.1 Modeling of indicators and aggregation rules

Usually, decision makers aggregate measures using different aggregate functions to meet their analysis goals. For example, in our case study, managers may want to compute the maximum sales price by applying the Max aggregator over all dimensions, the total sales price by applying the Sum over all dimensions, and the stock level by applying the Avg along the Time dimension and the Sum along the other dimensions.
To model such different complex situations, we believe that it is necessary to distinguish between the following:

- **the measure** that represents an attribute that is subject to aggregations;
- **the aggregation rule** that specifies the aggregate function that is applicable to a measure along a certain number of dimensions, hierarchies or between two aggregation levels; and

- **the indicator** *that defines an ordered set of correct aggregation rules for a measure in response to an analysis need.*

In this way, we can easily associate multiple indicators for one measure (e.g. "Max_Price" and "Total_Price" with the measure "price"). We refer to this requirement as *explicit indicators*.
Aggregation rules for "Max_Price" and "Total_Price" indicators simply use one aggregate function along all of the dimensions (i.e. Max and Sum, respectively); however, more complex scenarios can be formulated.
Indeed, certain indicators may be defined by ordered sets of aggregation rules, where each rule may specify the aggregate function along a set of dimensions. For example, to compute the indicator "Stock_Level", we must apply Avg along "Time" after applying Sum along other dimensions [Horner et al., 2004]. We refer to these requirements as *multiple ordered aggregation rules per indicator* and *explicit dimension-related aggregation rules*.
Moreover, when dimensions are composed of multiple hierarchies, it is possible to have different aggregate functions for each hierarchy of the same dimension [Salehi, 2009; Prat et al., 2010]. For example, decision makers may want to aggregate the "quantity-sold" using the Sum along the "CustomerLocation" hierarchy and the Avg along the "CustomerType" hierarchy. We call this requirement *explicit hierarchy-related aggregation rules.*
In specific situations, the aggregation rule may change between two aggregation levels of the same hierarchy [Salehi, 2009; Prat et al., 2010]. For example, to obtain the maxima of city's turnovers for every combination of levels, we first sum the "price" values from "Store" to "City" and then apply Max. We call this requirement *explicit sub hierarchy-related aggregation rules*.
Finally, indicators such as "Profit" and "Unit_Price" may be derived from other indicators; "Profit", for example, is calculated by subtracting the "Total_Price" from the "Total_Costs"; we refer to this requirement as *explicit derived indicators*.

6.5.2.2 Typing of aggregate functions

In [Shekhar et al., 2001], SOLAP aggregate functions are classified into three types according to their distributivity: distributive (e.g. Sum and GeometricUnion), algebraic (e.g. Avg and Centroid) and holistic (e.g. Mode and EquiPartition). With distributive aggregators, sub-aggregates can be safely reused to compute coarser ones. Algebraic aggregators are expressed as finite algebraic expressions over distributive aggregators; e.g. Avg is computed using Count and Sum. Holistic aggregators are all of the other aggregate functions that are non-distributive and non-algebraic. We refer to this knowledge as *the aggregator distributivity type*.
Similar to dimensions and measures, aggregate functions can be classified with regard to the data types to which they apply; the categories are numerical (Sum), spatial (GeometricUnion), Boolean (Or), Textual (List [Ravat et al., 2008]), Temporal (TemporalUnion), and generic (Count). We refer to this knowledge as *the aggregator applicability type*.
We state that representing these two characteristics of aggregators (i.e. the distributivity and the applicability types) at the conceptual abstraction level is very important for multiple reasons. The distributivity type can be used to ease the materialised view selection process by illuminating at an early stage the views that are calculated using non-distributive functions. More importantly, it can be only obtained from users[24] when the aggregators are application-specific. The applicability type can be used to check the validity of the aggregation model at

[24] The user information and needs should be represented at the conceptual abstraction level because the languages used at this level are understood by users.

an early stage. For example, indicators associated with spatial measures must be defined using spatial aggregators.

6.5.2.3 Support for multi-granular measures

As in our proposal, measures can be acquired at non-leaf levels, two new requirements emerge: *measure ETL semantics* and *explicit disaggregation functions*.
For example, suppose that the "price" values for 2010 are stored in the SDW at the "Year" level, with the Sum semantics (i.e. they correspond to yearly totals). Then if the decision makers want to compute the "Avg_Price" indicator for 2010, the result will be incorrect because for 2010 the returned value is not the average but the sum. This implies that when aggregate functions of indicators and aggregate functions used to store measures at coarser granularities are different, a disaggregation function should be used to have semantically correct aggregates. For example, users can choose to disaggregate the "price" values in 2010 by dividing them by 12 and then average them.
The *measure ETL semantics and disaggregation function* are the aggregate function used to store measure values at non-leaf aggregation levels and the function used to disaggregate these values, respectively.

6.5.2.4 Data quality and Aggregation quality

Data and aggregation qualities are very important topics in (S)DW systems because these systems are used to support decision-making processes. As shown in [Salehi, 2009; Boulil et al., 2010a], Integrity Constraints (ICs) are a good way to address these issues during early stages of the SDW development process. Consequently, *good support for ICs* at the conceptual abstraction level appears as a new important requirement. This means that the conceptual spatial MD model must explicitly represent all of the elements that may be involved in the different categories of ICs [Boulil et al., 2010a] to allow their conceptual definition. Particularly, by explicitly representing hierarchies, Inter-hierarchy ICs that involve members of two hierarchies or more can be easily expressed. For example, the customer locations (members of the "CustomerLocation" hierarchy) must be spatially disjoint from the store locations ("StoreLocation" hierarchy).
Moreover, as shown in [Salehi, 2009; Boulil et al., 2010a], OCL represents a very adequate language for expressing and implementing spatial data cube ICs. Using OCL and UML allows for automatic MDA-based implementations of (spatial) data cube ICs [Boulil et al., 2010a]. Consequently, the conceptual MD model must support OCL.

6.6 Related work

In this section, we present the most relevant conceptual MD models proposed in the literature for conventional DWs (Section 6.5.1) and SDWs (Section 6.5.2). In particular, we discuss the limitations of spatial multidimensional models according to previously described requirements (Section 6.5.3).

6.6.1Conventional models

In conventional data warehouses, many graphical conceptual models have been proposed in the literature. These models provide graphical support for the main static MD components and are based on either existing frameworks (e.g. UML, ER, and Description Logic) or ad hoc formalisms; see [Abelló et al., 2006; Kamble, 2008] for more details.
The study in [Golfarelli et al., 1998] proposes an ad hoc conceptual multidimensional model, the Dimensional-Fact Model (DFM), which supports the main static MD structures and allows for graphical representations of the measure additivity constraints (referred to as *type*

compatibility conditions in [Lenz et Shoshani, 1997]). In addition, the authors discuss two interesting issues: overlapping hypercubes and aggregation in empty facts.
Studies in [Sapia et al., 1998], [Tryfona et al., 1999], [Malinowski et Zimanyi, 2006] and [Kamble, 2008] propose ER-based models. Particularly, the proposal in [Malinowski et Zimanyi, 2006] appears as an interesting framework for the modeling of complex DW applications. This paper pays particular attention to the conceptual and logical design of complex hierarchies (e.g. non-strict, generalised, and multiple alternative). Furthermore, *this is the only model in both the DW and SDW fields that explicitly distinguishes the different hierarchies that compose the same dimension; hierarchy names are attached to the aggregating relationship roles when necessary*.

On the other hand, some studies aim at defining UML extensions [Lujan-Mora et al., 2006; Pinet et Schneider, 2009; Prat et al., 2010]. Reference [Abelló et al., 2006] proposes a UML extension called YAM^2, which allows for modeling a richer set of MD structures and for *representing multiple related hypercubes*. Another interesting feature is that it makes use of UML packages to organise the modeling elements into different abstraction levels, thereby simplifying the model. This model also represents *dimension-related aggregation rules* of measures (*Summarisation*). As recognised by the authors themselves, all of the semantic relationships make the YAM^2 too complex or even cumbersome. Reference [Lujan-Mora et al., 2006] proposes a UML profile that defines a set of stereotypes, tagged values and OCL constraints to represent the major MD properties at the conceptual level. This profile is implemented in the Rational Rose tool. Reference [Pinet et Schneider, 2009] proposes a UML-based model that unifies the representations of facts and of dimension members, thereby allowing for more flexibility in the design of DW applications. This model is enriched by a set of OCL constraints, namely structural, to guarantee its consistency.
Finally, [Prat et al., 2010] proposes another UML model and expresses different types of aggregation rules using the Production Rule Representation (PPR) language[25].

6.6.2Spatial models

Research in spatial multidimensional modeling is relatively recent. Since the pioneering work of [Han et al., 1998], some conceptual spatial MD models have been proposed in the literature; these models aim at extending the traditional multidimensional model with spatial concepts.
The majority of these studies adopt the computer science point of view regarding spatiality; they suggest that labelling specific components of SDWs (e.g. measure, dimension, and fact) as 'spatial' requires them to have a geometric representation. Only [Salehi et al., 2010] adopts the 'geomatics' point of view of spatiality, which considers spatial data as any data that is used to localise phenomena on the Earth regardless of its representation (geometry or text).
Besides being non-standards-based, some models [Jensen et al., 2004; Bimonte et al., 2005; Damiani et Spaccapietra, 2006; Salehi et al., 2010; Viswanathan et Schneider, 2010] do not provide designers with graphical notations. Consequently, using these models to conceptually design SOLAP applications can be a very difficult task. Except for [Salehi et al., 2010], these models have been proposed to address specific complex issues of SOLAP applications: handling imprecision in partial containment hierarchies [Jensen et al., 2004], dependency between spatial and numeric aggregations [Bimonte et al., 2005], representing spatial measures at different cartographic scales [Damiani et Spaccapietra, 2006], and handling field data [Ahmed et Miquel, 2005].

[25] The Production Rule Representation (PRR) is an OMG standard for modelling production rules on UML models [OMG, 2009].

Other studies propose graphical conceptual models that extend existing standard languages (ER (Entity/Relationship) and UML). These models are described in the rest of this sub-section.
Reference [Malinowski et Zimányi, 2008] extends the MultiDimER model [Malinowski et Zimanyi, 2006] with a set of spatial concepts, such as spatial level, spatial measure, and spatial hierarchy. The authors make use of MADS[26] pictograms to represent spatial features such as geometries of levels and spatial topological relationships between spatial levels (e.g. inside).
To the best of our knowledge, only [Malinowski et Zimányi, 2008; Pinet et al., 2010; Pinet et Schneider, 2010] propose UML extensions to conceptually design SDWs. Reference [Glorio et Trujillo, 2008] extends the UML profile that is presented in [Lujan-Mora et al., 2006]. This paper [Glorio et Trujillo, 2008] introduces two new stereotypes, spatial measure (a measure with a geometrical data type) and spatial level (an aggregation level having at least one geometrical attribute). In a similar way, [Pinet et Schneider, 2010] extends the Unified Object Constraint Model [Pinet et Schneider, 2009] by introducing two spatial concepts: *spatial measure* (a measure with a geometrical data type) and *spatial identified class* (a class having a geometrical attribute representing its shape). Reference [Pinet et al., 2010] formalises the SOLAP concepts proposed by [Rivest et al., 2005] by defining a UML profile that represents (spatial) hypercubes and (spatial) dimensions by UML packages, (spatial) levels by classes and measures by attributes. This profile also provides users with interesting graphical notations for some spatial and temporal MD elements.
Finally, only [Glorio et Trujillo, 2008; Malinowski et Zimányi, 2008; Pinet et al., 2010] implement their models as plug-ins in the Eclipse [27] IDE and the Objecteering[28] UML-based tool, respectively.

6.6.3Discussion

In this section, we discuss the limitations of existing conceptual models for both OLAP and SOLAP applications according to the modeling requirements presented in Section 6.5. Table 1 shows the evaluation of graphical spatio-multidimensional models.

6.6.3.1 Classical modeling requirements

The existing models meet the main classical modeling requirements (e.g. measure set and descriptive attributes). However, only [Malinowski et Zimányi, 2008] represents hierarchies explicitly. In fact, most of these models support multiple aggregation paths per dimension; however, paths forming the different hierarchies of a dimension cannot be distinguished because the hierarchies are not graphically represented. This fact makes it very difficult to express some ICs (*Intra and Inter-hierarchy ICs* [Ghozzi et al., 2003a; Boulil et al., 2010a; Pinet et al., 2010]) and *hierarchy-related aggregation rules*.
Moreover, regarding MD element typing requirements, the spatially extended MultiDimER model [Malinowski et Zimányi, 2008] also appears as the model that better addresses this issue. This model meets measure, dimension, hierarchy and aggregation level typing requirements; however, graphical notations are not provided for some types (e.g. the spatial hierarchy and the spatial dimension).

Let us now examine the existing models according to the new requirements.

[26] MADS is a well-known model for the conceptual design of spatio-temporal databases [Parent et al., 1999]
[27] http://www.eclipse.org
[28] http://www.objecteering.com

6.6.3.2 Modeling of indicators, aggregation rules and aggregate functions

All of the existing models do not make the distinction between the measure (attribute subject to aggregations) and the indicator (a combination of one measure and a set of aggregation rules; see Section 6.5.2.1). Consequently, explicit base indicator and derived indicator requirements are not met by most of them. Reference [Abelló et al., 2006] proposes a concept that is in some respects similar to the indicator, KindOfMeasure. This concept specifies the aggregation mode of a set of measures by defining a set of dimension-related aggregation rules. The problem of this approach, in addition to the derived indicators not being supported, is that one cannot define multiple indicators per measure.

As explained in Section 6.5.2.1, the indicators may be defined by different types of aggregation rules (dimension-related, hierarchy-related and sub-hierarchy-related). Apart from [Abelló et al., 2006; Kamble, 2008], which express textually dimension-related aggregation rules, all of the existing proposals do not appropriately represent the aggregation rules at the conceptual abstraction level. References [Golfarelli et al., 1998; Lujan-Mora et al., 2006; Prat et al., 2010] consider general aggregation constraints rather than aggregation rules. In other words, they express what aggregations are wrong rather than how users expect to aggregate measures.

Moreover, expressing, understanding and implementing these rules can be very studious tasks for users and designers because of the use of inappropriate formalisms, such as non-standard formalisms [Golfarelli et al., 1998; Kamble, 2008], UML notes [Lujan-Mora et al., 2006], and a new standard language [Prat et al., 2010][29].

Regarding aggregate functions, all of these models represent aggregate functions by simple labels and ignore their other properties (applicability and distributivity types; see Section 6.5.2.2). We suggest a more appropriate modeling of aggregate functions by extending the UML Operation metaclass, thereby allowing for more sophisticated and automated treatments.

We must highlight here that spatial models provide worse support for these aggregation properties than conventional ones. Except for [Bimonte et al., 2005], which uses an ambiguous ad hoc formalism to define aggregate functions that link cells of different granularities (similar to [Kamble, 2008]), none of the existing proposals in SDWs provide support for conceptual modeling of these issues. At the logical abstraction level, [Salehi, 2009] expresses different aggregation constraints using an ad hoc language (hybrid natural with pictograms), which leads to the same modeling and implementing difficulties.

6.6.3.3 Support for multi-granular measures

All of the proposed models do not provide the necessary support for multi-granular measures. First, they do not explicitly represent multi-granular measures. In [Jensen et al., 2004], facts can be related to non-bottom aggregation levels, but no graphical notation is suggested. Second, these studies do not consider disaggregation functions and measure ETL semantics concepts that are necessary requirements to correctly calculate the indicators associated with these measures (see Section 6.4.3.)

6.6.3.4 Data quality

As recognised in [Abelló et al., 2006; Boulil et al., 2010a; Pinet et Schneider, 2010], UML-based models are more suited to address the data quality issues in DWs and SDWs. In fact, these models can be enriched by OCL constraints that can be easily implemented because of

[29] To the best of our knowledge, the Production Rule Representation (PRR) language used by the authors is not integrated in any modelling tool yet.

the integration of this standard language (OCL) in most of the UML-based tools. Addressing this issue with other formalisms, such as ER or ad hoc ones, can be a very difficult and time-consuming task. Consequently, the UML SDW models, such as [Glorio et Trujillo, 2008; Pinet et al., 2010; Pinet et Schneider, 2010], handle this issue better. Moreover because these proposals do not explicitly represent hierarchies, we cannot easily express some constraints such as *Intra/Inter-hierarchy ICs.*

Modeling requirements	**[Malinowski et Zimányi, 2008]**	**[Lujan-Mora et al., 2006]**	**[Pinet et Schneider, 2010]**	**[Pinet et al., 2010]**
1. Classical requirements				
Measures				
Measure sets	YES	YES	YES	YES
Explicit multi-granular measures	NO	NO	NO	NO
Dimensions				
Descriptive attributes	YES	YES	YES	YES
Identifying attributes	YES	YES	YES	YES
Multiple hierarchies	YES	YES	YES	YES
Explicit hierarchies	YES	NO	NO	NO
Aggregate functions	NO	Partially	NO	NO
Typing of MD elements				
Measure using type	Partially	Partially	Partially	Partially
Measure additivity type	YES	Partially	NO	NO
Typing of dimensions	YES	NO	Partially	YES
Typing of hierarchies	YES	NO	NO	NO
Typing of aggregation levels	YES	Partially	Partially	YES
Complex data structures				
Complex hierarchies	YES	YES	YES	YES
Fact-dimension relationships	Partially	YES	YES	NO
2. Modeling of indicators and aggregation rules				
Explicit indicators	NO	NO	NO	NO
Explicit derived indicators	NO	NO	NO	NO
Multiple ordered aggregation rules per indicator	NO	NO	NO	NO
Aggregation rules	NO	Partially	NO	NO
3. Typing of aggregate functions				
The distributivity type	NO	NO	NO	NO
The applicability type	NO	NO	NO	NO
4. Support for multi-granular measures		NO		
Measure ETL semantics	NO	NO	NO	NO
Disaggregation functions	NO	NO	NO	NO
5. Data quality	Partially	Partially	Partially	Partially

Tableau 6.1. The evaluation of graphical conceptual spatial MD models.

Finally, all of the existing models whether spatial or not, standards-based or not, focus on the modeling of static MD structures. In addition to some limitations regarding static aspects (e.g. multi-granular measures, explicit hierarchies, and data quality), these studies do not provide enough support for the conceptual representation of dynamic aspects related to the aggregation of measures (e.g. indicators, aggregation rules and aggregate functions).

Exploiting well-known UML constructs, we propose in this chapter a new UML profile that allows the modeling of both warehouse and aggregation aspects of spatial data cubes in a very simple and declarative manner. In addition to traditional modeling requirements (e.g. explicit hierarchies, multi-granular measures, aggregate functions, and typing of MD elements), this profile meets all of the newly identified requirements: (a) modeling of indicators and aggregation rules, (b) typing of aggregate functions (using and applicability types), (c) support for multi-granular measures (measure ETL semantics and disaggregation functions) and (d) support for data quality issues.

6.7 The UML Profile for spatial data cubes

In this section, we detail our profile for spatial data cube conceptual modeling (Section 6.7.2). This profile satisfies all of the requirements that are defined in Section 6.5. The UML profiling principles are introduced in Section 6.7.1.

6.7.1UML Profiles

UML can be extended in two ways: (a) with profiles to adapt it to specific platforms (e.g. J2EE/EJB) or domains (e.g. finance and telecommunications) or (b) by creating new metamodels (composed of metaclasses and metarelationships) that modify its metamodel [OMG, 2011c].

Using profiles, existing UML metaclasses (e.g. property, class, and association) can be customised with three mechanisms: *stereotypes, tagged values and constraints*.

A stereotype is an extension of a UML metaclass that defines additional tagged values, constraints and optionally a new graphical notation (icon). At the model level, it can be visualised in two ways: using its icon or a string representing its name between a pair of guillemets (<<*stereotype name*>>). Tagged values represent properties of stereotypes. At the model level, each tagged value is attached to a model element to which a stereotype is applied. A tagged value is represented by its name and an associated value from its value domain and is placed between a pair of braces (*{tagged value name=value}*). The *constraints* refine the definitions of stereotypes and tagged values and prevent their arbitrary use by designers when defining models. They can be used, for example, to restrict the attribute value domains or the stereotype associations. These constraints are often formalised using OCL.

6.7.2The SDW profile description

In our approach, both static and dynamic aspects of MD models are declaratively modelled by means of UML class diagram elements (e.g. class, package, property, and operation). In addition to well-known requirements (such as the modeling of measures and dimensional attributes as UML properties, facts and aggregation levels as UML classes, and multiple hypercubes and dimensions as packages), we explicitly represent aggregate functions and aggregation rules by operations, indicators by classes, and hierarchies by packages. Similar to [Abelló et al., 2006], we make use of packages to organise the MD constructs into different levels of detail. This organisation facilitates the understanding of complex models.

Our proposal is formally defined as a UML profile that is implemented in the MagicDraw tool[30]. To the best of our knowledge, MagicDraw is the only tool that enables the checking of OCL constraints of profiles. Our profile defines a set of stereotypes and tagged values for the conceptual modeling of spatial data cubes. It is organised into three main packages (see Figure 6.3). The SDW Core Model package allows for representing the main static concepts of SDWs. The Aggregation Model package allows for representing the aggregation knowledge of SDWs (i.e. the indicators and their aggregation rules). In the constraints

[30] http://www.magicdraw.com

package, a set of OCL constraints is defined to prevent the arbitrary use of the stereotypes and tagged values of our profile. We present in this section some representative profile-inherent structural constraints. User-defined constraints are presented in Section 6.8.

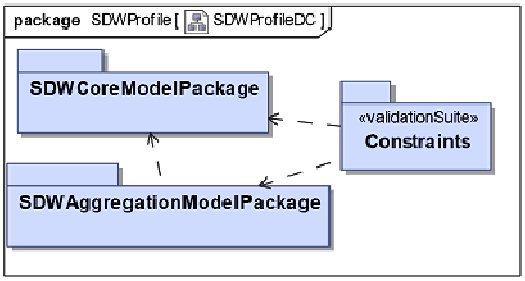


Figure 6.3. The SDW Profile – Main packages.

6.7.2.1 The SDW Core Model Package

We represent the metamodel of the SDW core model package in Figure 6.4. Each SDW model (<<SDWModel>>) is viewed as a non-empty finite set of hypercubes (<<Hypercube>>) for which at least one is spatial.

The hypercubes represent subjects of analysis. Formally, each hypercube consists of a (possibly empty) finite set of measures (<<Measure>>) gathered into a fact class (<<Fact>>) and a non-empty finite set of dimensions (<<Dimension>>). It is considered spatial (*isSpatial=true*) if it contains at least one spatial dimension or at least one spatial measure.

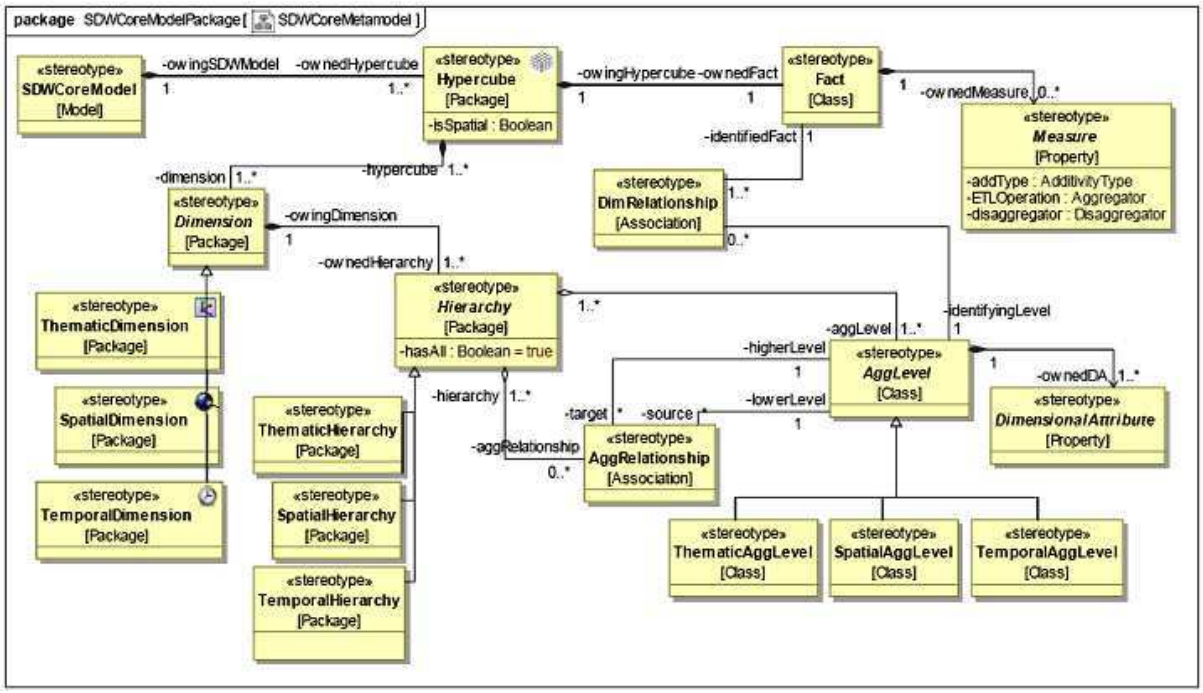


Figure 6.4. The SDW Profile – The SDW core model package.

In Figure 6.5, the SDW model of a retail company presented in Section 6.4 is defined by two hypercubes, "SupplyingHypercube" and "SalesHypercube". The sales hypercube is spatial because it contains two spatial dimensions, "Stores" and "Customers" (see Figure 6.6). The supplying hypercube is not spatial. These two hypercubes are represented by two UML packages and share two dimensions ("Time" and "Products").

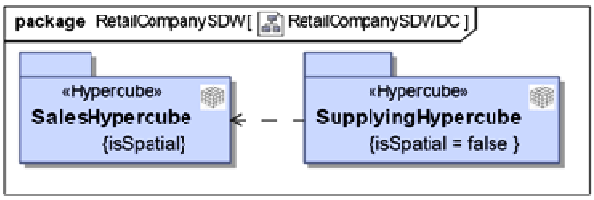


Figure 6.5. The retail SDW – Hypercubes.

Measures are represented by the *Measure* stereotype, which is defined as an extension of the UML property metaclass (see Figure 6.4). Dimensions are represented by the *Dimension* stereotype, which extends the UML package metaclass. Each dimension is defined by a non-empty finite set of hierarchies (see Figure 6.4).
Hierarchies allow for analysing facts at different levels of detail. Formally, each hierarchy (<<Hierarchy>>) consists of one or more related aggregation levels that form a Directed Acyclic Graph (DAG) with exactly one bottom level and several top levels. All of the hierarchies of a dimension have the same bottom level, which should be related to the hypercube's *Fact*. Hierarchies can have a level "All" with a single instance (*hasAll* = true). *In contrast to existing models, we represent hierarchies explicitly as specific packages using the Hierarchy stereotype* (see Figure 6.4).

Figure 6.6 shows the class diagram of the sales hypercube. This hypercube is described by five measures (e.g. "price" and "quantity_sold") that are grouped into the "Sales" fact class. These measures are analysed following five dimensions, "Products", "Stores", "Orders", "Customers" and "Time". Each dimension is described by at least one hierarchy. For example, the "Products" dimension contains one hierarchy ("ProductCategory"), and the "Customers" dimension contains two hierarchies ("CustomerType" and "CustomerLocation"). We will provide details about the *NumericalMeasure* stereotype and its tagged values in the next sub-sections.

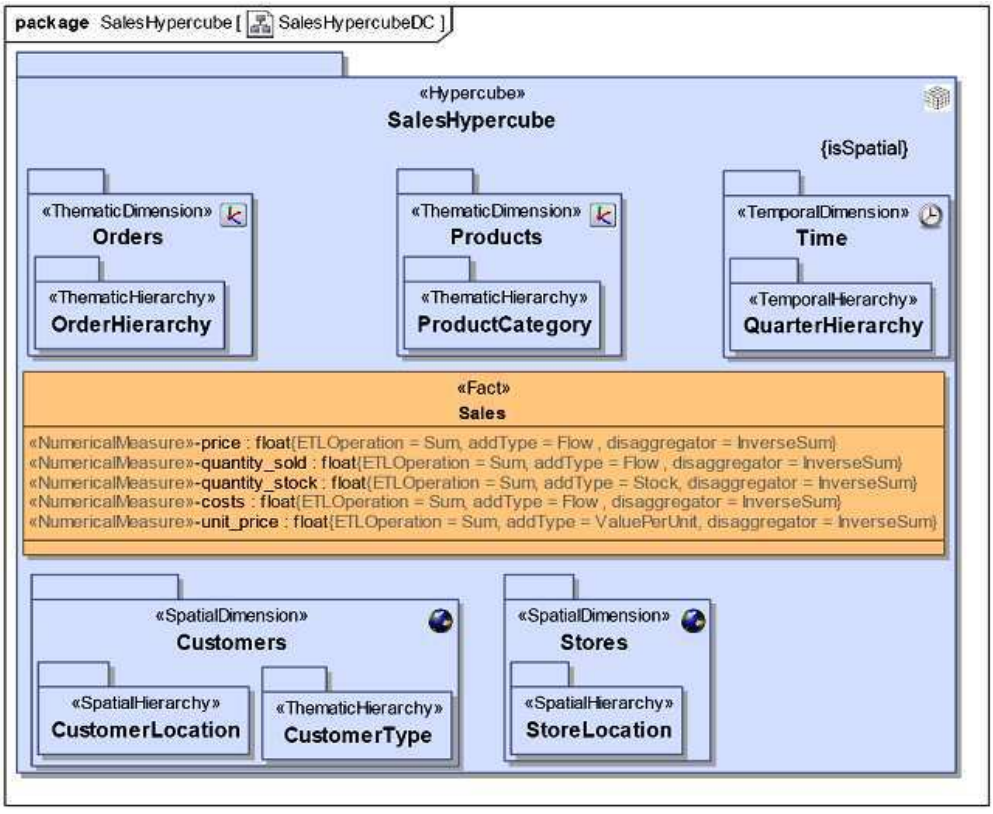


Figure 6.6. The sales hypercube – Dimensions and measures.

Aggregation levels (<<*AggLevel*>>) define interesting viewing levels of measures according to multidimensional analysis requirements. Each aggregation level may contain several dimensional attributes (<<DimensionalAttribute>>), which can be of different types, as we will see below. Aggregation levels as fact classes are represented by extensions of the UML class metaclass (see Figure 6.4).
To form hierarchies, aggregation levels must be linked to each other using aggregating relationships (<<AggRelationship>>) that define full or partial containment relationships between their members.
For example, the "CustomerLocation" hierarchy consists of four related aggregation levels ("Customer", "City", "State" and "Country"), and the "CustomerType" hierarchy consists of two related aggregation levels ("Customer" and "Type") (see Figure 6.7). Each one of the aggregation levels is described by a number of attributes. For example, "City" contains four attributes (e.g. "city_id", and "city_name").

To define the grain of measures, the fact class should be related to aggregation levels using dimensioning relationships (<<DimRelationship>>) (see Figure 6.4). In each hypercube, the *Fact* must be linked to at least a bottom aggregation level of each dimension. As shown in Figure 6.7, the "Sales" *Fact* is related to six aggregation levels ("Order", "Customer", "Product", "Store", "Month" and "Quarter") by the *DimRelationship* associations to define the grains of sales and their analysis context.

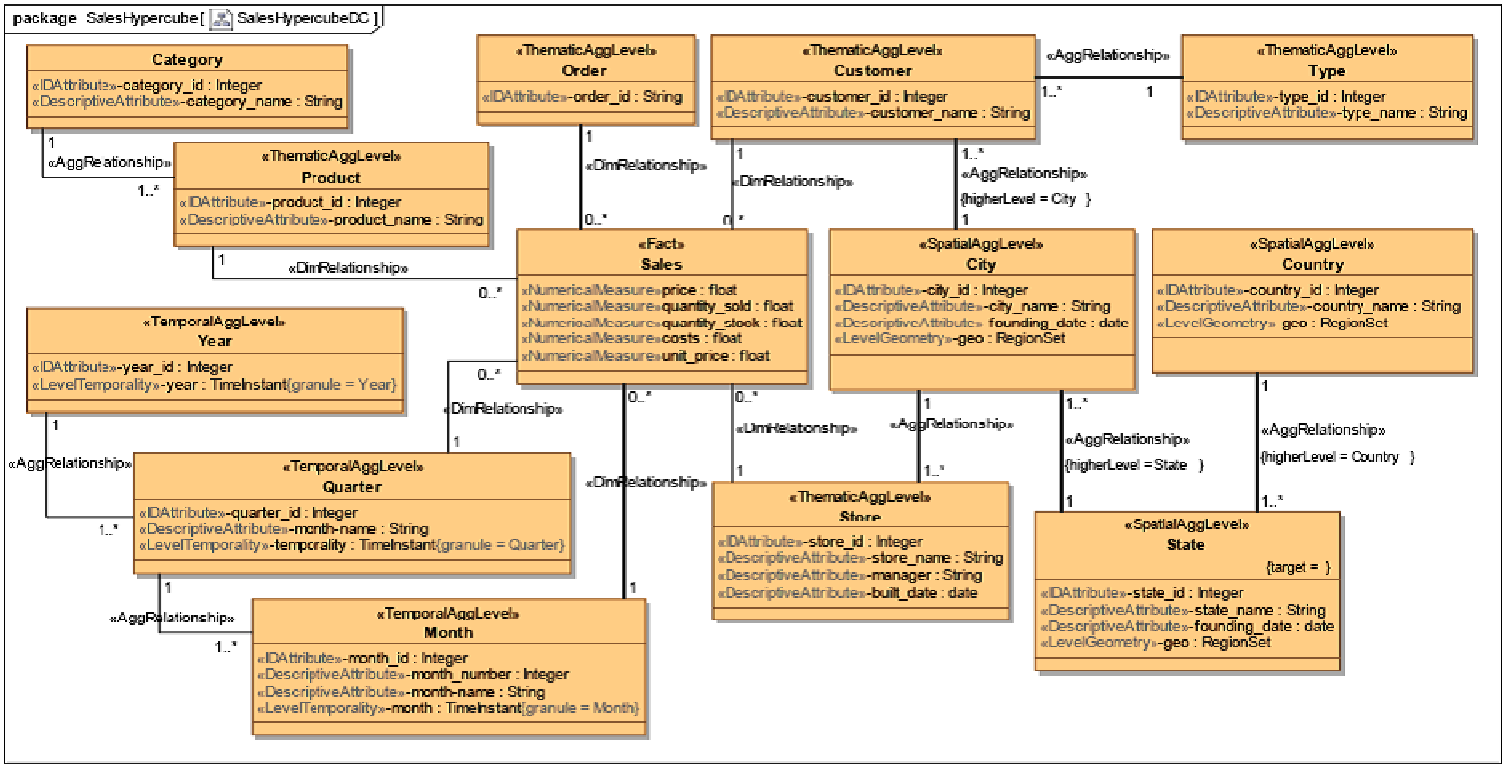


Figure 6.7. The sales hypercube – The hierarchies and the grain of measures

To avoid an arbitrary use of our profile, we formalise each stereotype and tagged value with a set of OCL constraints. At the model level (i.e. when the profile is used by designers), these constraints are checked by MagicDraw at the conceptual abstraction level; this prevents incorrect modeling. For example, to guarantee that each spatial hypercube contains at least one spatial dimension or at least one spatial measure, we define the following OCL constraint in the context of the *Hypercube* stereotype.

```
context Hypercube inv AtLeastOneSpatialDimensionOrOneSpatialMeasure:
isSpatial = true implies (
ownedMember->select(m | m.oclIsTypeOf(SpatialDimension))->size()>0 or
ownedMember->exists(m | m.oclIsTypeOf(Fact) and m.ownedAttribute->select(
a | a.oclIsTypeOf(SpatialMeasure))->size()>0 ) ) )
```

According to this constraint, if we label the conventional hypercube "SupplyingHypercube" as spatial (*isSpatial=true*), MagicDraw returns the error message: "*A spatial hypercube must contain at least one spatial dimension or at least one spatial measure*". This is because this hypercube does not contain any spatial dimensions or any spatial measures (see Figure 6.8).

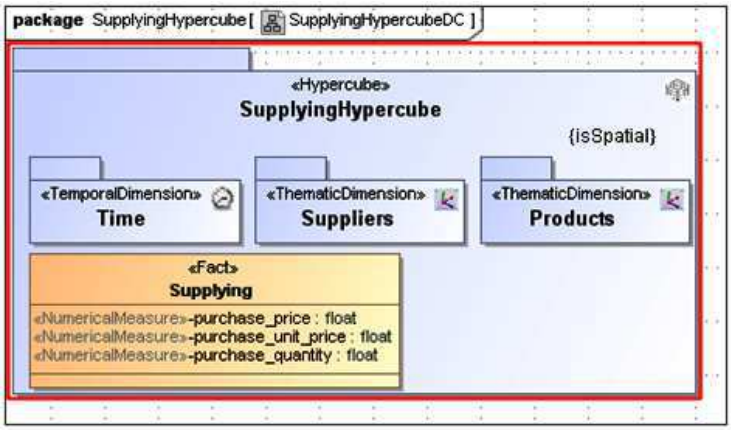


Figure 6.8. Example of incorrect MD model.

6.7.2.1.1 Typing of MD elements

As stated in Section 6.5.1.4, the precise typing of multidimensional elements is a very important requirement. In our profile, we propose a precise typing of measures, dimensions, hierarchies, aggregation levels and aggregate functions. We define for each type a specific graphical notation to increase the readability.

6.7.2.1.1.1 Typing of measures

As previously described in Section 6.5.1.4.1, the representations of the using and the additivity types of measures are very important requirements because these two characteristics influence their aggregations. To meet the measure using type requirement, we specialise the *Measure* stereotype into five sub-types: *NumericalMeasure*, *SpatialMeasure*, *BooleanMeasure*, *TextualMeasure* and *TemporalMeasure* (Figure 6.9).

Following their additivity type, measures are classified into *flow*, *stock* and *value per unit* measures. These three types are defined by the enumeration *AdditivityType*. The additivity type of each measure is captured by its *addType* tagged value definition.

In the sales hypercube of Figure 6.6, all of the measures are numerical. Regarding their additivity, the measures "price", "quantity_sold", and "costs" are of a flow type (the *addType* tagged value is set to *Flow*); "quantity_stock" is of a stock type and "unit_price" is of a value per unit type.

6.7.2.1.1.2 Typing of dimensional attributes, aggregation levels, hierarchies and dimensions

Dimensional attributes are modelled as attributes owned by aggregation levels (Figure 6.9). We distinguish four sub-types: (a) identifying attributes (<<IDAttribute>>), used for grouping purposes when performing Roll-up operations; (b) descriptive attributes

(<<DescriptiveAttribute>>), used in predicate roles of Slice and Dice operations; (c) <<LevelGeometry>> attributes, used to represent geometrical shapes of spatial dimension members; and (d) <<LevelTemporality>> attributes, used to model temporal extents of temporal members. The *LevelGeometry* attributes must be of spatial data types (e.g. Point, and Polygon). The *LevelTemporality* attributes are characterised by their temporal data types (e.g. TimeInstant, and TimeInterval) and their time granularities (e.g. Second, and Minute) captured by the *granule* tagged value definition. The different time granularities are defined by the *TimeGranularity* enumeration.

To define data types of spatial and temporal MD attributes, we have created new data types by considering the MADS [Parent et al., 1999] spatial (e.g. Point, Line, Region, and RegionSet) and temporal (e.g. TimeInstant, TimeInterval, and TimeInstantSet) data types. These new data types are defined as extensions of the UML DataType metaclass. For thematic attributes, we consider the OCL and UML predefined data types (e.g. Real, String, and Boolean).

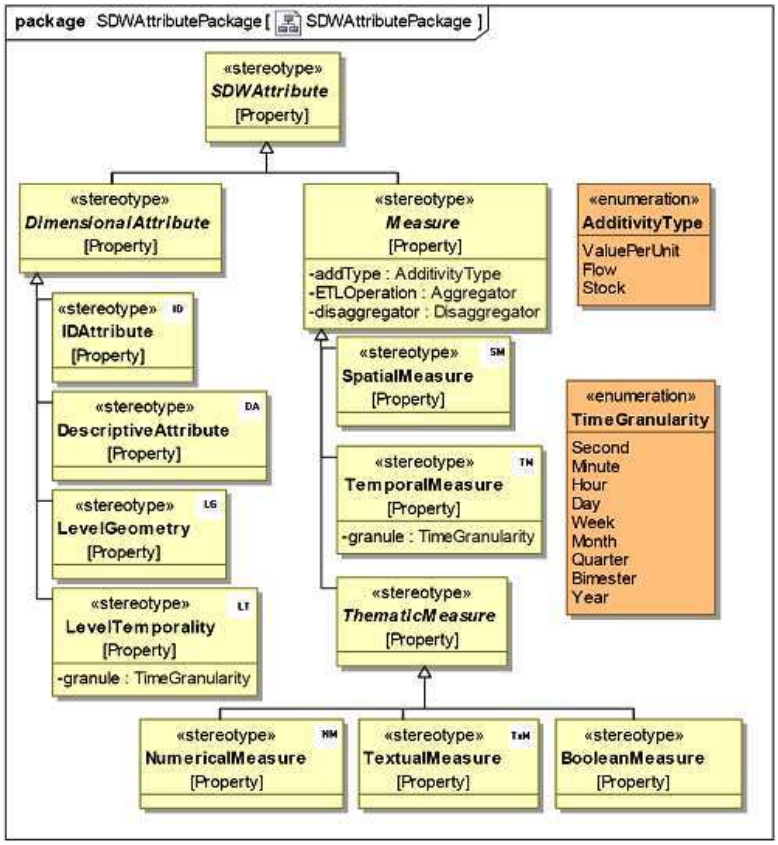


Figure 6.9. The SDW Profile – The SDW attribute package.

As presented in Figure 6.4, we specialise dimensions, hierarchies and aggregation levels into three categories (thematic, spatial, and temporal). The spatial dimension (<<SpatialDimension>>) represents the locations of facts as an analysis axis and includes at least one spatial hierarchy. The temporal dimension (<<TemporalDimension>>) records time instants or time periods when facts occurred; it only contains temporal hierarchies. To be contrasted with the spatial and temporal ones, the thematic dimension (<<ThematicDimension>>) stores only members that are neither visualised on maps nor define time frames of facts.

The following OCL constraint guarantees that each spatial dimension contains at least one spatial hierarchy.

```
context SpatialDimension inv AtLeastOneSpatialHierarchy:
self.ownedMember->exists (m | m.oclIsTypeOf(SpatialHierarchy))
```

A spatial hierarchy (<<SpatialHierarchy>>) includes at least one spatial aggregation level, while a temporal hierarchy (<<TemporalHierarchy>>) contains only temporal aggregation levels. A thematic hierarchy (<<ThematicHierarchy>>) contains only thematic aggregation levels.
A thematic aggregation level <<ThematicAggLevel>> contains only identifying and descriptive attributes. In addition to identifying and descriptive attributes, a spatial aggregation level (<<SpatialAggLevel>>) is characterised by its geometry (<<LevelGeometry>>), and a temporal aggregation level (<<TemporalAggLevel>>) by its temporality (<<LevelTemporality>>).
The following OCL constraint guarantees that each spatial aggregation level contains exactly one *LevelGeometry* attribute.

```
context SpatialAggLevel inv ExactlyOneLevelGeometry :
self.ownedAttribute->exists (a | a.oclIsTypeOf(LevelGeometry))
```

Thus, reconsidering the sales hypercube (see Figure 6.6), "Orders" and "Products" are thematic dimensions; "Customers" and "Stores" are spatial dimensions; and "Time" is a temporal dimension. Each one of these dimensions is composed of specific types of hierarchies. For example, the spatial dimension "Customers" is composed of one thematic hierarchy ("CustomerType") and one spatial hierarchy ("CustomerLocation"). The latter is defined by four related aggregation levels (one thematic level and three spatial levels), as shown in Figure 6.7.
The aggregation levels are described by different types of dimensional attributes according to their nature (thematic, spatial or temporal). For example, the spatial aggregation level "City" contains one identifying attribute ("city_id"), two descriptive attributes ("city_name" and "founding_date"), and one *LevelGeometry* attribute ("geo") having a RegionSet data type (see Figure 6.7).

6.7.2.1.2 Support for multi-granular measures

Unlike existing models, in our model, fact classes can be linked to non-leaf aggregation levels using dimensioning relationships (<<DimRelationship>>) to represent multi-granular measures (see Figure 6.4). As stated in Section 6.5.2.3, in this context, it is important to know the ETL semantics of measures (e.g. totals, minima, and averages) to safely perform aggregations. To capture this knowledge at the conceptual abstraction level, we add the *ETLOperation* tagged value definition to the *Measure* stereotype (see Figure 6.4).
To seek simplicity, we state that values of a given measure must have the same semantics regardless of their level of details. To guarantee this assumption, we define the following OCL constraint:

```
Context Measure inv oneETLSemanticForAllMeasureValues:
Measure.allInstances()->notExists(v| v <> self and
v.ETLOperation.name <> self.ETLOperation.name)
```

Another important requirement in this context is the specification of the disaggregation functions that users want to apply to these measures (see Section 6.5.2.3). We represent such information using the *disaggregator* tagged value added to the *Measure* stereotype (Figure

6.4). The *Aggregator* and *Disaggregator* stereotypes are explained in Section 6.7.2.2.1 and Section 6.7.2.2.4, respectively.
In Figure 6.7, the "Sales" *Fact* is related to two levels of the "Time" dimension ("Month" and "Quarter"), to represent measure values that are acquired at two different granularities, monthly sales and quarterly sales. All of the measures have the same semantics (i.e. totals) (the *ETLOperation* tagged value is set to Sum) regardless of their grains of acquisition (see Figure 6.6). The function used to disaggregate the quarterly sales from the "Quarter" to the "Month" level is an inverse function of the Sum (*disaggregator = InverseSum*). This function is defined in Section 6.7.2.2.4.

6.7.2.2 The Aggregation Model Package

The aggregation model package represents the aggregation knowledge of spatial data cubes at the conceptual abstraction level. It extends UML with a set of new elements that reference some elements of the core SDW model (Section 6.7.2.1). Its metamodel is depicted in Figure 6.10.

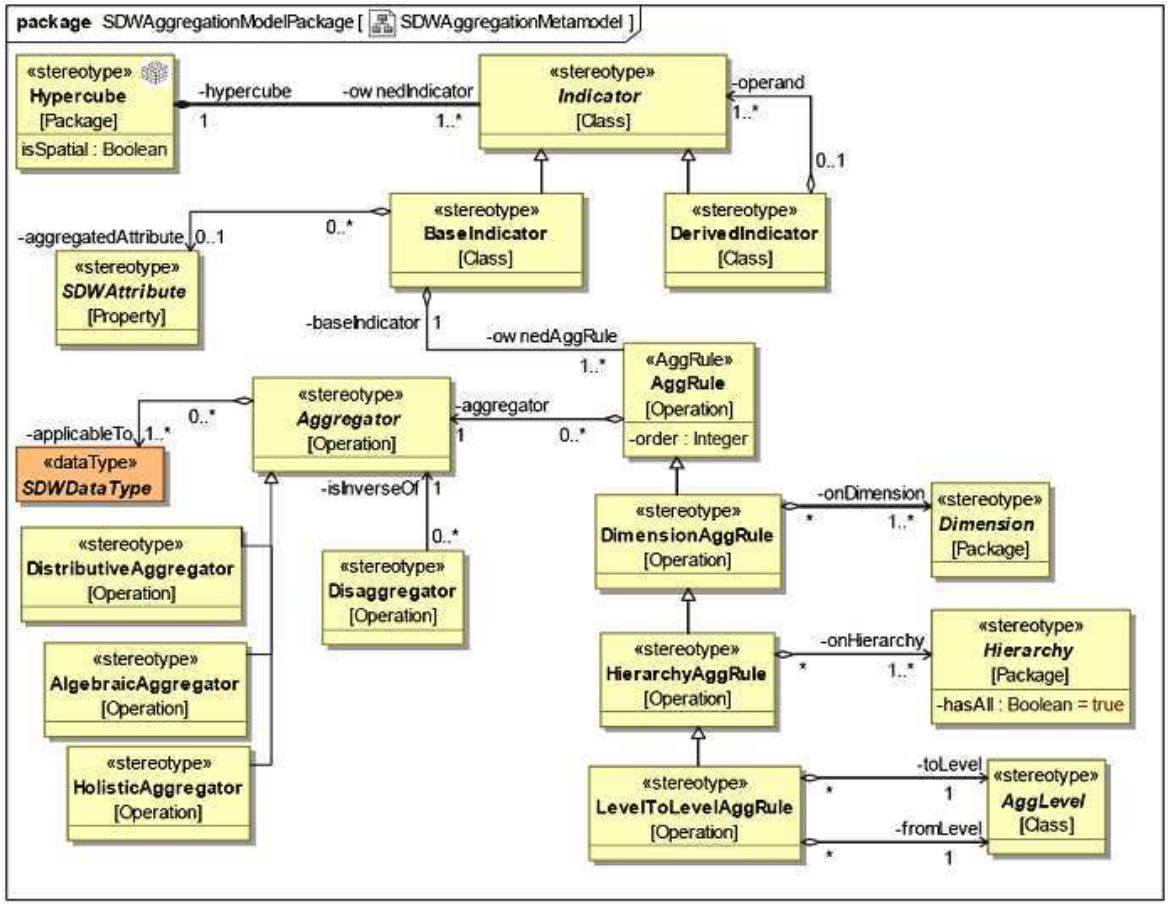


Figure 6.10. The SDW Profile –The SDW Aggregation Model package.

In our approach, each hypercube (<<Hypercube>>) can be enriched with a finite set of indicators (<<Indicator>>), which can be derived (<<DerivedIndicator>>) or not (<<BaseIndicator>>).
Each base indicator involves either an SDW attribute (a measure or a dimensional attribute) or nothing. The latter (nothing) is particularly important when analysing factless facts[31]. In

[31] Factless Facts are described by empty sets of measures and are typically used for event tracking [Golfarelli et al., 1998].

this context, we can only count fact instances or aggregate attribute values of dimension members involved in facts.
The following OCL constraint verifies that the measure referenced by the base indicator exists in the SDW model (when it involves a measure).

```
context BaseIndicator inv ValidInvolvedMeasure:
(aggregatedAttribute->notEmpty() and
aggregatedAttribute.OclIsTypeOf(Measure)) implies
hypercube.ownedFact.ownedMeasure-> includes(aggregatedAttribute)
```

For example, as shown in Figure 6.12, the indicator "Total_Quantity_Sold" involves the measure "quantity_sold"; the indicator "Total_Nb_Orders" involves the dimensional attribute "order_id" and computes the total number of orders; and "Total_Nb_Sales" calculates the total number of sales and does not involve any element of the core model. In the remainder of this chapter, we refer to the element that is subject to aggregations as the **Aggregated Element**.
Moreover, each basic indicator is defined by a finite ordered set of aggregation rules, and each aggregation rule is associated with an aggregator (an aggregate function), to conceptually define how the indicator should be calculated when navigating along dimension hierarchies.

6.7.2.2.1 Modeling of aggregate functions

In our proposal, aggregate functions are modelled as UML operations using the <<Aggregator>> stereotype (Figure 6.10). Following the *Aggregator* definition, each aggregate function is characterised by a finite set of SDW data types to which it can be applied (in line with the applicability type requirement, Section 6.5.2.2), which is captured by the *applicableTo* tagged value that references the SDW data types defined in the SDW core model package.
For example, the Count aggregator can be applied to all SDW data types (*applicableTo=SDWDType*), Sum can be applied to only numerical data types, and the GeometricUnion can be applied to only spatial data types (Figure 6.11).
In line with the distributivity type requirement (see Section 6.5.2.2), the *Aggregator* stereotype is specialised into three sub-types: (a) distributive (<<DistributiveAggregator>>), (b) algebraic (<<AlgebraicAggregator>>), and (c) holistic (<<HolisticAggregator>>) (see Figure 6.10).
For example, GeometricIntersection is distributive (the applied stereotype is *DistributiveAggregator*), Centroid is algebraic, and EquiPartition is holistic (Figure 6.11).
It is worth noting here that new aggregators (namely user-defined ones) can be easily added to the pre-defined aggregators (Figure 6.11) to enable richer analysis.

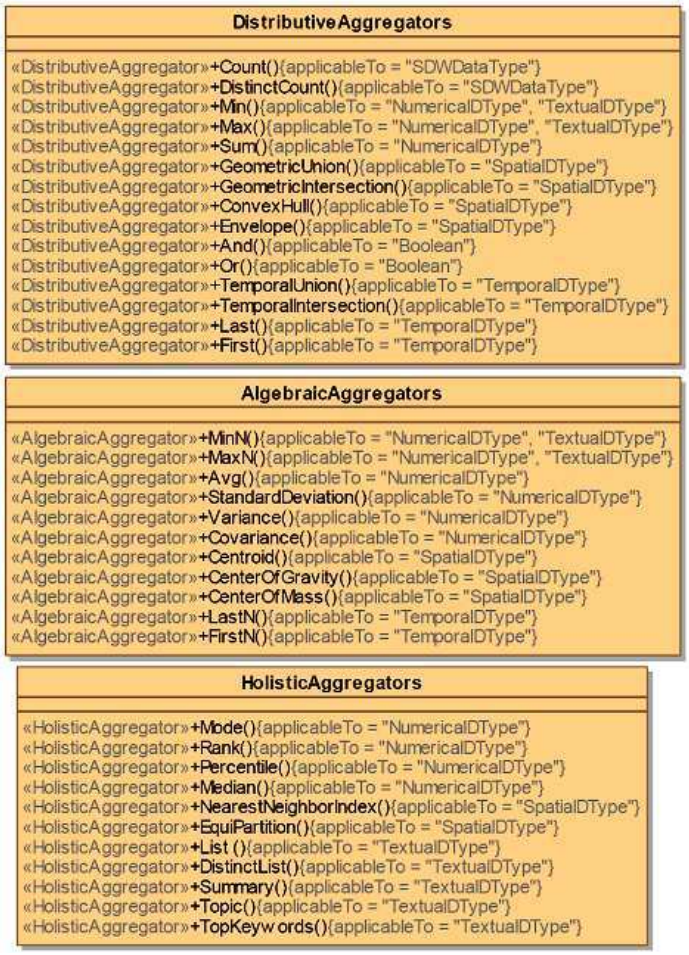


Figure 6.11. The SDW Profile – Predefined Aggregators.

6.7.2.2.2 Aggregation rules

An aggregation rule specifies the aggregate function that must be applied to aggregate values of the *Aggregated Element* along a given dimension or a given hierarchy or between two given aggregation levels of the same hierarchy. Each aggregation rule is characterised by its name and its precedence order (*order*) with respect to the other aggregation rules of the base indicator (see Figure 6.10).

In our profile, aggregation rules are modelled by extensions of UML operations; the *order* and *aggregator* tagged values are their parameters. As described in Section 6.5.2.1, aggregation rules are classified into four categories (see Figure 6.10). These categories are described in the following sub-sections.

6.7.2.2.2.1 Aggregation rules stereotyped as <<AggRule>>

Each one of these rules defines the aggregate function (*aggregator tagged value*) that should be applied to the aggregated element independently of any dimensions (i.e. along all dimensions) (see Figure 6.10).

The following OCL constraint ensures that each base indicator must contain at most one aggregation rule of this type.

```
context BaseIndicator inv AtMostOneAggRule:
ownedAggRule-> select(r|r.oclIsTypeOf(AggRule)->size()<=1
```

In Figure 6.12, all of the base indicators are defined using *AggRule* aggregation rules ("R1" to "R6"). For example, "R1" states that the Sum must be applied to the "quantity_sold" to compute the "Total_Quantity_Sold" base indicator.

Figure 6.12. Examples of base indicators that contain only AggRule rules.

6.7.2.2.2.2 Aggregation rules stereotyped as <<DimensionAggRule>>

A *DimensionAggRule* defines an applicable aggregate function to the aggregated element involved by the base indicator when navigating along a non-empty set of the hypercube's dimensions (*onDimension*) (see Figure 6.10). For example, to compute the "Stock_Level" indicator (Figure 6.13), we must first sum the measure "quantity_stock" on the dimensions "Customers", "Orders", "Products" and "Stores" (rule "R7" with precedence order = 0), then apply Avg on "Time" (rule "R8" with a precedence order=1).

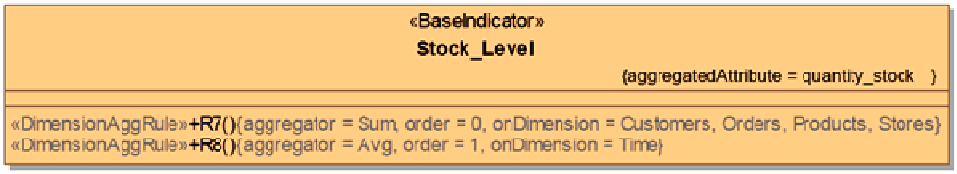


Figure 6.13. An example of a base indicator that contains DimAggRule rules.

The following OCL constraint verifies that each base indicator does not contain conflicting *DimensionAggRule* rules that define different aggregate functions on the same dimension.

```
context BaseIndicator inv noConflictingDimensionAggRules:
self.ownedAggRule->select(r|r.oclIsTypeOf(DimensionAggRule))->forAll(r1,
r2|(r1<>r2) implies
r1.onDimension->intersection(r2.onDimension)->isEmpty())
```

6.7.2.2.2.3 Aggregation rules stereotyped as <<HierarchyAggRule>>

A rule of this type defines the applicable aggregate function on a non-empty set of hierarchies (Figure 6.10). For example, the following base indicator (Figure 6.14) is defined by two aggregation rules: "R9" and "R10". "R10" is a *HierarchyAggRule*, which states that "quantity-sold" values must be averaged (Avg) when navigating on the "CustomerLocation" hierarchy to compute this indicator. Sum is specified for the other hierarchies ("R9").

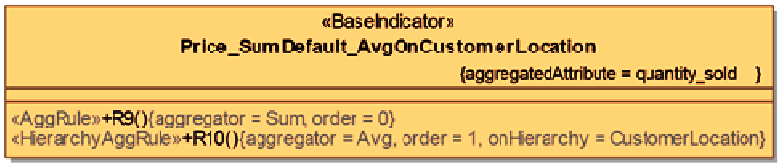


Figure 6.14. An example of a base indicator that contains a HierarchyAggRule rule.

6.7.2.2.2.4 Aggregation rules stereotyped as <<LevelToLevelAggRule>>

A rule of this type defines the applicable aggregate function when navigating from an aggregation level to another aggregation level that is coarser but of the same hierarchy (Figure 6.10). For example (Figure 6.15), to compute the maxima of city turnovers ("MaxTurnoverPerCity"), we must sum the "price" values from the "Customer" level to the "City" level to calculate the turnovers per city ("R12"); subsequently, we apply Max ("R13"). On the other dimension hierarchies, Sum is stated ("R11").

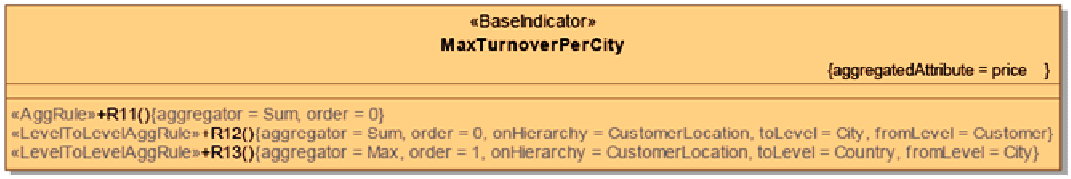


Figure 6.15. An example of a base indicator that contains LevelToLevelAggRule rules.

6.7.2.2.3 Derived indicators

As stated in Section 6.5.2.1, indicators can also be calculated from other indicators. In our profile, a derived indicator (<<DerivedIndicator>> is defined by its name and a finite algebraic expression over the other indicators (derived or not) (see Figure 6.10). This expression is specified by a <<derive>> OCL 2.0 expression on an attribute named 'formula'. The *DerivedIndicator* definition is formalised using OCL constraints specified at the metamodel abstraction level. For example, the following OCL constraint guarantees that every derived indicator has exactly one attribute named *formula* with an OCL OpaqueExpression data type and a non-null default value.

```
context DerivedIndicator inv oneAttributeNamedFormula:
ownedAttribute-> select(a| a.name = 'formula' and a.type.name =
'OpaqueExpression' and a.default <> '')->size() = 1
```

In Figure 6.16, an example of a derived indicator is depicted ("Profit"). It is defined using an OCL 2.0 Opaque Expression (the *formula* tagged value) and computes the value difference of two base indicators, "Turnover" and "Total_Costs" (see Figure 6.12), which are referenced by the *operand* tagged value.

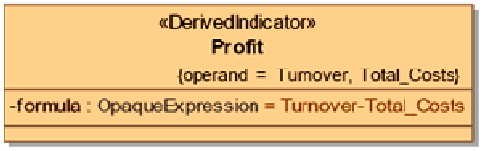


Figure 6.16. An example of a derived indicator.

6.7.2.2.46.2.2.4 Modeling of disaggregation functions

As stated in Section 6.5.2.3, the modeling of disaggregation functions is a mandatory requirement in the context of multi-granular measures. In our profile, these functions are

modelled as inverse functions of aggregate functions with the <<Disaggregator>> stereotype (see Figure 6.10). Each aggregate function can have several inverse functions.
An example of a disaggregation function is shown in Figure 6.17 (*InverseSum*). This is an inverse function of Sum for which the signature is formalised as follows, using the OCL concrete syntax:

```
context Disagreggator: InverseSum (V:NumericalDType, S:Bag(Tuple(m:
member, factor: Real )) : Bag(Tuple(m:member, v: NumericalDType))
```

This function has two inputs: (a) the aggregate value V and (b) a bag of tuples S, representing dimension members and their estimated contribution factors to V. The function returns another bag of tuples by assigning to each member m a sub value of V (v).

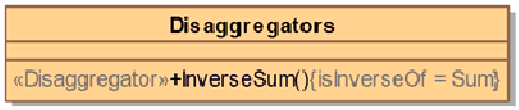


Figure 6.17. An example of a Disaggregator.

For example, using this disaggregator, we can compute all of the "Total-Quantity_Sold" values at the "Month" level by disaggregating "quantity-sold" measure values stored at the "Quarter" level.

6.8 Support for data and aggregation qualities

As stated in [Salehi, 2009; Boulil et al., 2010a], Integrity Constraints (ICs) are an excellent way to address the data and aggregation quality problems in SDWs. In [Boulil et al., 2010a], we proposed a detailed SDW IC classification that is based on the MD concepts (e.g. fact, and dimension member) that can be involved with SDW constraints. The IC classes identified can fall into four main categories:

(a) metadata ICs that verify the validity and the compatibility of metadata of different integrated data sources (e.g. all of the spatial data of a hypercube should be at the same geographic scale);
(b) data ICs that check the consistency of warehoused data (i.e. facts, dimension members and their relationships);
(c) aggregation constraints that guarantee correct aggregations of measures when navigating on dimension hierarchies; and
(d) exploration ICs that ensure consistent formulations of some MD queries (e.g. Inter-hierarchy ICs).

These four categories are further divided into several sub-categories. For example, aggregation constraints are divided into schema, completeness, semantic and user-defined sub-categories.

By explicitly representing the main warehouse and aggregation-related MD elements, our model facilitates the expression of the different classes of ICs identified in [Boulil et al., 2010a]. Specifically, by representing (spatial) data cube elements (such as hierarchies, aggregate functions, indicators, and aggregation rules), we overcome limitations of existing models regarding expressing some IC types (Inter-hierarchy ICs and aggregation constraints) [Boulil et al., 2010a]. Furthermore, being UML-based, our profile supports OCL and allows the definition of ICs following MDA principles.

In the following, we provide some examples of data ICs expressed on the sales hypercube presented in the previous section (Section 6.8.1) and propose an interesting approach to address semantic aggregation constraints (Section 6.8.2).

6.8.1Data Integrity Constraints

Data ICs allow checking the consistency of warehoused data. In [Boulil et al., 2010a], we proposed a detailed categorisation of these ICs. We also proposed to use the Spatial OCL[32] language to express them at the conceptual level and to use the Spatial OCL2SQL code generator[33] to automatically generate their implementations in the Oracle Spatial DBMS. Here, we present OCL expressions of two data ICs that are defined on the sales hypercube presented in Section 6.7.2.1; for more samples, see [Boulil et al., 2010a].

(a) An example of Inter-level IC. The following data IC expressed in Spatial OCL verifies that each city (i.e. each member of the "City" level) is geographically contained in its state (member of the "State" level). In terms of the classification proposed in [Boulil et al., 2010a], it is an Inter-level IC because it involves members of two aggregation levels.

```
context City inv EachCityIsCoveredByItsState:
geo.isInside(county.geo) or geo.coveredBy(county.geo)
```

(b) An example of Intra-fact IC. This data IC ensures that the "unit_price" measure value is equal to the sale amount divided by the sold quantity ("price/quantity"). It is an Intra-fact IC because it checks the value of a sales fact at the same time.

```
context Sales inv CorrectUnitPriceValue:
unit_price = price/quantity
```

6.8.2Semantic aggregation constraints

Several authors have studied how measure values should be aggregated along dimension hierarchies to ensure correct aggregates, an issue that is often referred to as *summarisability*[34] in the literature. They identified a number of semantic aggregation constraints[35]. These constraints apply to all data cubes and are based on the semantics (characteristics) of the MD elements involved in aggregations (i.e. measures, aggregate functions, and dimension hierarchies). For example, population numbers and temperatures cannot be summed along time.

In all of the existing works, modeling and implementation tasks of such constraints are performed by designers and developers. In our proposal, designers and developers are relieved from these tasks. *These constraints are defined using OCL at the metamodel level: they are a part of our MagicDraw profile*. When designers define the aggregation model, they are checked by MagicDraw at the conceptual abstraction level before generating the implementation of the aggregation model (see Section 6.9). *We highlight here that this novel approach to address such constraints is rendered possible because our profile provides the appropriate support to represent aggregation aspects of (S)OLAP applications*.

---

[32] Spatial OCL is a spatial extension of OCL, which is used to define the expected topological relationships between simple and complex geometries [Duboisset et al., 2005; Pinet et al., 2007].

[33] Spatial OCL2SQL is an extension of the OCL2SQL code generator developed by the Dresden University of Technology [Demuth et Hussmann, 1999; Demuth et al., 2001] Spatial OCL2SQL generates SQL integrity check mechanisms (e.g. queries and triggers) from Spatial OCL expressions [Duboisset et al., 2005; Pinet et al., 2007].

[34] Summarizability refers to the possibility of accurately computing aggregate values at a coarser level of detail from values at a finer level of detail [Mazón et al., 2009].

[35] These studies refer to these aggregation constraints using different terms, including type compatibility conditions and semantic aggregation rules.

6.8.2.1 OCL specifications of main semantic aggregation constraints

We present here the OCL specifications of main semantic aggregation constraints identified in the literature and also of some new ones (b, f, and g). They are expressed in terms of the profile metamodel to ease their understanding; their MagicDraw counterparts are slightly different (e.g. "self.ownedMember(m|m.oclIsTypeOf(AggRule))" corresponds to "self.ownedAggRule" in the constraint (a)).

(a) Correct order of aggregators. In [Horner et al., 2004], it is stated that «For a given measure making the sum of averages, the minima and maxima do not make sense». This constraint is expressed using OCL on the context of the *BaseIndicator* stereotype, as follows:

```
context BaseIndicator
def: SumRules: Set{AggRule} = self.ownedAggRule->select(r|
r.aggregator.name = 'Sum')
def: AvgMaxMinRules: Set{AggRule}= self.ownedAggRule->select(r|
r.aggregator.name = 'Avg' or r.aggregator.name = 'Max' or
r.aggregator.name = 'Min')
inv SumBeforeAvgMinMax: SumRules->forAll(rSum|
AvgMaxMinRules->forAll(rAvgMaxMin| rSum.order < rAvgMaxMin.order))
```

(b) Aggregator and aggregated attribute type conformity. The type of aggregator (e.g. spatial and numerical) must conform to the data type of the aggregated attribute. For example, a spatial measure can be aggregated only with spatial aggregators. This new constraint is expressed as follows:

```
context AggRule inv AggregatorAndAggAttributeTypeConformity:
not(baseIndicator.aggregatedAttribute.OclIsUndefined()) implies
aggregator.applicableTo->exists(dt|
 baseIndicator.aggregatedAttribute.oclIsTypeOf(dt))
```

(c) Aggregation of fact instances as objects. In our approach, if decision makers want to count fact instances, they can create base indicators that are associated with any SDW attribute (the *aggregatedAttribute* tagged value is set to NULL). In this case, only Count and DistinctCount aggregate functions can be applied [Golfarelli et al., 1998]. This rule is guaranteed by the following OCL constraint:

```
context AggRule inv CountIfNullAggregatedAttribute:
if (baseIndicator.aggregatedAttribute->oclIsUndefined())
then (aggregator.name = 'Count' or aggregator.name = 'DistinctCount')
```

(d) Aggregation of value per unit measures. If the measure is of a value per unit additivity type, then Sum cannot be applied [Lenz et Shoshani, 1997].

```
context AggRule inv notSumValuePerUnitMeasure:
if (baseIndicator.aggregatedAttribute.OclIsKindOf(Measure) and
baseIndicator.aggregatedAttribute.addType = AdditivityType::ValuePerUnit)
then aggregator.name <> 'Sum'
```

(e) Aggregation of stock measures along temporal dimensions. The following OCL constraint verifies that measures of type stock are not aggregated using the Sum on temporal dimensions [Lenz et Shoshani, 1997].

```
context DimensionAggRule inv notSumStockMeasurePerTime:
if (baseIndicator.aggregatedAttribute.OclIsKindOf(Measure)and
baseIndicator.aggregatedAttribute.addType = AdditivityType::Stock and
onDimension->exists(d|d.oclIsTypeOf(TemporalDimension)))
then aggregator.name <>'Sum'
```

(f) Aggregation of multi-granular measures. As previously stated in Section 6.5.2.3, if a multi-granular measure is aggregated using aggregate functions that are different from its ETL aggregator (*ETLOperation*), then a disaggregation function must be specified for this measure.

```
context Measure inv aDisaggregatorIf:
if self.owingFact.owingHypercube.ownedIndicator->exists(i|
 i.oclIsTypeOf(BaseIndicator) and
 i.aggregateAttribute = self and
 i.ownedAggRule->exists(r| r.aggregator <> self.ETLOperation))
then not self.disaggregator.oclIsUndefined()
```

(g) At least a base indicator per measure. In our approach, we consider that it is meaningless to define a measure without specifying how it can be aggregated (one or more indicators). Consequently, we state that each measure must be associated with at least one base indicator.

```
context Measure inv AtLeastOneIndicatorPerMeasure:
self.owingFact.owingHypercube.ownedIndicator->select(i|
i.oclIsTypeOf(BaseIndicator))->exists(i|i.aggregatedAttribute = self)
```

6.8.2.2 An example of an incorrect aggregation model

As stated previously, all of these constraints are verified by MagicDraw at the conceptual abstraction level before generating the OLAP implementation schema (see Section 6.9). For example (Figure 6.18), if we define a base indicator, "Sum_Unit_Price", which specifies the Sum for aggregating the value per unit measure "unit_price", MagicDraw (according to the constraint (d) presented above) returns the error message, "*Do not sum value per unit measures*" when validating the conceptual MD model.

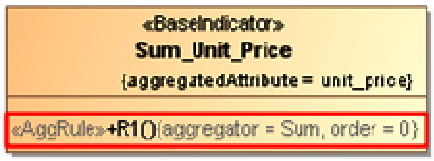


Figure 6.18. An example of an incorrect base indicator.

It is important to note here that, in additional to these common semantic aggregation constraints, decision makers may express aggregation constraints that are specific to their SDW application. For example, they may want to forbid some users from performing some aggregations that reveal strategic information.
Finally, similar to Metadata and Exploration ICs, these user aggregation constraints require some extensions of our profile or OCL to ease their conceptual design and to allow their automatic implementations.

6.9 Implementation

In this section, we show how conceptual (spatial) MD models designed using our profile can be implemented following a ROLAP architecture composed of the spatial DBMS Oracle Spatial and the OLAP Server Mondrian.

ROLAP architectures use relational DBMSs (e.g. Oracle Spatial) to maintain data in tables and OLAP servers (e.g. Mondrian) to build logical OLAP structures (e.g. data cubes and hierarchies) on the top of the DBMSs. These implementations require the definition of two parts (or two types of metadata):

- the Relational DB schema that models the persistent storage of MD data in terms of tables and columns and defines integrity check mechanisms (e.g. SQL triggers) to verify the consistency of these MD data and
- the OLAP schema that defines OLAP structures (data cubes, dimensions, indicators and their aggregation rules) and their mapping to the DB structures.

The following sub-sections describe the implementation of these two parts for the retail MD model designed using our profile in Section 6.7.2.

6.9.1Implementation of the SDW and data ICs

The Relational DB schema can be defined following three schema types: star, snowflake or starflake. In the star schema, every dimension is represented by one table containing all of the dimensional attributes of the dimension's aggregation levels as columns. In the snowflake schema, dimensions are normalised, and each aggregation level is mapped into one table. The starflake schema combines the two representations by normalising some dimensions or parts of them and denormalising others. In all of these schema types, each fact class is represented by one table that references the dimension tables by using foreign keys.

The well-known constellation schema (a constellation of stars, or snowflakes, or starflakes) is obtained when the MD model is composed of two or more hypercubes, which eventually share some dimensions.

The choice between normalising and denormalising dimensions is often based on the storage cost and the expected query performance. In contrast to denormalised dimensions, normalised dimensions are easy to maintain and optimise the storage space; however, they decrease the query performance because many joins need to be performed when executing queries [Malinowski et Zimányi, 2008].

The mappings of conceptual MD models to these implementation schemas have already been investigated in many studies [Glorio et Trujillo, 2008; Malinowski et Zimányi, 2008]. Specifically, different solutions for implementing multi-granular measures are proposed in [Iftikhar et Pedersen, 2010].

In Figure 6.19, we present an excerpt of the star schema implementing the retail conceptual MD model in Oracle Spatial.

```
create table SALES (
/* Measures */
 PRICE double,
 QUANTITY_SOLD double,
 /* Etc. */
/* Foreign keys to dimension tables */
 STORE_ID integer not null,
 foreign key(STORE_ID) references STORES(STORE_ID),
 PRODUCT_ID integer not null,
 foreign key(PRODUCT_ID) references RODUCTS(PRODUCT_ID),
  /* Etc. */
);
```

(a) Sales fact table

```
create table STORES (
/* Store aggregation level */
 STORE_ID integer primary key,
 STORE_NAME varchar2(30),
/* City aggregation level */
 CITY_ID integer not null,
 CITY_NAME varchar2(30),
 CITY_GEO MDSYS.SDO_GEOMETRY,
/* State aggregation level */
 STATE_ID integer not null,
 /* Etc. */
);
```

(b) Stores dimension table

Figure 6.19. Excerpt of the retail star schema.

The integrity check mechanisms that implement the user-defined ICs, which are designed following our conceptual framework (i.e. the classification proposed in [Boulil et al., 2010a]), can be generated using MDA-based code generators such Spatial OCL2SQL, as demonstrated in [Boulil et al., 2010a].
As an example, the SQL query implementing the Spatial OCL IC (a) presented in Section 6.8.1 is depicted in Figure 6.20. This query returns the tuples of the "STORES" table that do not satisfy the constraint.

```
select * from STORES SELF
where not (
       MDSYS.SDO_RELATE(SELF.STATE_GEO, SELF.CITY_GEO,
              'mask=CONTAINS querytype=WINDOW')= 'TRUE'
        OR  MDSYS.SDO_RELATE(SELF.STATE_GEO, SELF.CITY_GEO,
              'mask=COVERS querytype=WINDOW')= 'TRUE'
      );
```

Figure 6.20. Oracle Spatial SQL Query implementing the constraint (a).

It is worth noting here that the ICs' implementations depend on the DB schema type (star, snowflake or starflake). In star schemas, all of the SQL queries that implement all of the ICs that involve different levels of the same dimension are defined on one table (i.e. the table that implements the dimension), while in snowflake schemas, the constraints are checked on the tables that implement the levels they involve. The implementation of ICs in snowflake schemas is detailed in [Boulil et al., 2010a].

6.9.2Implementation of the aggregation model

We have implemented a code generator that supports our aggregation model. In this sub-section, we describe how our tool automatically implements the aggregation model in terms of a Mondrian XML schema. Mondrian[36] is an open-source OLAP server that is part of the Pentaho Business Intelligent Suite. Mondrian builds OLAP logical structures (e.g. data cubes, dimensions and measures) on the top of any DBMS using a specific XML file. It is based on the pseudo-standard OLAP query language, MDX (MultiDimensional eXpressions language)[37].
The novelty of our proposal is the implementation of complex indicators using MDX formulas. Indeed, to the best of our knowledge, only two studies [Hahn et al., 2000; Pardillo et al., 2010a] address the derivation of OLAP metadata from conceptual MD models; these studies assume the aggregate function Sum by default and do not address other aggregate functions.
Thus, supposing that we have the Oracle Spatial Relational DB star schema presented above (Figure 6.19), we automatically generate the Mondrian schema using a Java-based tool developed in the Eclipse IDE. This Java tool takes as input the XMI file of the spatial MD model designed using our profile and generates the Mondrian schema with an ad hoc MDX formula for each indicator and its aggregation rules (*AggRule*, *DimensionAggRule*, *HierarchyAggRule* and *LevelToLevelAggRule*).

Figure 6.21 shows an excerpt of the generated Mondrian schema representing the OLAP dimensions and the hierarchies of the sales hypercube. The mappings of these OLAP structures to the star DB schema tables and columns are also indicated.

[36] http://www.pentaho.com
[37] http://msdn.microsoft.com/en-us/library/ms145506.aspx

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Schema name="RetailCompanySDW">
    <!-- The Mondrian definition of the sales hypercube -->
  - <Cube name="SalesHypercube">
      <Table name="SALES" />
      <!-- The sales hypercube dimensions -->
    - <Dimension name="Stores" foreignKey="STORE_ID">
      - <Hierarchy hasAll="true" name="StoreLocation" primaryKey="STORE_ID">
          <Table name="STORES" />
          <Level type="Numeric" name="Country" column="COUNTRY_ID" nameColumn="COUNTRY_NAME" />
          <Level type="Numeric" name="State" column="STATE_ID" nameColumn="STATE_NAME" />
          <Level type="Numeric" name="City" column="CITY_ID" nameColumn="CITY_NAME" />
          <Level type="Numeric" name="Store" column="STORE_ID" nameColumn="STORE_NAME" />
        </Hierarchy>
      </Dimension>
    + <Dimension name="Products" foreignKey="PRODUCT_ID">
    + <Dimension name="Time" foreignKey="MONTH_ID">
    + <Dimension name="Customers" foreignKey="CUSTOMER_ID">
      <!-- The Mondrian definitions of the sales indicators -->
```

Figure 6.21. The retail Mondrian schema – The sales hypercube dimensions.

In the rest of this section, we detail the MDX implementations of the different types of indicators that are designed for sales analysis. Note that these MDX formulas are part of the retail Mondrian schema (Figure 6.21).

A) The base indicators with only one aggregation rule of type *AggRule*

As shown in Figure 6.22, each simple base indicator (e.g. "Total_Quantity_Sold") is implemented as a Mondrian measure element by indicating the aggregated DB column ("QUANTITY_SOLD") and the aggregator (Sum). These two XML properties correspond respectively to the *aggregatedAttribute* and *aggregator* tagged values of the conceptual base indicator (see Figure 6.12).

```
<!-- The base indicators containing only one aggregation rule of type AggRule -->
<Measure name="Total_Quantity_Sold" column="QUANTITY_SOLD" aggregator="sum" formatString="Standard" />
<Measure name="Turnover" column="PRICE" aggregator="sum" formatString="Standard" />
<Measure name="Total_Costs" column="COSTS" aggregator="sum" formatString="Standard" />
```

Figure 6.22. The retail Mondrian schema – Simple base indicators.

B) The base indicators with at least one aggregation rule of type *DimensionAggRule*

Each base indicator of this type (e.g. "Stock_Level" in Figure 6.13) is implemented using a calculated member element (Figure 6.23). In this example, the MDX formula returns the average of "STOCK_QUANTITY" if the "Time" dimension is selected and its sum if others are selected. Note that this calculated member uses an intermediate measure element ("quantity_stock_Sum") whose visibility is set to false.

```
<!-- The base indicator Stock_Level -->
<Measure name="quantity_stock_Sum" column="QUANTITY_STOCK" aggregator="sum"
  formatString="Standard" visible="false" />
<CalculatedMember name="Stock_Level" dimension="Measures" formatString="#,###.##">
  <formula>Avg(Descendants([Time].CurrentMember, [Time.QuarterHierarchy].[Month]),
    [Measures].[quantity_stock_Sum])</formula>
</CalculatedMember>
```

Figure 6.23. The retail Mondrian schema – The Stock_Level base indicator.

C) The base indicators with aggregation rules of type *HierarchyAggRule*

Like indicators of type B above, these indicators are implemented using calculated member elements. For the exemplified indicator (Figure 6.24), the MDX formula returns the average of sold quantities if the "CustomerLocation" hierarchy is selected and their sum if the others are selected; this computation is performed using the "Total_Quantity_Sold" measure element that has already been defined (Figure 6.22).

```
<!-- The base indicator Price_SumDefault_AvgOnCustomerLocation -->
- <CalculatedMember name="Price_SumDefault_AvgOnCustomerLocation" dimension="Measures"
    formatString="#,###.##">
    <formula>IIf(([Customers].CurrentMember.Hierarchy.Name = "Customers.CustomerLocation"),
      Avg(Descendants([Customers].CurrentMember, [Customers.CustomerLocation].[Customer]),
      [Measures].[Total_Quantity_Sold] ), [Measures].[Total_Quantity_Sold] )</formula>
  </CalculatedMember>
```

Figure 6.24. The retail Mondrian schema – The Price_SumDefault_AvgOnCustomerLocation base indicator.

D) The base indicators with aggregation rules of type LevelToLevelAggRule

These indicators are also defined as calculated member elements. For example, the "MaxTurnoverPerCity" base indicator represented in Figure 6.15 (Section 6.7.2.2.2.4) is implemented by the calculated member shown in Figure 6.25. The MDX formula returns the maxima of city turnovers if levels from "State" to "All" are selected; otherwise, it returns the "Turnover" measure defined above in Figure 6.22.

```
<!-- The base indicator MaxTurnoverPerCity -->
- <CalculatedMember name="MaxTurnoverPerCity" dimension="Measures" formatString="#,###.##">
    <formula>IIf((([Stores].Level.Name = "(All)") OR ([Stores].Level.Name = "Country") or
      ([Stores].Level.Name = "State") ), Max(Descendants([Stores].CurrentMember,
      [Stores.StoreLocation].[City]), [Measures].[Turnover]), [Measures].[Turnover] )</formula>
  </CalculatedMember>
```

Figure 6.25. The retail Mondrian schema – The MaxTurnoverPerCity base indicator.

E) The derived indicators

Finally, we also implement derived indicators by calculated member elements. For example, the derived indicator "Profit", whose conceptual design is shown in Figure 6.16 (Section 6.7.2.2.3), is implemented by the calculated member depicted in Figure 6.26. The MDX formula defines a subtraction of "Total_Costs" from "Turnover". These two measure elements are defined in Figure 6.22 above.

```
    <!-- The derived indicator Profit -->
  - <CalculatedMember name="Profit" dimension="Measures" formatString="#,###.##">
      <formula>[Measures].[Turnover]-[Measures].[Total_Costs]</formula>
    </CalculatedMember>
  </Cube>
```

Figure 6.26. The retail Mondrian schema – The Profit derived indicator.

6.10Conclusions and Perspectives

Data Warehouses (DWs) and OLAP tools are often used in BI applications in support of decision-making processes [Kimball et Ross, 2002]. DWs provide OLAP tools with integrated and historical data stores that are structured in a manner that reveals the inherent multidimensionality of decisional data. Based on some metadata, OLAP tools query these data stores and construct analytical models (cubes, dimensions, statistical summaries or indicators) that transform raw business data into strategic business insights. These tools implement OLAP operators (e.g. Roll-up) to allow decision makers to navigate the constructed analytical models.

On the other hand, Spatial Data Warehouses (SDWs) and Spatial OLAP (SOLAP) tools were introduced with the aim of enriching OLAP with spatial analysis by taking into account the geographical part of the increasing volumes of geo-referenced data, which are generated by means of several technologies (e.g. sensor networks) [Bédard et al., 2006]. SOLAP combines OLAP analysis with cartographic visualisations of GIS systems, allowing understanding of the geographical distributions of analytical data. SOLAP extends OLAP with spatial OLAP operators (e.g. Spatial Roll-up) to allow interactive spatio-multidimensional explorations of analytical data.

In the literature, the conceptual design of DW and SDW systems has been largely studied, and many MD models have been proposed. Mainly, extensions of standard formalisms of conceptual database modeling (ER and UML) are suggested. The reason is that these formalisms require much less learning and modeling effort than ad hoc ones. Nevertheless, an increasing number of authors introduce UML-based models because this language provides powerful constructs to better represent static and dynamic aspects of complex applications. Another outstanding advantage of UML is that it can be easily extended to adapt it to specific domains or platforms.
In general, all of the existing MD models (spatial or not) focus on the design of the warehouse structures and ignore the aspects related to the aggregation of data, which are central to DW applications. In addition to some limitations concerning the modeling of the warehouse model part (e.g. multi-granular measures and data quality), these models do not provide the appropriate support for the conceptual design of the aggregation model part (indicators and their aggregation rules).

In this work, motivated by the lack of a standard formalism for the conceptual design of (spatial) data cubes that takes into account these two parts (warehouse and aggregation model parts), we have proposed a new UML profile, and we have implemented it in the UML-based tool called MagicDraw. This profile, in addition to satisfying the main classical modeling requirements identified in the literature (e.g. explicit complex hierarchies, and aggregate functions; see [Abelló et al., 2006]), meets some new important mandatory requirements: (a) the modeling of indicators and their aggregation rules, (b) the aggregate functions' typing, (c) support for multi-granular measures and (d) support for data quality issues.
As highlighted throughout this chapter, Integrity Constraints (ICs) are an excellent way to address data and aggregation quality issues, which are critical in decisional applications. By providing explicit support for the main MD components (warehouse and aggregation components), our profile facilitates the conceptual expression of the different types of ICs (identified in [Boulil et al., 2010a]). Moreover, being UML-based, it supports OCL, allowing for automatic implementations of theses ICs using OCL-based code generators such as Spatial OCL2SQL.
Specifically, exploiting the aggregation support and the precise typing of MD elements provided by this profile, we have proposed an interesting approach to address some aspects of data aggregation quality at the conceptual abstraction level. The main semantic aggregation constraints identified by the literature are defined using OCL and are implemented as part of our profile. This enables the semantic correctness of the aggregation models defined by designers to be checked at the conceptual level before generating their implementations in terms of OLAP schema metadata.
Finally, we have proposed the automatic implementation of the aggregation MD model part in terms of a Mondrian OLAP schema using a Java-based tool developed in the Eclipse IDE. The novelty of our implementation proposal is the implementation of complex indicators and their aggregation rules in terms of MDX formulas in the Mondrian OLAP server.

We are now investigating the definition of an extension of our profile to allow decision makers to express their own aggregation constraints (referred to as User-defined aggregation constraints in [Boulil et al., 2010a]). These constraints are specific to the application domains and may express, for example, the allowed (or forbidden) aggregate functions for some combination of measures and dimension hierarchies. Moreover, proposing implementations in the OLAP tier (client or server) that will check these constraints when decision makers formulate ad hoc queries would be an interesting issue to investigate.
Finally, as our topic of interest is data and aggregation qualities in SDW applications, our future research will most likely be the definition of other extensions of our profile to

conceptually express the metadata and exploration ICs. In the same spirit, we will investigate their implementations in the SOLAP architecture tiers.

7 Chapitre 7 : Towards the Definition of Spatial Data Warehouses Integrity Constraints with Spatial OCL

7.1 Preamble

Spatial Data Warehouses (SDWs) and Spatial OLAP systems are an effective solution to perform multidimensional spatial analysis on geographical data. However, the quality of such analysis heavily depends on the quality of stored data. Due to this, some works have been attempted to address the issues of data quality of SDWs using Integrity Constraints (ICs). In this chapter, we propose a conceptual framework for SDW ICs based on two new classifications. Moreover, we describe the specification and implementation of some IC classes using the Object Constraint Language (OCL) and an automatic code generator, called Spatial OCL2SQL.

7.2 Introduction

In this chapter, motivated by an automatic implementation of SDW ICs, we first propose two complementary and orthogonal classifications of SDW ICs. Our classifications extend and cover all existing ones, and they constitute a reference framework for the definition of SDW ICs. In the first classification, ICs are grouped with respect to their implementation level in the SOLAP architecture. In the second one, ICs are categorized based on which multidimensional elements (e.g. fact, member, etc.) they involve, and mapped on the implementation classification. After that, using examples, we demonstrate that many ICs can easily be specified using OCL and Spatial OCL at the conceptual level. Spatial OCL [Duboisset, 2007], a spatial extension of OCL, is used to define the expected topological relationships between simple and complex geometries. Finally, we show how these OCL specifications may be automatically implemented using an MDA-based tool, called Spatial OCL2SQL.

The remainder of this chapter is organized as follows. Section 7.2 surveys and discusses related work. The case study is introduced in Section 7.3. In Section 7.4, we describe the two proposed SDW IC classifications, and in Section 7.5, we show some examples of OCL and Spatial OCL specifications of constraints. In Section 7.6, the automatic implementation of these ICs in Oracle Spatial 11g is presented. Finally, Section 7.7 concludes and outlines our future work.

7.3 Related work

In conventional data warehouses, several works have addressed the issues of data quality by using integrity constraints. [Carpani et Ruggia, 2001] propose the first multidimensional conceptual model supporting ICs. They use logical predicates to specify (at a conceptual level) ICs involving one or several dimension members, several dimension levels, and several dimensions. [Ghozzi et al., 2003b] propose a multidimensional model extended with specific constraints, called intra-dimension and inter-dimension constraints. These constraints define inclusions and exclusions between hierarchies of one or several dimensions, and they are specified using logical expressions at a conceptual level. [Mazón et al., 2007] define a set of QVT (Query/View/Transformation language) relations based on

multidimensional normal forms to verify and to enforce the correctness of the data warehouse conceptual model w.r.t the data source model. A survey on summarizability[38] problems is presented in [Mazón et al., 2009]. The authors suggest that summarizability issues related to disjointness and completeness conditions [Lenz et Shoshani, 1997] can be expressed as multiplicity constraints at the schema level; but, they do not consider the type compatibility condition [Lenz et Shoshani, 1997].

In the context of SDWs, [Malinowski et Zimányi, 2008] present a spatial multidimensional conceptual model, called Spatial MultiDimER, extending the Entity/Relationship model. They namely use pictograms to express topological spatial relationships (e.g. spatial inclusion) between spatial levels. They also implement these constraints with triggers and SQL views for Oracle Spatial. [Glorio et Trujillo, 2008] propose a UML profile for SDWs; but they consider a very small number of ICs. [Bimonte et al., 2009] provide a logical multidimensional model for SDWs, which takes into account a new kind of constraints defining dependencies between spatial and alphanumeric aggregation functions.

Finally, although the definition of ICs to improve data quality in SDWs is a very important issue, not enough works have been done in this area. The author in [Salehi, 2009] proposes the most detailed and complete classification (w.r.t to the above works). Nevertheless, this classification can be extended to consider other ICs, such as those involving several (spatial) hypercubes, those related to spatial metadata, etc. In addition, none of the existing proposals uses a *standard language* such as OCL to specify SDW ICs at the conceptual level. The use of a standard language will ease the implementation of ICs following an MDA approach. Finally, the majority of these works have paid little attention to the *code generation*. Indeed, none of them implements SDW ICs using existing MDA-based tools.

7.4 Case study

The case study used to exemplify our work concerns the spatial multidimensional analysis of damages caused by fire disasters. This example is issued from [Salehi, 2009]. This model is described by the UML class diagram of Figure 7.1. This SDW model defines three dimensions and one fact described by three measures. The first dimension, "Time", is organized following three levels: "Day", "Month" and "Year". The spatial dimension, "Location", has two exclusive hierarchies: "USA location" and "Canadian location". The former consists of levels "City", "County", "State", and "Country". The later consists of levels "City", "County", "Province", and "Country". Similarly, the third dimension, "Fire class", has two exclusive hierarchies, "USA fire class" and "Canadian fire class". The levels of the three dimensions have a number of descriptive attributes. For example, the spatial level "City" has a geometric attribute, called "location", which represents geometrically city locations. The measures are: (i) "fire-zone", which represents geometrically locations of fire zones, (ii) "number of injuries", and (iii) "destroyed-area", which represents areas (km^2) of destroyed residential zones.

[38] Summarizability refers to the possibility of accurately computing aggregate values with a coarser level of detail from values with a finer level of detail [Mazón et al., 2009].

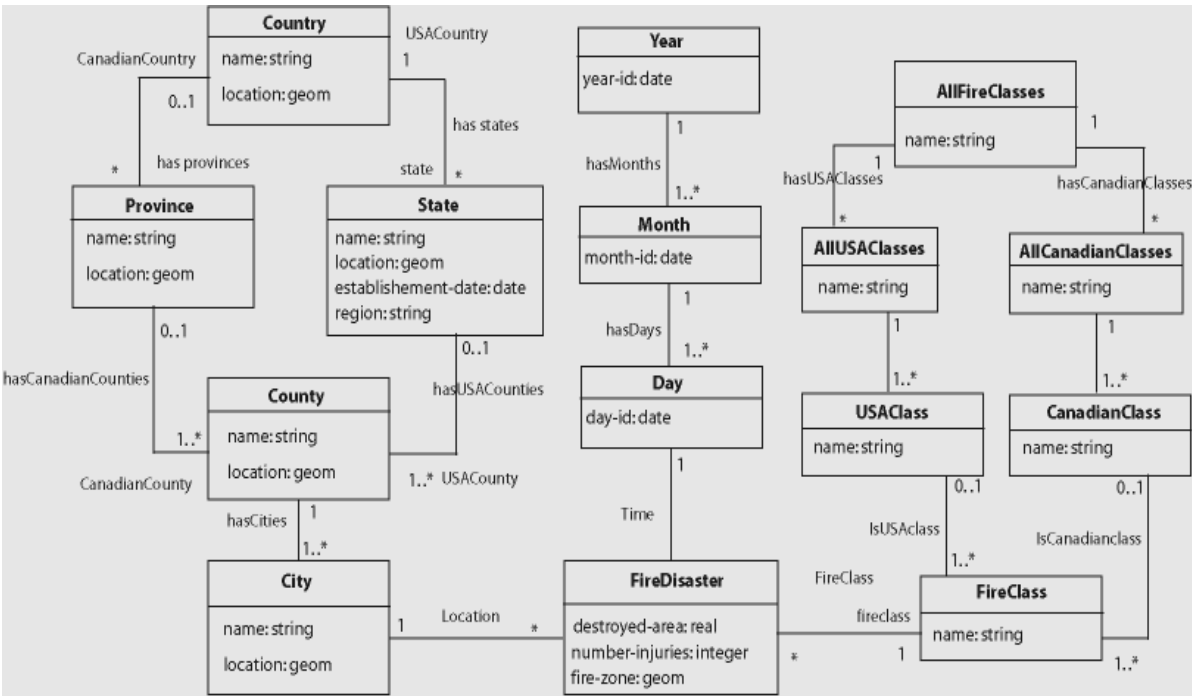


Figure 7.1. The "fire disaster" multidimensional model. Adapted from [Salehi, 2009].

7.5 New classifications of SDW ICs

We present in this section two new orthogonal and complementary classifications: (1) *implementation-based classification*, which categorizes ICs according to their level of implementation in the SOLAP architecture (see Section 7.4.1); and (2) *SDW concept-based classification*, which groups ICs on the basis of which multidimensional elements they involve (see Section 7.4.2). They stand for a reference framework to ease identification, MDA-based expression and implementation of SDW ICs. Indeed, for each IC, its conceptual expression depends on the involved SDW concepts; and its physical translation depends on the SOLAP tier where it is implemented. The later is indicated by the proposed mapping between the two classifications (see Figure 7.2). For example, Computational SDW ICs can only be implemented in the SOLAP tier (see Figure 7.2), as they are used to guarantee correct aggregation results when performing multidimensional queries.

7.5.1Implementation-based classification

For this classification, we distinguish the three following categories: ETL Tier ICs, SDW Tier ICs and SOLAP Tier ICs.

7.5.1.1 ETL tier ICs

These constraints are implemented in the ETL tools in order to prevent inconsistent data to be loaded into the SDW [Mũoz et al., 2009]. These ICs can be integrated in Spatial ETL programs such as Spatial Data Integrator[39] jobs, GeoKettle[40] transformations, etc.

7.5.1.2 SDW tier ICs

They are often implemented using SQL views and triggers in the spatial DBMS, because they exclusively verify the validity of warehoused data. For example, these ICs can be used

[39] http://www.spatialdataintegrator.com
[40] http://www.spatialytics.org/projects/geokettle

to identify incorrect fact instances and members. To avoid cumbersome ETL processes, they must be checked once all data are loaded.

7.5.1.3 SOLAP Tier ICs

These constrains are mainly used to guarantee the correct aggregations of measures, and the correct and coherent spatial data cube exploration. They can be implemented either in the SOLAP client or in the SOLAP server. For example, they can be implemented in the query parser of the SOLAP Server.

7.5.2SDW concept-based classification

This classification characterizes ICs according to the SDW elements they involve. Its positioning compared to existing classifications and possible implementation levels for each IC class (defined by our implementation classification) are shown on Figure 7.2. This classification takes into account all previously defined ICs: keeping in mind that any IC generally involves one or many SDW schema attributes, we simplify and extend the most complete classification presented in [Salehi, 2009], by adding metadata ICs and inter hypercube ICs. The main idea is that decision-makers may express the need to prevent the storage and/or the analysis of sets of members or fact instances related to one or several hypercubes. Due to the space limitation, we only show examples for Intra Hypercube ICs.

7.5.2.1 Intra Hypercube ICs

They involve members or fact instances of one hypercube. They are classified in sub-classes as described in the sequel.

7.5.2.1.1Constraints on members

These constrains may be used to identify incorrect members and incorrect relationships between members. They can involve one or several attributes belonging to one or several levels.

7.5.2.1.1.1 Intra Level SDW ICs

They involve members and relationships between members of one level.
Constraint 1: The USA states (members of the level "State") located in the "West" region must be spatially disjoint from the USA states of the "Northeast" region. This IC defines a topological relationship between two sub-sets of members belonging to one level, which is "State".

7.5.2.1.1.2 Inter Level SDW ICs

They involve members of at least two levels.
Constraint 2: Only states (members of "State" level) located in "West", "Midwest" or "Northeast" regions can meet provinces (members of "Province" level).

7.5.2.1.2Constraints on fact instances

These constrains involve one or several fact instances. Thus, they can be intra or inter facts. They can be used either to identify incorrect fact instances or to prevent inconsistent SOLAP queries, involving one or several fact instances, to be executed.

7.5.2.1.2.1 Intra Fact SDW ICs

They involve one or several measures of one fact instance.
Constraint 3: Fire disasters which happen in USA locations cannot be related to "Canadian fire class" hierarchy. Similarly, fires happening in Canadian locations cannot be related to

"USA fire class" hierarchy. This IC defines the correct dimensioning of each one of the "FireDisaster" fact instances.

7.5.2.1.2.2 Inter Fact SDW ICs

They involve at least two fact instances.
Constraint 4: In the USA, the area of destroyed residential zones by fires of class "A", are greater than the area of destroyed residential zones by fires of class "E".

7.5.2.1.3 Summarizability SDW ICs

These constraints ensure correct aggregation of measures in a given spatial hypercube. We distinguish three sub-classes.

7.5.2.1.3.1 Computational SDW ICs

They guarantee a correct computation of aggregated measures in data cubes. More precisely, they define semantically correct and incorrect combinations of the form <*measure, dimension, aggregation function*> [Salehi, 2009]. Such a combination specifies which aggregation functions are allowed to aggregate a given measure among a given dimension. These constraints can also indicate the correct order among the correct combinations of aggregation [Abelló et al., 2006]. For example, in order to discover the worst fire class per city (which causes the greatest number of injuries), we should apply these two ordered combinations: (a) aggregating the number of injuries using the AVG function along the temporal dimension, and then (b) applying the MAX function along the "Fire class" dimension.

7.5.2.1.3.2 Completeness SDW ICs

They ensure the existence of all necessary data (fact instances and members) for correct SOLAP analysis. They coincide partly with the third condition of summarizability presented in [Lenz et Shoshani, 1997]. For example, if only fact instances related to two cities are warehoused, then the aggregation result computed at the "County" level will be not significant for analysis as it is incomplete.

7.5.2.1.3.3 Schema SDW ICs

They are conditions that must hold when defining the dimension hierarchies and the granularity adopted for representing the fact’s measures. They mainly relate to multiplicities of relationships between levels of aggregation in hierarchies, and between fact and dimension levels [Mazón et al., 2009].

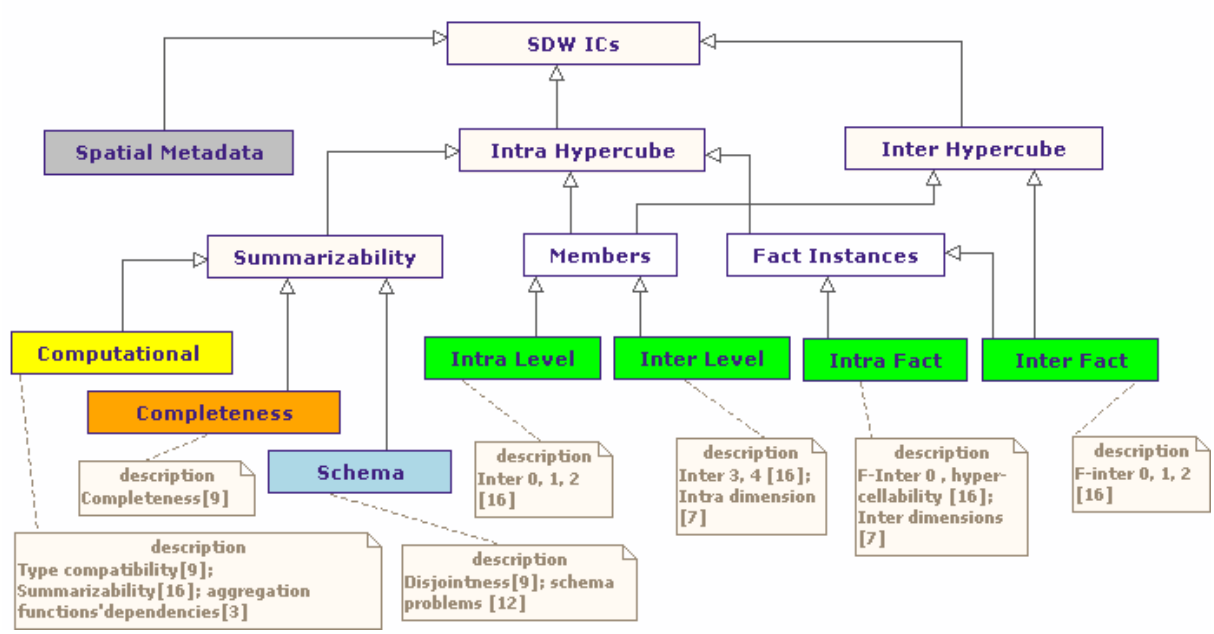


Figure 7.2. The SDW concept-based classification.

7.5.2.2 Inter Hypercube ICs

They involve members and fact instances of at least two hypercubes. They can be specialized in three sub-classes: (a) Intra Level ICs, which involve members of one shared level between two or more hypercubes [Cabibbo et Torlone, 2004]; (b) Inter Level ICs, which imply many levels belonging to different hypercubes; (c) and finally, Inter Fact ICs, which involve several fact instances of various hypercubes. For example, assume that we have another spatial hypercube, called "Flood hypercube", which analyses floods in both USA and Canada countries. This hypercube shares "Time" and "Location" dimensions with the "Fire disaster" hypercube; and defines another dimension called "Flood category", and two measures: "rainfall level" and "flooded zone" (geometric measure). Then, the constraint stating that "*fire and flooded zones respectively related to fire and flood facts happening at the same time do not geographically meet if the rainfall level is greater than a given threshold*" is an Inter Hypercube and Inter Fact SDW IC, as it involves facts of the two spatial hypercubes.

7.5.2.3 Spatial Metadata ICs

Geographic information is characterized by a number of metadata, e.g. coordinate systems, scales, etc. The warehoused spatial data may have incompatible metadata, since they are collected from various heterogeneous data sources. Consequently, constraints on spatial metadata must be defined.

7.6 Spatial OCL specification of SDW Intra Hypercube ICs

In this section due to space reasons, we show how some of the above examples of SDW Intra Hypercube ICs presented in Section 7.4.2.1 can be easily modelled using the Spatial OCL language. Spatial OCL can be used for example to express all the eight topological relationships (disjoint, cover, etc.) of the 9-Intersection Model between simple and complex geometries [Duboisset, 2007; Pinet et al., 2007]. More details about its expressivity can be found in [Duboisset, 2007].

Constraint 1: This IC (see Figure 7.3) is defined using a Spatial OCL expression of type invariant[41] (keyword “inv”). This invariant is specified on the "State" class. Using the Spatial OCL syntax[42], it defines a spatial topological relationship, "areDisjoint", between geometries of two "State" instances, referenced by "self.location"[43] and "s.location" expressions.

```
Context State inv:
State.allInstances -> forAll(s | (
(self.region='West' and
s.region='Northeast') or
(self.region='Northeast' and s.region='West')
 ) implies
 self.location.areDisjoint (s.location))
```

Figure 7.3. The Spatial OCL expression of the Constraint 1.

Constraint 3: This constraint (see Figure 7.4) is applied to one instance of "FireDisaster" class. It checks that the current fact instance ("self") is not simultaneously related to an USA city and a Canadian fire class; and it is not simultaneously related to a Canadian city and a USA fire class. The expression "State.allInstances.county.city.firedisaster" returns the collection of fact instances related to the USA locations. Similarly, the expression "CanadianClass.allInstances.fireclass.firedisaster" returns the collection of fact instances related to the Canadian fire classes. Finally, the OCL expression "collection -> excludes (self)" returns true if the object "self" does not belong to the collection.

```
context FireDisaster inv :
(State.allInstances.county.city.firedisaster
   -> excludes(self) xor
CanadianClass.allInstances.fireclass.firedisaster
   -> excludes(self) ) and
(Province.allInstances.county.city.firedisaster
   -> excludes(self) xor
USAClass.allInstances.fireclass.firedisaster
   -> excludes(self) )
```

Figure 7.4. The Spatial OCL expression of the Constraint 3.

7.7 Implementation

In order to automatically implement the Intra Hypercube ICs and the SDW data model in the Oracle Spatial 11g DBMS, which is the specific implementation platform in our MDA architecture, we have used the MDA-based code generator called Spatial OCL2SQL.

The Spatial OCL2SQL is a Java open source tool which integrates the spatial extensions of OCL, called OCL 9IM and OCL ADV [Duboisset, 2007; Pinet et al., 2007]. It automatically generates SQL scripts for Oracle Spatial from OCL constraints. Its main inputs are: (i) a conceptual data model defined by an XMI[44] schema, (ii) an OCL constraint file and (iii) a geometric metadata file. The tool outputs SQL scripts to create the SDW physical schema in

[41] An OCL invariant expresses a constraint or a condition on a set of objects that must be true at any time.
[42] In Spatial OCL, the expression “(geomA).topological-relationship (geomB)” returns true if the topological relationship occurs between geomA and geomB [Duboisset, 2007].
[43] The keyword self is used to reference an instance of the contextual class, i.e. the class specified after the keyword context.
[44] XML Metadata Interchange, an OMG metadata interchange standard [OMG, 2007].

the Oracle Spatial DBMS and a set of SQL triggers and queries to implement the OCL constraints allowing data integrity checks. Details about the transformation rules between Spatial OCL and SQL can be found in [Duboisset, 2007; Pinet et al., 2007].
Our implementation is carried out following four steps: (i) Definition of the XMI schema that represents the SDW conceptual model (see Figure 7.1); (ii) Definition and validation of the (spatial) OCL constraints taking into account the OCL syntax supported by the Spatial OCL2SQL tool and the elements defined in the XMI file; (iii) Definition of the spatial metadata file which is used to update the spatial metadata table of Oracle Spatial; (iv) and finally, Execution of the SQL scripts generated by the tool in Oracle Spatial 11g to implement the SDW schema and the ICs.

7.8 Conclusions

In this chapter, we first propose a reference framework, based on two classifications, to ease identification, conceptual expression and implementation of SDW ICs. A mapping between classes of the two classifications is also defined: for each concept-based class of SDW ICs, we indicate the possible implementation levels (implementation-based classes). This mapping helps us to explicitly define PIM-To-PSM translations (adapted to the involved SOLAP tier) which allow automatic implementation of constraints. After that, we focus on the Intra Hypercube SDW ICs that can be implemented in the SDW tier (Spatial DBMS). In order to allow their MDA-based implementation, we use Spatial OCL language [Duboisset, 2007] to model them at the conceptual level and the MDA-based code generator called Spatial OCL2SQL to enable their automatic implementation in the Oracle Spatial DBMS.
We are now working on the definition and implementation of the other SDW ICs (Spatial Metadata, Inter Hypercube and Summarizability ICs) in order to validate our classifications. For that reasons, we are defining a SDW profile that integrates SDW ICs and offers rich visual notations to represent them at the conceptual level. Using this profile, we aim at defining specific patterns to help designers in defining SDW ICs.
Finally, we are convinced that there is still a lot of work to be done concerning the implementation of SOLAP tier ICs. For example, implementation techniques similar to DMBS triggers could be defined to prevent the execution of incorrect SOLAP queries when exploring the SDW content.

PARTIE IV : EXPERIMENTATIONS

8 Chapitre 8 : Using Data Warehouses for Storage and Visualization of Simulation Model Results

8.1 Preamble

Researchers and stakeholders need efficient tools to store, display, compare and analyze data produced by simulation models. One common way to manage simulation results is to use text files; however, text files make it difficult to explore the data. Spreadsheet tools (e.g. Open Office, MS Excel) can help to display and analyze model results, but they are not suitable for very large volumes of information. In this chapter, we propose to use data warehouse (DW) technology to store model results and to facilitate model visualization/analysis/comparison. This technology allows model users to easily produce graphic reports and charts. We show how a generic DW schema for simulation result data storage can be adapted to a model simulating hydrological transfers of pesticides at the plot scale. We present the different steps that we followed to design and implement DW model storage. The proposed methodology can be reused for other simulation models. Our proposal is addressed to researchers who work in the environmental modeling field and who wish to learn about the possibilities that DW technology offers for simulation results management.

8.2 Introduction

Environmental modeling is extensively used to study complex phenomena, such as urbanization, climate change, pollutants transfer, etc. [Hirabayashi et al., 2011; Li et Mao, 2011; Trolle et al., 2011; Pogson et al., 2012]. These models allow researchers and stakeholders to represent, understand, explain and formulate/validate hypotheses about environmental phenomena to predict their evolution. These tasks usually involve several disciplines and several types of information (spatial, temporal, biological, meteorological, hydrological, economical information, etc.). The analysis of simulation/numerical model results is useful to compare or calibrate models with several sets of observed input data and to provide a global vision of model behavior.

Models can output huge volumes of result data [Fernández-Quiruelas et al., 2011]. Moreover, models can simulate a stochastic process that requires several replications of each simulation to obtain representative results. These replications increase the quantity of result data, which makes exploration and analysis difficult. As shown in [Mahboubi et al., 2010], scientists or model users need to extract aggregated information from large amounts of simulation result data (e.g. regularities or synthetic indicators). Researchers and stakeholders also need to perform comparative analyses of results coming from different sets of input data, versions of models or sets of parameters. For that purpose, they need efficient tools to store, display, compare and analyze simulation results data. One typical method for storing simulation results is to use text files; text-based storage makes it difficult to explore, select, and visualize the data. Spreadsheet tools (for example, Open Office, MS Excel) can help to display and analyze the simulation results, but they are not suitable for very large volumes of information.

In this chapter, we propose to exploit the data warehouse (DW) technology [Berson et Smith, 1997; Martin, 2008; Pinet et Schneider, 2010] to store and facilitate the exploration of simulation result data. Analysis of simulation result data using Spatial OLAP technologies has been firstly investigated in [Ali et al., 2007; McHugh et al., 2007; Chaker et al., 2009]

which constitutes a new and original approach. DWs are dedicated to storing and displaying a very large volume of information (in the same database) via local network or Internet. Furthermore, DWs allow production of graphic reports, charts, etc. DWs are usually relational databases (DBs), so this technology provides efficient methods for user authentication, integrity constraint controls, data backup, data insertion and data selection [Pokorný, 2006] [Basta et Zgola, 2011].

Multidimensional schemas represent the different types of information stored in a DW. Recently, the authors of [Mahboubi et al., 2010] proposed a generic multidimensional schema to facilitate the construction of DWs to store simulation result data. This schema provides the main elements that must be defined during the design of this type of database. This proposal has been tested on only one model, called "Savanna" [Jeltsch et al., 1996; Jeltsch et al., 1998; Jeltsch et al., 1999]. Here, we introduce a new application of this schema: we used the method of [Mahboubi et al., 2010] to design a DW for storage and visualization of data resulting from MACRO [Larsbo et Jarvis, 2003], a model that simulates hydrological transfers of pesticides at the plot scale. This work has been realized in the context of interdisciplinary projects, including researchers from hydrology, agronomy, mathematics and computer sciences. As we will present in this chapter, our DW multidimensional schema has been designed with the UML[45]-based MagicDraw tool [No Magic, 2012] and its extension for DW [Boulil et al., 2012c]. A prototype of DW has been implemented with free software: PostgreSQL [PostgreSQL, 2012], JRubik [Rubik, 2012], and Mondrian [Pentaho, 2012b].

This work began as a part of the project "Environmental Information Systems for pesticides" (2007-2010) [Miralles et al., 2011], financed by Irstea, and has continued as a part of the "Miriphyque" project, financed by the French Ministry of Ecology, to support for risk assessment and management (2010-2013) [Carluer et al., 2011]. Miriphyque aims to build a method for evaluating the potential for pesticide contamination of surface waters across a small catchment, while taking into account the influence of landscape elements and the spatial and temporal dimensions; typically, these include the position of plots and landscape features in the watershed, the technical itineraries and the chronic of hydro-meteorological events. This method will be simpler than the complex hillslope models and more sophisticated than simple risk indicators that have been poorly developed for application to catchments. The originality of the approach lies in its hybrid property, i.e. associating indicators and hydrological modeling at the catchment scale. Indeed, using a DW to store all of the model simulation results is a natural step, as this technology allows for easy handling of a large amount of data.

Our proposal is addressed to environmental researchers who wish to learn about the main advantages that DWs offer for storage of simulation results. Section 8.2 introduces the main DW concepts, section 3 discusses the generic schema of [Mahboubi et al., 2010], section 4 shows how we have applied it to the MACRO model, and finally, section 5 summarizes the different steps that we followed to build our DW. The methodology presented in this chapter could be reused for building DWs for other models.

[45] UML: Unified Modelling Language [Booch et al., 1999].

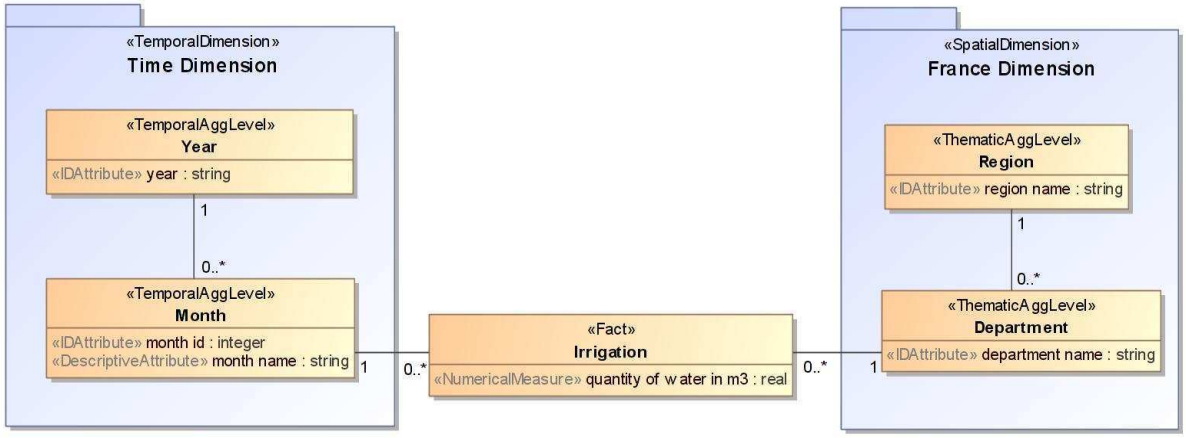


Figure 8.1. An example of a multidimensional schema.

8.3 What is a Data Warehouse?

A DW is usually represented by a multidimensional schema [Pinet et Schneider, 2010]. Figure 8.1 shows an example of a multidimensional schema defined with the UML-based formalism proposed in [Boulil et al., 2012c]. Data that will be visualized and analyzed by DW users are called *facts*. In the example, the facts are information about irrigation. In the UML representation of Figure 8.1, all of the facts are instances (i.e. objects) of the class marked "<<Fact>>". A fact can contain one or several attributes, which are numerical *measures* used for data analysis. In the example, the measure for analysis is the amount of water (in m^3) used for irrigation in agriculture. The measures stored in the warehouse are the quantities of water per French administrative Department[46] per month.

In DW, queries allow for aggregation of measures (e.g. sum or average). In the example, the total amount of water used for irrigation can be calculated at different *levels*: by year and/or French administrative Region. A hierarchy created by one or several levels is called a *dimension* of analysis. Here, we have two dimensions: one temporal dimension (Month < Year) and one spatial dimension (Department < Region). In DW, aggregations can be calculated for each combination of levels. For example, if the sum aggregation function is chosen by the user, the quantity of water can be calculated by any of the following: per month; per year; per Department; per Region; per Month and Department; per Month and Region; per Year and Department; or per Year and Region.

Usually, relational DB management systems (e.g. PostgreSQL, Oracle, MySQL) are used to store DWs. On-Line Analytical Processing (OLAP) tools [Berson et Smith, 1997] enable visualization of data and aggregation results in tables with multiple entries (i.e. pivot tables). To compose a table with OLAP tools, users choose a measure, an aggregation function and different levels of dimensions. For example, suppose a user creates a table with the measure "quantity of water in m^3", the sum function and the levels "Department" and "Year". Figure 8.2 presents the resulting table (showing an aggregation by year and Department). Each table cell contains the total amount of water used for each pair {Department, Year}. These values are calculated from the DW using the finest granularity found in the data, i.e. the amount per month and Department. Now, suppose instead that a user chooses the levels "Region" and "Year"; Figure 8.3 shows the resulting table. The addition of new levels and new dimensions

[46] In the geographical organization of France, an administrative Region is subdivided into several administrative Departments. France is composed of 27 Regions and 101 Departments.

to a DW will increase the number of possible tables. The OLAP systems also allow users to visualize results using graphic displays (graphs, pie-charts, etc.).

Here, we sum up the main components of DW architecture [Berson et Smith, 1997]. Typically, data extracted from external sources is cleaned and loaded in the DW in using *Extraction-Transformation-Loading (ETL)* tools (e.g. Talend [Talend, 2012b]). At the implementation level, data can be stored in a relational DB using a particular structure (e.g. star schema, snowflake schema). Unlike traditional DB structures, DW allows denormalized physical implementation i.e. redundancies (repeated values) are tolerated in DW, which speeds up data access time. Specific techniques such as materialized views, indexes and caching methods can also be implemented to improve performance. *OLAP servers* are used to perform queries to create the mapping between multidimensional concepts of dimensions, facts, measures, aggregation functions and relational DB elements (e.g. tables and columns). Users communicate with these servers via *OLAP clients*. These tools offer a visual representation of query results by means of tabular and graphic displays.
In Table 8.1, we propose a comparison of the DW/OLAP technologies versus spreadsheet tools, which constitute another possible method for data exploration. Table 8.1 shows that DW and spreadsheet tools present similar modes of visualization (e.g. information can be displayed in tables and charts), but spreadsheets are limited in terms of security, access time for large volumes of information and Intranet/Internet data diffusion.
DW technologies have been widely used in the retail sector, but many other potential applications exist. For example, integration of different data sources into a DW can also be performed for epidemiological studies [Bernier et al., 2009; WHO, 2009]. However, DWs of environmental data are still quite rare, even though they could eventually generate a strong interest. [Schulze et al., 2007a; Nilakanta et al., 2008a; Pinet et al., 2010] provide a few examples of DWs for storing and visualizing environmental data. The use of DW technology for the storage of environmental simulation results data is a new domain of application [Mahboubi et al., 2010].

	Region 1			**Region 2**				**…**
	Dpt 1	**Dpt 2**	**Dpt 3**	**Dpt 1**	**Dpt 2**	**Dpt 3**	**Dpt 4**	**…**
Year 1	50000	45000	32000	60000	45000	47000	44000	…
Year 2	50000	47000	30000	58000	45000	47000	43000	…
Year 3	52000	44000	30000	59000	45000	45000	43000	…
…	…	…	…	…	…	…	…	…

Figure 8.2. The amount of water (in m^3) by year and Department.

	Region 1	**Region 2**	**…**
Year 1	127000	136000	…
Year 2	127000	135000	…
Year 3	126000	133000	…
…	…	…	…

Figure 8.3. The amount of water (in m^3) by year and Region.

Tools Criterion	Spreadsheets	DW
Access time for large volumes of data	No data indexing method Limited optimization techniques All data are usually loaded in memory	Optimizations for data queries and insertions (provided by DB); efficient indexing methods Based on a client-server architecture; clients can load and display only a portion of the stored data
Security	Very basic method for user authentications is provided	Users authentication and management of access rights Methods for data backups Integrity constraints can check and guarantee the consistency of information Data encryption is usually possible
Internet/Intranet access	Additional implementation is needed to diffuse data to the Internet/Intranet	Based on an Internet/Intranet client-server architecture
Visualization	Tables with multiple entries Charts	Tables with multiple entries Charts Map production with spatial OLAP technology
Data exploration	Wizard and assistants are often provided to help create tables	OLAP allows for easy modifications to measure and dimension level, easy traversal of level hierarchies, quick swapping between aggregation functions and simple selection of only a subset of the data OLAP server support for complex data queries (e.g. with the MDX language)

Tableau 8.1. A comparison of spreadsheet tools and DW.

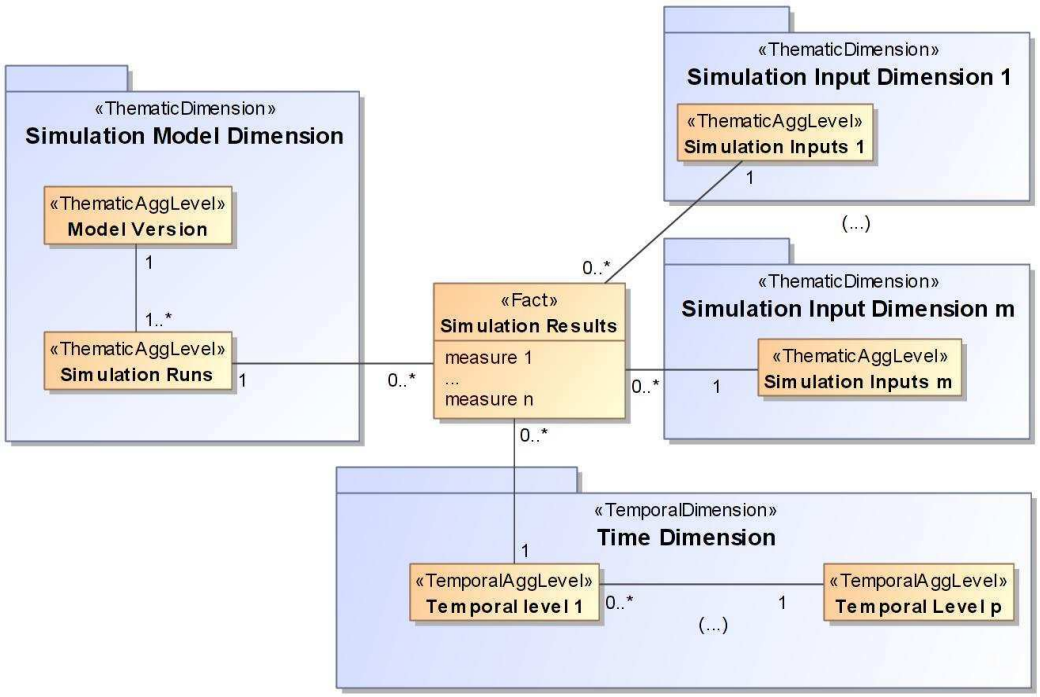


Figure 8.4. A generic multidimensional schema for storage of simulation results [Mahboubi et al., 2010].

8.4 A data warehouse for storing simulation results

[Mahboubi et al., 2010] presented a generic multidimensional schema dedicated to simulation results. This schema defines different concepts that can be found in a DW of simulation results. These concepts are summarized in the template schema of Figure 8.4. The schema contains the following facts and main dimensions:

- **The "Simulation results" facts**: These facts are composed of the measures calculated from simulation runs that can be visualized and analyzed according to different dimensions. In the schema, a simulation result is linked to its simulation runs, its set of inputs and an instant (or a period) of time. These three types of dimensions are described below.

- **The "Simulation Model" dimension**: This dimension is important for the comparison of model runs or model versions. It can be organized into a hierarchy composed of two levels: (1) the simulation runs, which are each associated with (2) a model version.

- **The "Simulation Input" dimensions**: These dimensions represent observed data (e.g. rains, land use) or variables/parameters of the models (e.g. parameters describing the crop growth or the soil characteristics). Several "simulation input" dimensions can be present in the same multidimensional schema (i.e. in the same DW).

- **The Time dimension:** Simulation runs usually produce results at different time steps. This dimension indicates the period of time associated the simulation results. Different levels of this dimension can be applied (e.g. day, month, year, etc.).

Several UML notations are used in the diagram. <<TemporalDimension>> or <<TemporalAggLevel>> indicates that a dimension or a level are related to time (Boulil et

al.). <<ThematicDimension>> and <<ThematicAggLevel>> are used to label other dimensions and levels.

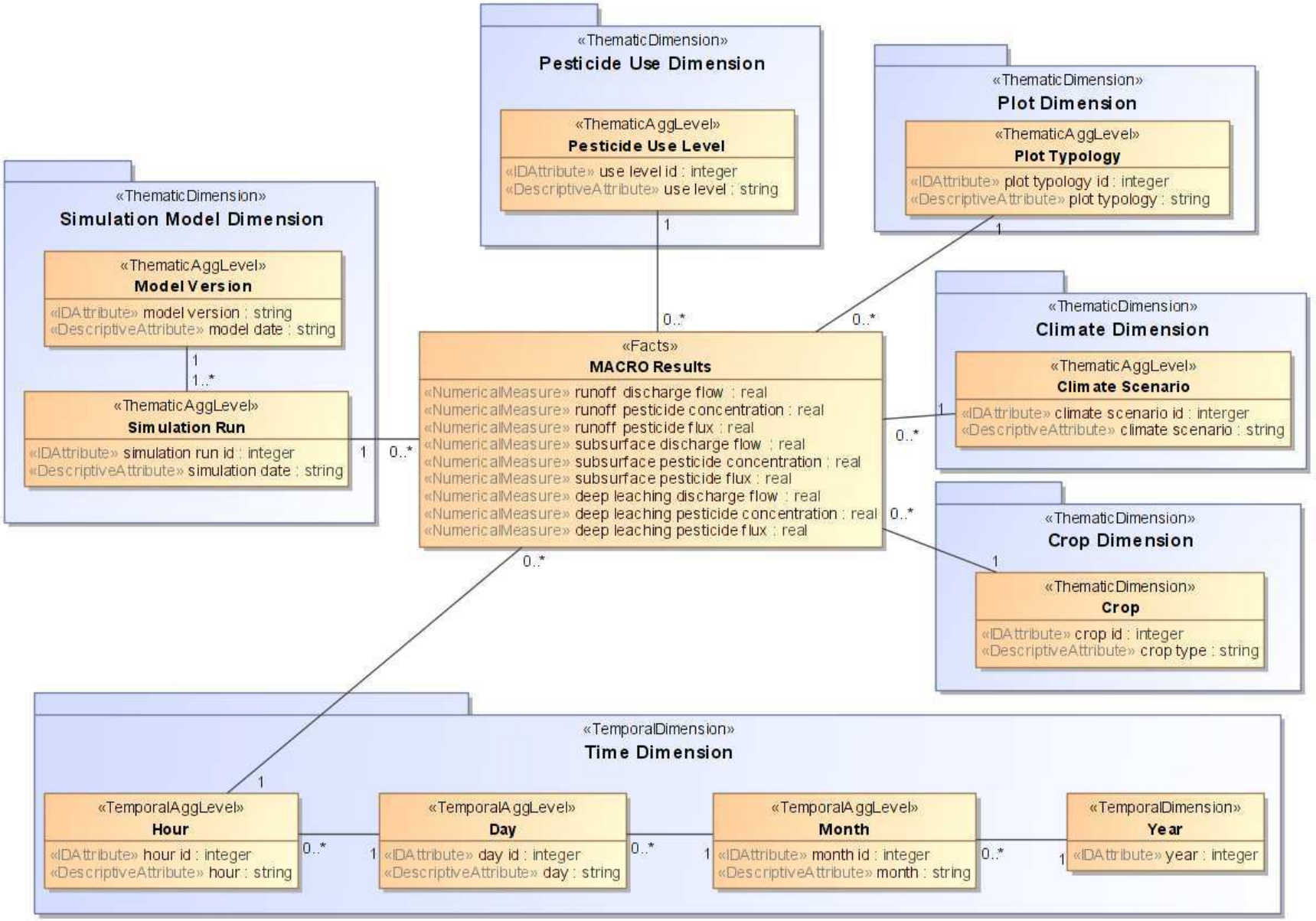


Figure 8.5. The multidimensional schema proposed for the storage of MACRO result data.

8.5 Case study: a Data Warehouse for a Pesticide Transfer Simulation Model

Here, our goal is to show an application of the generic schema to build a DW of simulation results. The example described in the present section will describe how to use the method described in Section 8.3. Our case study concerns storage of data resulting from the MACRO model. We present a DW structure for the storage of temporal data simulated by MACRO.

Note that we only briefly describe the model; detailed information on MACRO can be found elsewhere [Larsbo et Jarvis, 2003]. In this chapter, we focus our attention on the techniques used to store simulation results.

MACRO is a physically based model that simulates water and pesticide transfer using a dimensional approach for both in microporous and macroporous media. Water flow in micro porous parts of soil is described by the classical Richards equation [Richards, 1931] that uses a kinematic wave to describe macroporous flow. Pesticide transfer is represented by advection-diffusion equation, including sorption and degradation processes, in both micro- and macroporous parts of soil. For each time step of a simulation, this model calculates discharge and pesticide concentrations. These values are calculated for the following flow components: surface runoff; deep percolation under the soil profile; and tile drainage.

The schema from Figure 8.5 shows the structure of the DW used to store the MACRO results. The facts are stored in the "MACRO results" class (e.g. the data results produced from the simulation runs). The class is composed of different measures:

- Discharge flow [$L^3.T^{-1}$];
- Concentration of considered pesticide in the flow [$M.L^{-3}$]; and
- Flux of pesticide in the flow [$M.T^{-1}$].

Where M is Mass, L is length, and T is time. Each of these three values can be associated with surface runoff, subsurface lateral fluxes, or deep leaching.

Our multidimensional schema includes four "Simulation Input" dimensions linked to the following facts:

- Plot typology: The MACRO model needs a typology of plots. To avoid simulating each individual catchment field plot, each plot is linked to a "typical" plot, depending on soil type, slope, and whether the soil surface is slaking, thus dramatically reducing the number of simulations needed.

- Climate scenario: Similarly, each climatic year is linked to a typical scenario (i.e. dry, medium, or wet) based on the total amount of annual rainfall. The previous year is also considered as it influences the initial moisture content of soils.

- Crop: This class indicates the considered crop: either wheat, which seeds in late autumn, or corn, which seeds in April or May.

- Level of pesticide use: For both crops, two strategies are considered (re-emergence and post emergence).

The measures are calculated by the MACRO model for each hour, as indicated by the associations between the MACRO results class and the Hour class, which is the smallest unit of the time dimension.
Our schema also includes a simulation model dimension. In our experiment, results produced by two MACRO simulations of the watershed of La Fontaine du Theil in Brittany (France) were stored in the DW. In total, 2 years (totalling 17,520 hours) of data were simulated and stored. Here, only one model version has been used.
We only used freeware software to implement our solution. Our DW has been implemented with PostgreSQL. We chose two other popular tools to display and explore data: Mondrian for the OLAP server and JRubik as the OLAP client.
At the implementation level, we used a star schema (see Figure 8.6). This implementation uses a facts table. Its attributes are measures and foreign keys linked to different dimensions. A fact table contains measure values for the most detailed levels. Measure values are stored by hour, crop, climate scenario, plot typology, pesticide use level and simulation run. This schema also shows dimension tables. This type of table contains denormalized representations of dimension hierarchy levels; in other words, all levels of a dimension are stored in the same table. Denormalization is used very often in DWs. This method produces redundancies (repetitions) of values but vastly improves data access time.

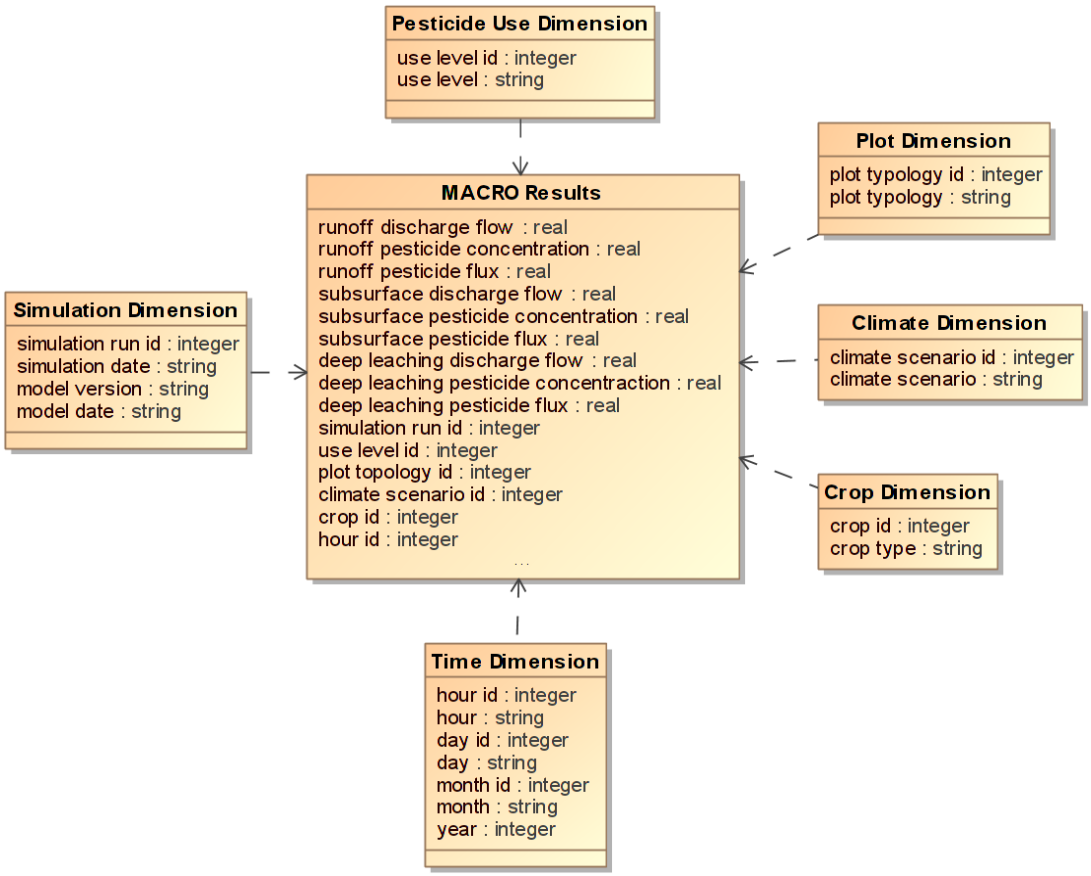
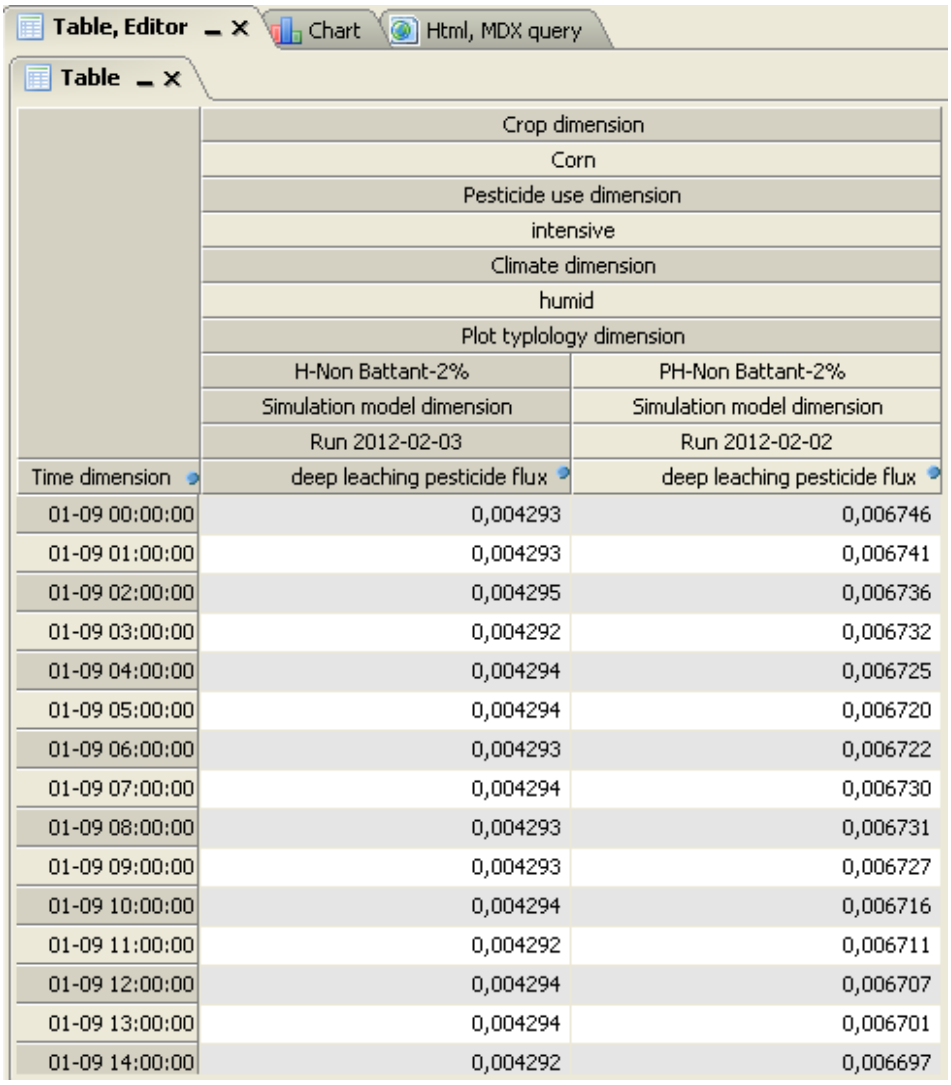


Figure 8.6. A DW implementation model.



Table, Editor | Chart | Html, MDX query

Table

	Crop dimension	
	Corn	
	Pesticide use dimension	
	intensive	
	Climate dimension	
	humid	
	Plot typlology dimension	
	H-Non Battant-2%	PH-Non Battant-2%
	Simulation model dimension	Simulation model dimension
	Run 2012-02-03	Run 2012-02-02
Time dimension	deep leaching pesticide flux	deep leaching pesticide flux
01-09 00:00:00	0,004293	0,006746
01-09 01:00:00	0,004293	0,006741
01-09 02:00:00	0,004295	0,006736
01-09 03:00:00	0,004292	0,006732
01-09 04:00:00	0,004294	0,006725
01-09 05:00:00	0,004294	0,006720
01-09 06:00:00	0,004293	0,006722
01-09 07:00:00	0,004294	0,006730
01-09 08:00:00	0,004293	0,006731
01-09 09:00:00	0,004293	0,006727
01-09 10:00:00	0,004294	0,006716
01-09 11:00:00	0,004292	0,006711
01-09 12:00:00	0,004294	0,006707
01-09 13:00:00	0,004294	0,006701
01-09 14:00:00	0,004292	0,006697

Figure 8.7. A visualization of data on two simulation runs with JRubik.

OLAP tools allow a user to compose a table by choosing the desired dimension levels in the multidimensional schema. Figure 8.7 shows a part of the table resulting from the selection of the measure “deep leaching pesticide flux” and all lowest dimension levels in JRubik. Each type of table entry corresponds to a dimension level presented in Figure 8.5. The chart in Figure 8.8 gives another example of data visualization produced with JRubik. This chart shows the evolution of the measure hour by hour for each simulation run. Numerous other types of graphics can be created.

Now, suppose that the user wants to compare two simulation runs. For this, he/she can select the aggregation function “average” and the level “Model Version” instead of “Simulation Run”. This selection will produce the average deep leaching pesticide flux between the two runs, delimited by hour (see Figure 8.9).

Figure 8.8. An exampleof a chart produced by JRubik that compares deep leaching pesticide flux from two simulation runs.

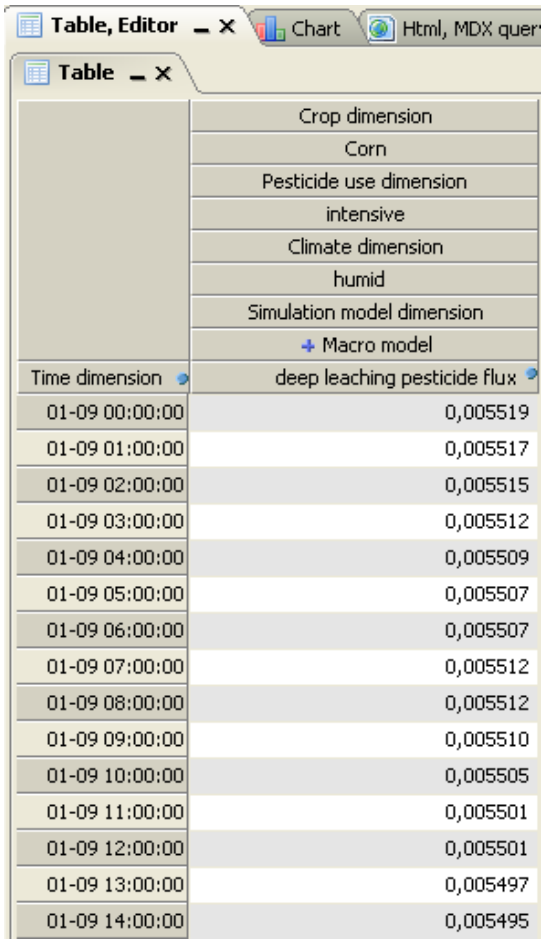


Figure 8.9. The average measure of two simulation runs, as displayed within JRubik.

In the system we implemented, we had only one type of pesticide. If we want to have a DW storing simulation results produced by different category of pesticides, we need to add a pesticide active matter dimension; see Figure 8.10 – the other dimensions remain unchanged.

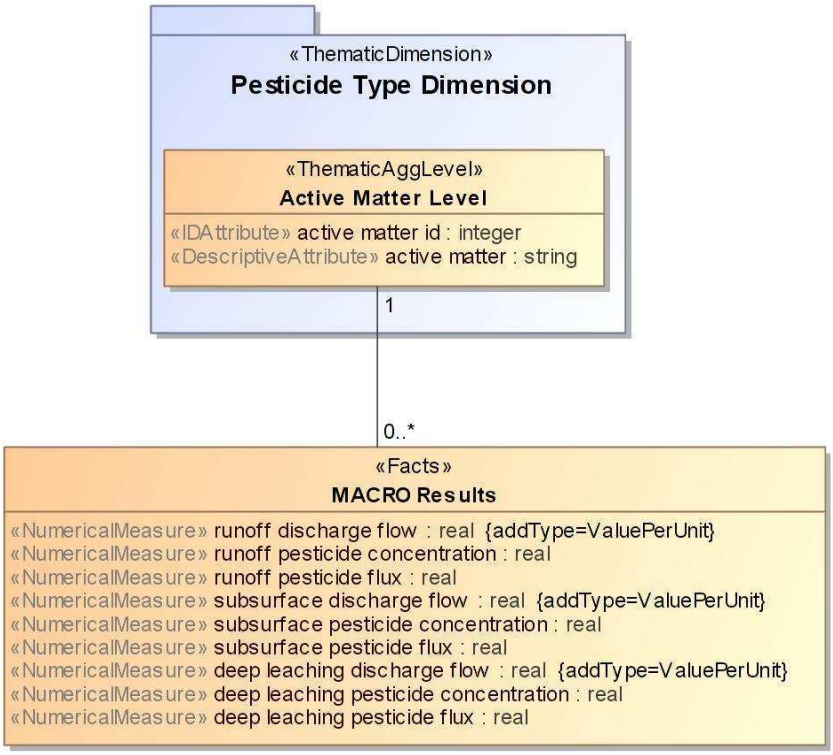
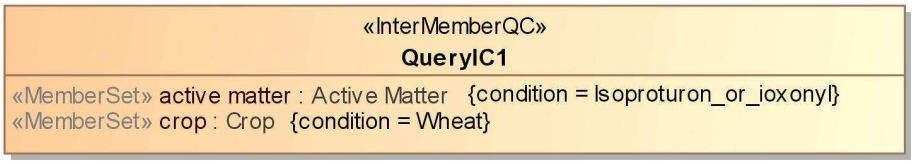


Figure 8.10. The active matter dimension.

Integrity constraints can be also specified in DW [Boulil et al., 2012c]. These constraints are rules implemented and checked in the DW in order to ensure the consistency and the good quality of the stored data. Here, we introduce some examples of constraints formalized with the UML-based formalism proposed in [Boulil et al., 2012c]. In Figure 8.10, the aggregation constraint {addType=ValuePerType} indicates that the sum function cannot be used on the measure. Data constraints can be also modeled. The following constraint expressed using the Object Constraint Language (OCL) [Boulil et al., 2012c] states that the three pesticide active matters “atrazine”, “dimethenamid” and “sulcotrione” cannot be used with the crop “maize”:

```
context MACRO_Results inv:
not (
(self.Pesticide_Type.active_matter = “atrazine” or
 self.Pesticide_Type.active_matter = “dimethenamid” or
 self.Pesticide_Type.active_matter = “sulcotrione”) and
self.Crop.crop_type = “maize”
)
```

This type of constraints can be checked on the data (during or after the DW loading). Alternatively, it is also possible to check constraints during DW explorations (i.e. during the use of OLAP tools). For example, Figure 8.11 shows an exploration constraint (i.e. a query constraint [Boulil et al., 2012c]) used to represent an “invalid” combination of level instances. This constraint formalizes that the two pesticide active matters “isoproturon” and “ioxonyl” cannot be used with the crop “wheat”. The QueryIC1 class shows instance levels that the user should not combine within the OLAP tool. Two OCL constraints provide details on these instance levels. [Boulil et al., 2012c] provides information on how query constraints can be checked during the data exploration.




```
context Active Matter inv Isoproturon_or_ioxonyl:
self.active_matter = “isoproturon” or self.active_matter = “ioxonyl”
context Crop inv Wheat:
self.crop_type = “wheat”
```

Figure 8.11. An example of query constraint.

8.6 Conclusion and future work

The volumes of simulation result data only continues to increase, and model users now need a technology specifically dedicated to store and explore the vast quantities of information produced by models. As demonstrated in this chapter, DW/OLAP technologies can help practitioners to easily analyse and explore simulation data.

Here, we summarize the methodology that we followed to implement our DW. First, we adapted the generic multidimensional schema proposed in [Mahboubi et al., 2010] to the target model. We chose measures and input dimensions that were suitable for the MACRO environmental model. The temporal granularity of MACRO provides the lowest level of the temporal dimension (i.e. one hour). We also added other temporal levels to allow users to aggregate measures per month or year. The DB schema was created with PostgreSQL. We chose a star schema (i.e. a denormalized implementation) to ensure the best data access performance. Then, results data were loaded into the DB from text files produced by MACRO runs. Mondrian and JRubik allow users to explore DW data very easily by choosing the desired levels of analysis. Researchers can benefit from the user-friendly JRubik graphic user interface to change the dimension level, to traverse level hierarchies, to change the aggregation functions, or to visualize a subset of results data. Our architecture also allows users to export a selection of data into other formats.

The next steps of our work will be to enrich and improve the generic multidimensional schema by experimenting with other environmental models and by considering feedback from users. We also plan to study the possibilities offered by spatial DW/OLAP [Malinowski et Zimanyi, 2008; Bédard et al., 2009; Bimonte, 2010; Sboui et al., 2010]. These recent tools are dedicated to display information stored in DWs by means of maps and can be useful for visualizing geo-referenced environmental models. The extraction, transformation, and loading processes are also important aspects of the architecture, and we also plan to propose a tool to facilitate these steps. To do so, we have started to extend SimExplorer [Chuffart et al., 2010], a software application for managing simulation runs, to help users create a DW and load model output.

9 Chapitre 9 : Definition and analysis of new agricultural farm energetic indicators using spatial OLAP

9.1 Preamble

Agricultural energy consumption is an important environmental and social issue. Several diagnoses have been proposed to define indicators for analyzing energy consumption at large scale of agricultural farm activities (year, farm, family of production, etc.). However, to define ad-hoc environmental energetic policies to better monitor and control energy consumption, new indicators at a most detailed scale are needed. Moreover, by defining detailed scale indicators, large quantities of geo-referenced data need to be collected to feed these energetic diagnoses. This huge volume of data represents another important limitation of systems that implement these diagnoses because they are usually based on classical data storage systems (such as spreadsheet tools and Database Management Systems). These systems do not allow for interactive analysis at different granularities/scales of huge volumes of data and do not provide any cartographic representation. By contrast, Spatial OLAP (SOLAP) and spatial data warehouse (SDW) systems allow for the analysis of huge volumes of geo-referenced data by providing aggregated numerical values visualized by means of interactive tabular, graphical and cartographic displays. Thus, in this chapter, we (i) propose new appropriate indicators to analyze agricultural farm energy performance at a detailed scale and (ii) show how SDW and SOLAP technologies can be used to represent, store and analyze these indicators by simultaneously producing expressive reports.

9.2 Introduction

Agriculture energy consumption depends on the method of production used. Direct agricultural energy consumption was estimated to be 28 Mtpe of a total consumption of 1142 Mtpe, which was 2.5% of the energy directly consumed by the EU25 in 2004 [SOLAGRO, 2007]; 55% of this energy was the result of fuel consumption. With the planned reduction in oil and rising oil prices, agriculture must reduce energy consumption to improve its economic development and decrease its environmental impact. Awareness of the importance of preserving non-renewable energy resources is a certainty, as evidenced by the energy development policies adopted in recent years by different governments (Energy Policy of 2005, the Grenelle Environment, etc.). Applied to the agricultural context, this reality requires a better assessment of the energy balance of farms in terms of energy performance. At present, many diagnoses that define a set of indicators exist to assess the energy performance of agricultural farms [Pradel et Boffety, 2012]. These diagnoses, which are specially adapted to a comprehensive assessment at the farm scale, are not necessarily relevant to evaluate the energy performance at a detailed scale (plot, technical operation, etc.). Moreover, by defining detailed scale indicators, large quantities of geo-referenced data have been collected to feed these energy diagnoses. This huge volume of data represents another important limitation of systems that implement these diagnoses because they are usually based on classical data storage systems, such as spreadsheet tools, which do not allow interactive analysis at different granularities/scales of huge volumes of geographic data. Moreover, diagnosis systems do not

provide any cartographic visualization of energy indicators by limiting important analysis capabilities associated with geographic data [MacEachren et al., 2004].
By contrast, Geo-Business Intelligence technologies such as Spatial Data Warehouses (SDW) and Spatial OLAP (SOLAP) systems are widely recognized as efficient tools for the on-line analysis of huge volumes of geo-referenced datasets. SOLAP systems have been defined by Y. Bédard as "*Visual platforms built especially to support rapid and easy spatiotemporal analysis and exploration of data following a multidimensional approach comprised of aggregation levels available in cartographic displays as well as in tabular and diagram displays*" [Bédard et al., 2001]. In the last years, SOLAP technology has been successfully used in several application domains such as geo-marketing, health monitoring, and agriculture [Nilakanta et al., 2008b].
Thus, in this chapter we propose (i) *some new appropriate indicators to analyze agricultural farm energy performance at a most detailed scale* and (ii) *show how SDW and SOLAP technologies can be used to represent, store and analyze these indicators by simultaneously producing cartographic and tabular reports*.
The case study of this work is a result of the EnergeTIC project. This project aims to use a scientific and technical solution to assess the energetic performance of farms through the use of ICT (Information and Communication Technologies) on the finest scale. Installed on agricultural equipment, the identified technological solutions (low-cost sensors, RFID, etc. provide reliable and continuous data to calculate energetic performance indicators at the finest scale (field, technical operation, etc.).
The chapter is organized as follows. Section 9.3 presents related work concerning agricultural energy diagnoses, SDW and SOLAP. New indicators for farm energy consumption at a detailed scale are presented in Section 9.4. Section 9.5 describes how the SDW system is used to represent these new indicators and their implementation in the SOLAP tool JMap SOLAP. Section 9.6 shows the definition of some Integrity Constraints (IC) to verify the consistency of warehoused data, aggregations and SOLAP queries. Conclusions and future work are presented in Section 9.7.

9.3 Related work

In this section, we recall the purpose of Spatial OLAP systems (Section 9.3.2), and provide a survey of agricultural farm energy consumption diagnoses (Section 9.3.1).

9.3.1 Agricultural farm energy consumption diagnosis and related indicators

In the last few years, several studies dealing with direct and indirect energy consumption at the farm scale have been proposed [Pervanchon et al., 2002; Bailey et al., 2003; Vilain, 2008]. These works aim to create complete diagnosis tools and/or energy performance assessment methods. The term "diagnosis" will be used in this chapter to refer to both "tool" and "method". These diagnoses, based on the energy consumption of farms and the energy value of agricultural products, allow for the calculation of energy balance and the assessment of the energy efficiency of farms.
In particular, the energy diagnosis aims to:

- quantify energy consumption per processes to identify possible improvements by acting either on the production system, practices or equipment
- compare energy performance of livestock and crop farming, and
- establish reference values for the above production.

These diagnoses are based on the calculation of energy indicators, which are variables that provide information on less accessible data. These indicators are references used to make decisions [Gras et al., 1989], as they allow for the understanding of a complex system to assist in the realization of objectives [Mitchell et al., 1995].

We performed a survey of farm energy performance diagnoses used in France (Table 9.1). These diagnoses can be grouped into 6 categories (some examples of indicators are also provided):

- *Global agro-environmental diagnoses*: these consider the farm as a whole and assess the global environmental impact of the farm (nitrogen excess, energy consumption, etc.) (*Sum of the quantities of direct and indirect energy used by the farm, expressed in MJ per year*)
- *Field agro-environmental diagnoses*: these assess the environmental performance of the farm at the field scale (*Sum of the quantities of direct and indirect energy used by the farm, expressed in MJ per hectare per year*)
- *Sustainability diagnoses*: these assess farm sustainability performance through environmental, social and economic sustainability issues
- *Energetic diagnoses:* these allow an energy balance at the farm scale by quantifying energy inputs and outputs (*Energy efficiency: ∑(energy produced) / ∑(direct and indirect energy consumed)*)
- *Prediagnoses or autodiagnoses*: these assess the energy consumption of farm equipment and give an idea of the main possible improvements (based on the farmer's qualitative assumptions)
- *Life cycle assessment*: these methods assess the environmental impact on a system by inventory pollutant emissions, raw materials and energy during its whole life cycle (*Energy consumption based on the functional unit used, either the hectare or milk liter*)

The main inputs of these diagnoses are direct and indirect energy flows and energy consuming operations (energy used for irrigation, for example). Direct energy flows are electricity, fuels and gases (propane, butane and city gas). Indirect energies are energies used to produce outputs such as fertilizers, crop protection products, etc.

The most common indicators are simple or aggregated indicators. For example, a simple indicator is the fuel quantity and two aggregated indicators are the energy balance and energy efficiency of farms. The calculation of these quantities is based on direct and indirect energy data collected from accountancy, paper documents or direct communication between farmers (equipment and building characteristics, produced quantities, etc.). Indeed, to the best of our knowledge, there is actually no technical solution implementation to collect farm energy data at this fine scale

Information obtained from these diagnoses allows for a good analysis of the global farm energy performance by identifying the most energy-consuming activities. As shown in Table 9.1, indicator calculation is most often limited to the global scale (i.e. farm scale) due to the lack of reliable data. An analysis at a most detailed scale (field, production activity or operation) will require more precise data acquisition systems.

This means that a set of new indicators at a most detailed scale is needed. Moreover, because collecting data at a fine scale produces a huge volume of data, classical systems used to analyze these indicators (e.g. spreadsheet tools and database management systems) are not sufficient in this context because they do not allow visual/cartographic interactive analysis at different granularities/scales of huge data volumes.

Diagnosis	Scale	Category
PLANETE	Farm	Energy diagnoses
IDEA	Farm	Sustainability diagnoses
DIALECTE	Farm	Global agro-environmental diagnoses
DIALOGUE	Field Farm	Global/field agro-environmental diagnoses
INDIGO	Field Farm	Field agro-environmental diagnoses
BILAN CARBONE	Economic activity	Life cycle assessment
AUTO DIAGNOSTIC ÉNERGÉTIQUE DES BATIMENTS D'ÉLEVAGE	Farm	Auto diagnosis
GESTIM	Farm Operation	Energy diagnoses
AUDIT ÉNERGÉTIQUE EN PRODUCTION LAITIÈRE	Farm	Auto diagnosis
KUL	Farm	Global agro-environmental diagnoses
SALCA	Farm	Life cycle assessment

Tableau 9.1. Farm energy consumption diagnoses.

9.3.2Spatial OLAP

SOLAP tools integrate OLAP and Geographic Information Systems (GIS) advanced functionalities to explore, visualize and analyze multidimensional geo-referenced data by means of tabular, graphic and interactive cartographic visualization [Bédard et al., 2001]. In this way, spatial decision makers can visually detect unknown patterns and spatial phenomena and verify and/or formulate hypotheses. An example of the use of SDW to analyze pollution by French departments is presented in [Bimonte et Pinet, 2010]. Here dimensions include (i) the temporal dimension organized into a classical calendar hierarchy (day, month, and year), (ii) a dimension representing pollutants and (iii) a spatial dimension representing the French administrative organization into departments and regions. The measure is the pollution value that is aggregated using the average. Using this SDW, users can answer questions like: "*What was the average pollution value per pollutant and department in 2000?*" or "*What is the average pollution value per month per region for inorganic pollutants?*".

A typical SOLAP architecture is composed of a Spatial DBMS to store warehoused (spatial) data; a SOLAP server, which implements OLAP operators; and a SOLAP client, which combines and synchronizes tabular, graphical and interactive maps to visualize and trigger SOLAP queries. An example of SOLAP visualization is shown in Figure 9.1 using the environmental SOLAP application previously described.

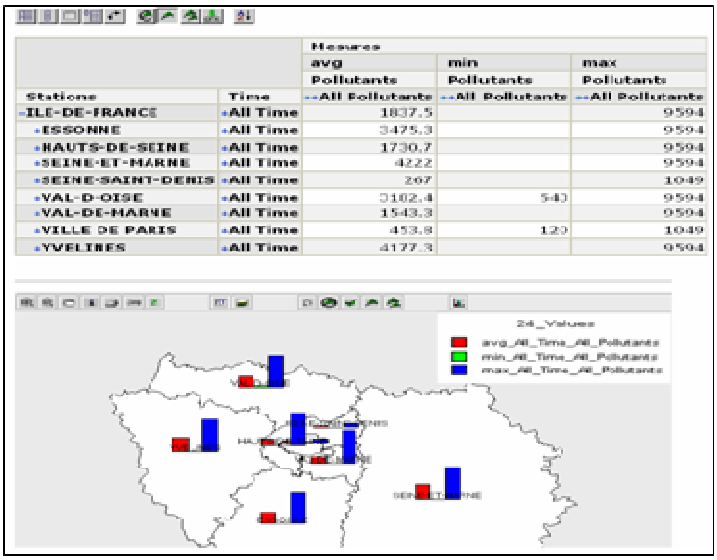


Figure 9.1. A SOLAP application concerning the monitoring of pollutants [Bimonte et Pinet, 2010].

There are various SOLAP application domains, such as health, urban monitoring and marketing. SOLAP has also been applied to the agricultural context [Thornsbury et al., 2003]. The monitoring of pollutants has been investigated in [Abdullah et al., 2004; Bimonte et Pinet, 2010]. Economic analysis of agriculture productions is presented in [Clements et al., 2002; Chaudhary et al., 2004; Schulze et al., 2007b; Nilakanta et al., 2008b; Rai et al., 2008]. These works highlight the relevance of using spatial decision support tools such as SDW and SOLAP in the agricultural context, and include evidence for particular issues concerning design of dimensions (complex hierarchies), facts (measures at different granularities, measure types, etc.) and architectural solutions (integration of GIS and Spatial Data Mining, etc.). However, to the best of our knowledge no works have studied the use of SDW and SOLAP systems to produce, aggregate and visualize energy consumption indicators of agricultural farms.

9.4 New energy indicators to assess farm performance at a detailed scale

The assessment of energy performance of farms at a detailed scale aims to establish what types of indicators are needed and the related assessment scale. We choose to design two types of indicators (Table 9.2):

- Indicators based on invoices, direct communication between farmers or administrative documents to assess indirect energy consumption (foodstuffs, fertilizers, pesticides, etc.).
- Indicators based on direct measurements collected by means of technological solutions installed on mobile equipment to assess direct energy consumption.
- The indicators were designed according to the most relevant analysis scale:
- Spatial scale: the hectare is the spatial reference unit used to express the technical-economic results on a farm. We will use it to express all the direct energy flows, such as fuel or gas for mobile equipment, or to express indirect energy flows such as inputs in crop management (e.g. fertilizers, pesticides).
- Temporal scale: the hour is the temporal reference unit for farmers. This unit is frequently used to express fuel consumption for mobile equipment and electric/gas consumption for electric equipment.

- Production scale: all of the energy flows involved in farm activities (crop or cattle management) can also be expressed at the production scale. The results will be expressed with the common production unit used by the farmer depending on the type of farm production (e.g. tons of dry matter, liters of milk, etc.).
- These indicators were calculated for the three main farm activities:
- Crop management activities, which include all the technical operations on crops (sowing, plowing, fertilization, harvest, etc.);
- Cattle management activities, which include all the technical operations on cattle, such as care, feeding and milk/meat production (milking, slaughtering, etc.);
- General activities, which cannot be allocated to cattle or crop management (cleaning, logistics, transport, etc.).

These indicators can be aggregated at the global scale using the sum.

An example indicator is the fuel consumption per plot, technical operation, year and production. An example of the value of this indicator is “140 liters of fuel used for the parcel ‘13 pal’ of the farm of Montoldre in 2010 during the plowing operation for the production of wheat”. An example of the aggregated indicator obtained from the previous indicator by aggregation on the spatial scale is the fuel consumption per department, technical operation, year and production. The aggregates are calculated by summing all the plots' values belonging to the same department (i.e. Allier).

9.5 Spatial OLAP analysis of new indicators

In this section, we propose a system to implement (represent, store and analyze) the indicators defined in Section 9.4. The main idea is to represent detailed indicators defined in Section 9.4 as measures of a spatial data warehouse and to represent their inputs (energy, time, spatial scale, technical operation and production) as dimensions. Thus, aggregated indicators correspond to aggregated measures.
In this way, the spatial data warehouse can provide answers to the questions of two types of users: farm managers and life-cycle assessment (LCA) practitioners. From the perspective of farm managers, useful SOLAP (i.e. (aggregated) indicators) queries are the "*number of interventions by culture*", etc. For LCA practitioners, questions arise in terms of life cycle assessment inventories. For example, it may be interesting to know the "*fuel to weed a plot of wheat*" and the "*average consumption of fuel to weed the entire farm per year*".

Energy flow			Indicator	Example	Indicator objective
Direct energy	Spatial scale	Fuels	FU[47]/crop ha/plot/technical operation	Energy consumption to weed 1 ha of wheat for plot X	Assess the most energy consuming operations for each technical operation for each crop Compare the energy consumption of the same operation for different crops
		Fuels, gas, renewable energies	FU/m²/type of building	Energy consumption to heat 1 m^2 of the milking parlor using electricity	Assess the most energy consuming buildings
	Temporal scale	Fuels, butane, propane gas	FU/hour/technical operation/equipment/crop	Energy consumption for 1 hour of weeding wheat with the XY equipment	Identify the most energy consuming equipment for each technical operation for each crop system
Indirect energy	Spatial scale	All inputs	FU/crop ha/technical operation	Energy consumption to weed 1 ha of wheat	Assess the most energy consuming operations for each technical operation for each crop
		Cleaning products	FU/m^2/building	Energy consumption to clean 1 m^2 of the milking parlor	Assess the most energy consuming buildings
	Temporal scale	All inputs	FU/year/production cycle/technical operation/crop	Annual (or production cycle) consumption to weed wheat	Identify the most energy consuming operations for each technical operation

Tableau 9.2. New indicators to assess farm performance at a detailed scale.

Using a SOLAP system to represent indicators overcomes the limits of existing diagnosis systems with respect to two aspects: it allows the interactive analysis of huge volumes of (aggregated) indicator values and it provides a cartographic representation.

Figure 9.2 presents the conceptual schema of the spatial data warehouse we propose, using the multidimensional conceptual model based on the UML presented in [Boulil et al., 2011]. The conceptual model presents stereotypes for each spatio-multidimensional element, such as "Fact" for the facts, "SpatialAggLevel" for spatial dimension levels (i.e. levels having a geometric attribute "LevelGeometry" that represents the locations of their members), "AggRelationship" for hierarchical associations between dimension levels, "DimRelationship "for associations between levels and facts, etc.

The SDW presents several measures that represent the previous indicators defined in Section 9.4. In particular, the measures are: the area worked (*surface_w*), the number of animals (*animaux_nb*), the amount of product (input represented with "*intrant*" and output denoted with "*extrant*") used during work or no work (denoted with "w" and "nw", respectively), the duration in hours (*duree_w* and *dure_nw*) and the distance traveled (*distance_parcourue_w* and *distance_parcourue_nw*). The measures are aggregated using the sum on all dimensions.

The difference between work and no work (*w* and *nw*) is used to quantify energy consumption both directly related to and not attributable to work. For example, the amount of fuel used to plow a field is associated with work, while the amount consumed during the turn of the machine, with the plow raised, is not considered directly related to work.

The different dimensions are:

[47] FU = Evaluated Flow unit (litre, kWh, kg, etc.)

- Campaigns (*Campagnes*): production cycles expressed in years (e.g. wheat produced in 2009).
- Time (*Temps*): classical temporal dimension, in which days are grouped by month and year.
- Products (*Produits*): the input and output products (intrant and extrant). Products are grouped recursively into larger classes of products.
- Operators (*Operateurs*): people who perform the operation.
- Equipments (*Equipements*): machines and tools used.
- Location (*Localisations*): the spatial dimension that groups plots (*Parcelle*) by farm (*Exploitation*), department (*Departement*) and region (*Region*).
- Productions: type of production (e.g. wheat)
- Technical Operations (*OperationsTechniques*): the technical operations performed, which are grouped by functions.

Using this spatio-multidimensional model, it is possible to represent and aggregate all indicators presented in section 3 such as the fuel consumption per plot, technical operation, year and production.
For the implementation of our application we have chosen the SOLAP tool JMap SOLAP because it is one of the best business solutions that incorporates all existing OLAP and GIS functionalities. JMap SOLAP is based on a three-tiers Relational SOLAP architecture as described in Section 9.3.2. The SOLAP server allows the definition of the elements of the spatial data warehouse and the various data access policies for different users through a simple visual interface. The SOLAP web client integrates intelligent mapping concepts that support the automatic creation of thematic maps, while ensuring compliance with the semiotics rules (colours, symbols, frames, etc.). The client provides simple and multi-maps with synchronized diagrams and tabular displays to visualize SOLAP queries results. The client also implements the SOLAP operators described in Section 9.3.2.

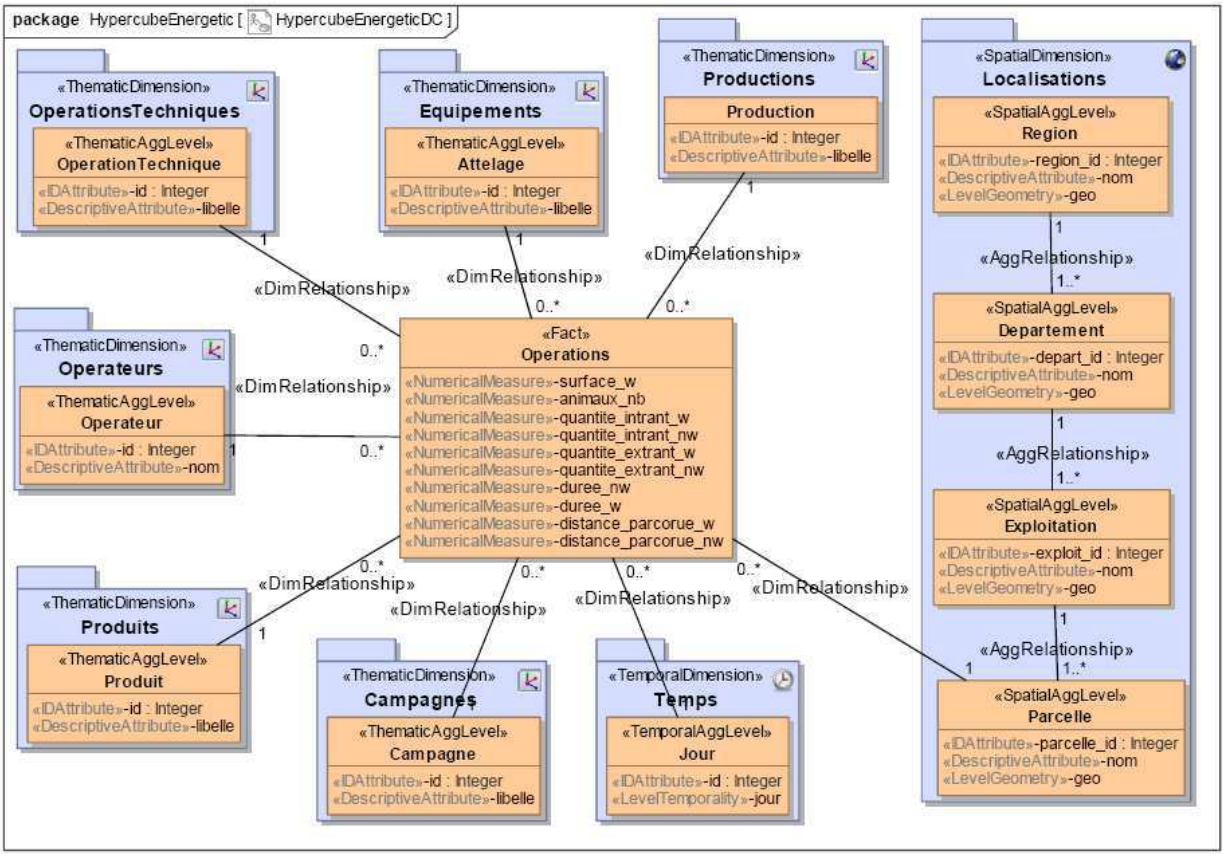


Figure 9.2. An SDW conceptual model for indicators of Table 9.2.

In the following figure we show how to formulate and visualize spatio-multidimensional queries to obtain, for example, the aggregated indicator "the amount of fuel used for cultivating one hectare of wheat on the Montoldre farm". Once the user has selected interesting items (e.g. dimension elements) for the indicator calculation, the table in Figure 9.3 is displayed showing the quantities of fuel consumed per hectare over the entire Montoldre farm. The result can also be displayed on a map (Figure 9.3).

The decision maker can apply a Spatial Drill-down operation directly on the map to obtain the energy consumption per plot. The result, shown in Figure 9.4, indicates that "PIQ 1" and "2PIQ" are the parcels that have consumed the most energy.

It is also possible to determine the most expensive technical operations in terms of fuel. Using the operator to drill-down on the technical operations dimension of the previous table, a new table is shown (Figure 9.5), which gives more details on energy consumption by plot.

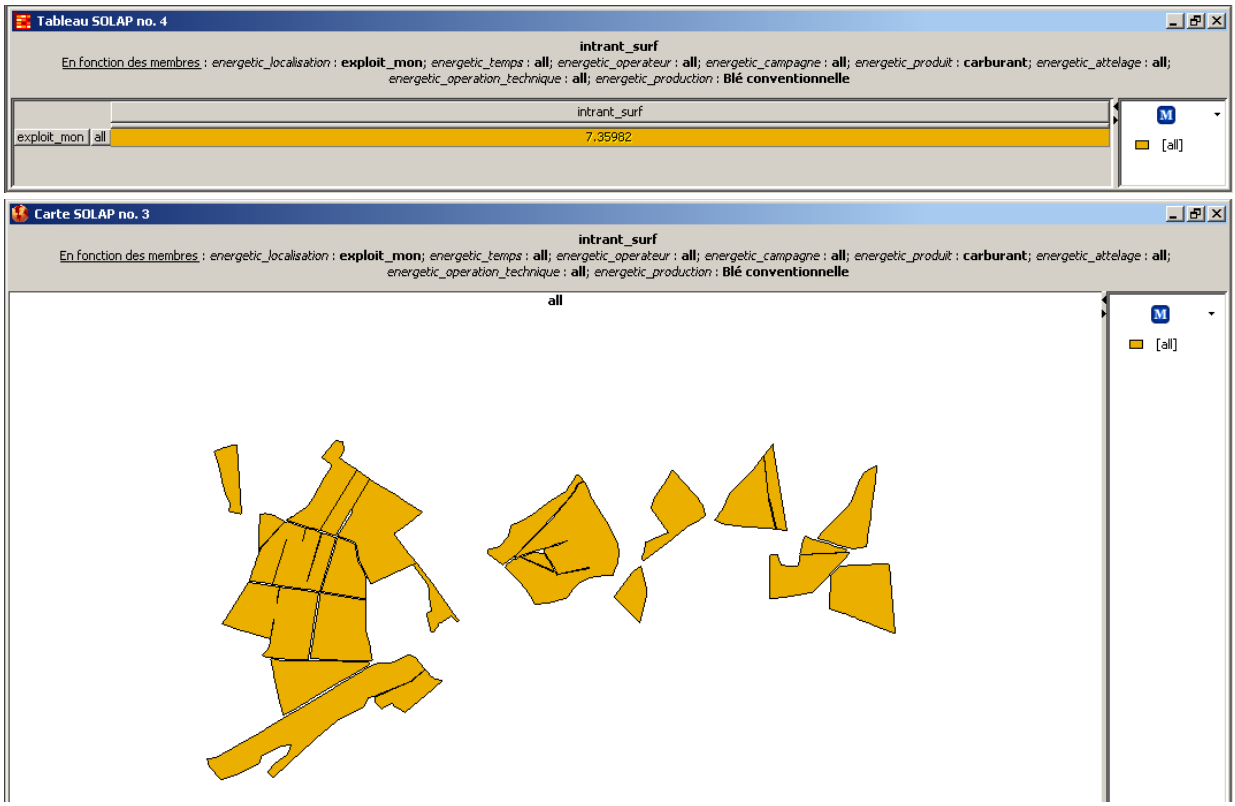


Figure 9.3. Visualization of the aggregated indicator "the amount of fuel used for cultivating one hectare of wheat on the Montoldre farm".

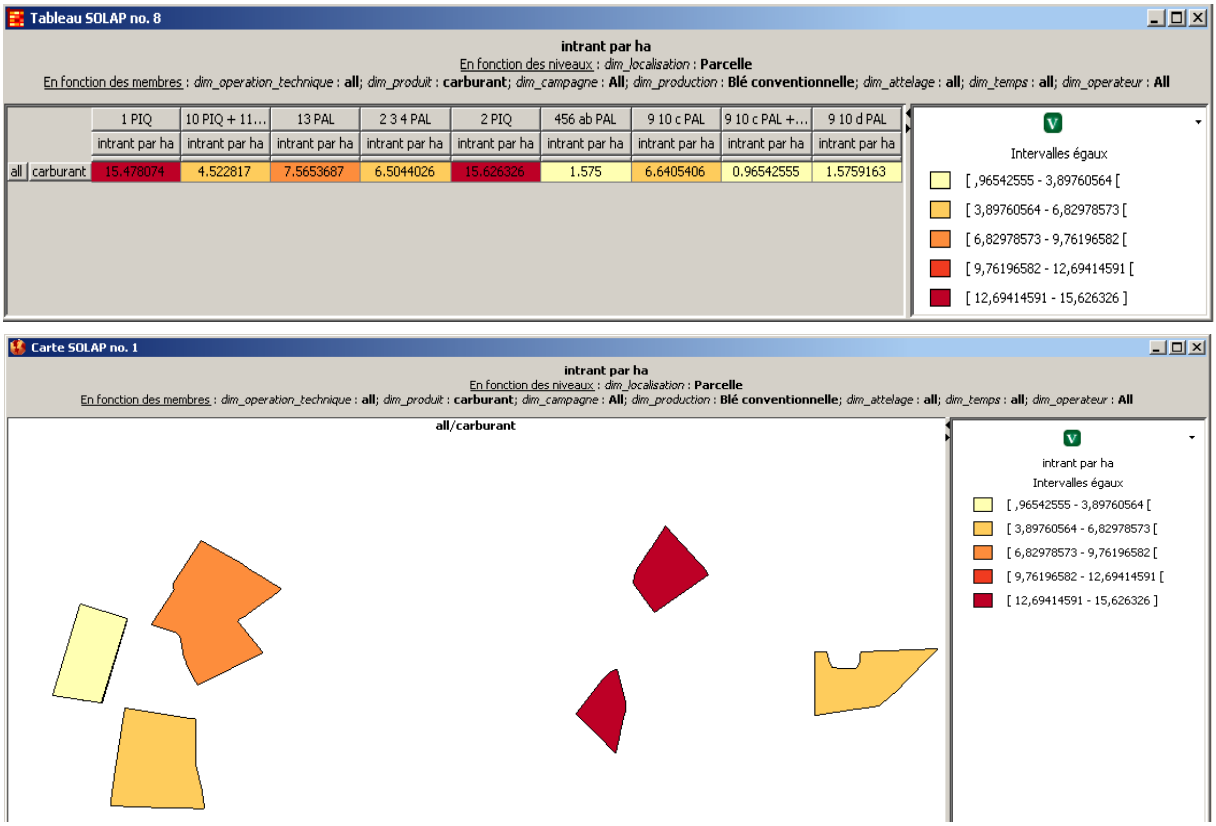


Figure 9.4. Visualization of the indicator "the amount of fuel used for cultivating one hectare of wheat for each plot of the Montoldre farm".

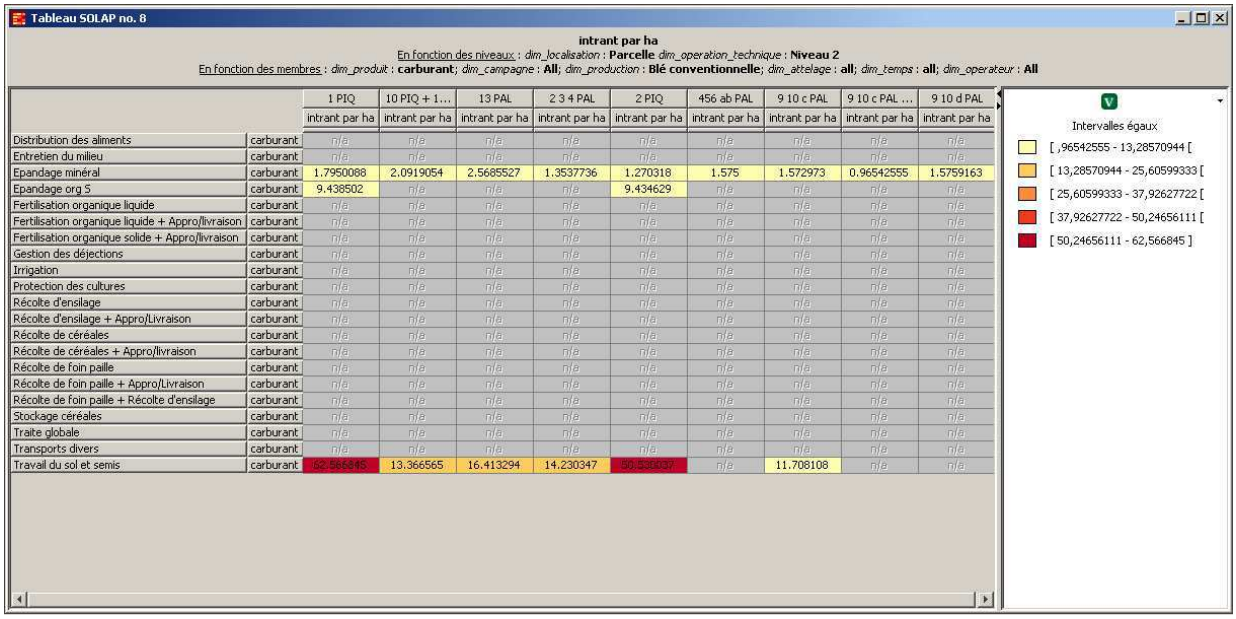


Figure 9.5. Visualization of the indicator "the amount of fuel used for cultivating one hectare of wheat for each technical operation for each plot of the Montoldre farm".

In the same way, the example of the indicator described in Section 9.4, "fuel consumption per plot, technical operation, year and production", can be easily visualized in our system by using the SOLAP query, the result of which is shown in Figure 9.6.

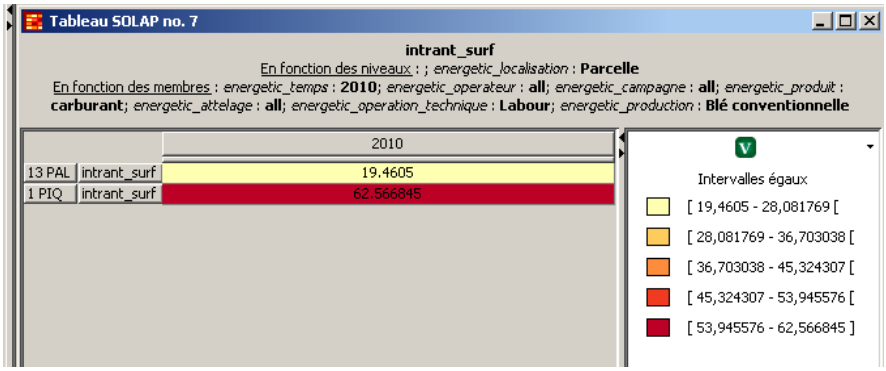


Figure 9.6. Visualization of the indicator "fuel consumption per plot, technical operation, year and production".

9.6 Integrity Constraints

As stated in [Boulil et al., 2011], in addition to the precision and completeness data quality elements, the quality of SOLAP analysis depends on three types of consistencies: (i) the consistency of warehoused data (absence of contradictions and errors in the stored data set), (ii) the consistency of aggregations of measures (inconsistent and meaningless aggregations of measures such as the sum of temperature values must be avoided) and the consistency of SOLAP queries (inconsistent queries such as the total of sales amounts in USSR during the year of 2011 must be avoided). These three types of inconstancies can be managed using three different categories of Integrity Constraints (IC): Data IC, Aggregation IC and Query IC respectively.

In this section we present an application of our proposals [Boulil et al., 2011] for the conceptual specification and the automatic implementation of these IC categories in the context of the EnergeTIC project. Note that, in order to implement these constraints, we use a three-tier SOLAP architecture composed of (i) the Oracle Spatial DBMS, (ii) the OLAP server Mondrian, and (ii) the SOLAP client JRubik instead of the JMap SOLAP tool which does not support the MDX (MultiDimensional eXpressions) language. This standard language for querying OLAP and multidimensional databases is used in our proposal to implement query constraints.

The first IC we present is a Data IC and in particular an "Inter-level IC" since it involves two aggregation levels (Parcelle and Exlpoitation). This IC states that the geometry of each parcel has to be spatially included into the geometry of its farm. It is specified using the Spatial OCL object constraint language on the UML model of Figure 9.2 as follows:

```
context Parcelle inv DataIC1:
self.geo.isInside(self.exploitation.geo) or
self.geo.coveredBy(self.exploitation.geo)
```

In order to automatically implement the data IC and the SDW data model in the Oracle Spatial DBMS, we use the code generator Spatial OCL2SQL. This tool transforms each Spatial OCL constraint into a SQL query, a SQL view and a set of triggers. The query and the view select the data (tuples of the table corresponding to the contextual class of the Spatial OCL IC) that do not satisfy the constraint. For example, the SQL view generated for the previous Spatial OCL IC is the following:

```
create or replace view DataIC1View as
(select * from PARCELLE SELF where not (
 (MDSYS.SDO_RELATE(
 (select GEO from EXPLOITATION where PK16 in (select
 RELATED_EXPLOITATION_PK16 from OV_PARCELLE where PK5 = SELF.PK5)),
 SELF.GEO,
 'mask=CONTAINS querytype=WINDOW') = 'TRUE') OR
 (MDSYS.SDO_RELATE(
 (select GEO from EXPLOITATION where PK16 in (select
 RELATED_EXPLOITATION_PK16 from OV_PARCELLE where PK5 = SELF.PK5)),
 SELF.GEO,
 'mask=COVERS querytype=WINDOW') = 'TRUE')
));
```

This SQL view selects the tuples of the table PARCELLE that do not satisfy the Data IC condition.

The triggers (one trigger for each table involved in the IC) verify if the SQL view is not empty when updating the corresponding table in which case they return an error message. For example, the following trigger defined on the table EXPLOITATION verifies if the SQL view DataIC1View is not empty when updating the table EXPLOITATION.

```
create or replace trigger tr_DataIC1View_on_EXPLOITATION
after insert or update or delete on EXPLOITATION
declare tmp number;
begin
 select NVL(COUNT(*),0) into tmp from DataIC1View;
 if (tmp > 0) then
 raise_application_error(-20000,'integrity violation at DataIC1View');
 end if;
end;
```

The second IC concerns the aggregation and in particular the conformity between the measure type and the aggregate function type.
The Aggregation rule R4 in the following figure (Figure 9.7) is not valid because it defines a spatial aggregation (GeometricUnion) on a numeric measure (duree_w). The OCL constraint implemented in the MagicDraw profile to prevent this kind of inconformity or inconsistency is the following:

```
context AggRule inv AggregatorAndMeasureTypeConformity:
not(baseIndicator.aggregatedAttribute.OclIsUndefined()) implies
aggregator.applicableTo->exists(dt|
 baseIndicator.aggregatedAttribute.oclIsTypeOf(dt))
```

This OCL constraint verifies that the measure's data type is included in the set of data types to which the aggregator can apply to.

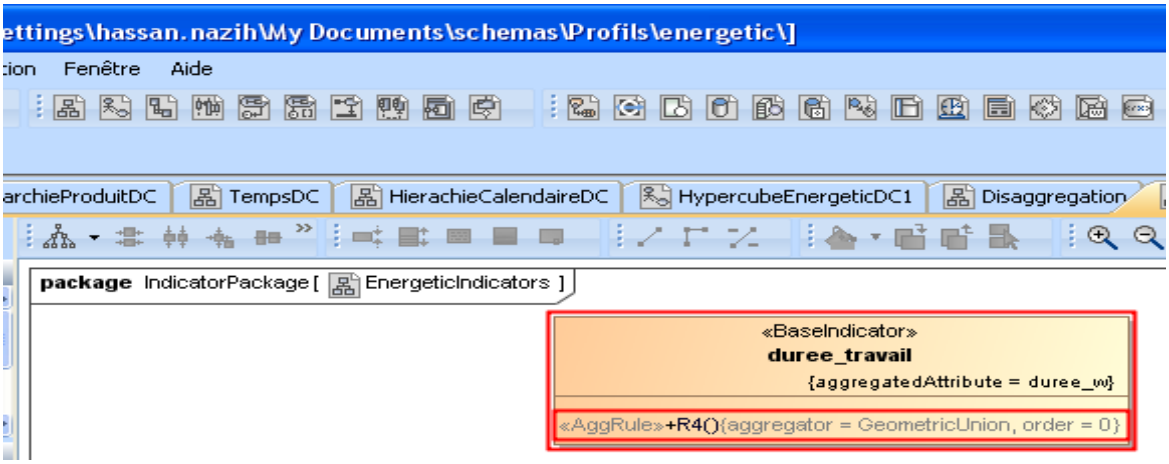


Figure 9.7. An example of an Aggregation IC's check.

Finally we present an example of Query IC which prevents an invalid combination of a measure and a dimension member. In our SOLAP application, it is not possible to have a value for the measure representing the number of animals (animaux_nb) for the spreading technical operation (Epandage). This Query IC is expressed using our UML profile as shown on Figure 9.8.

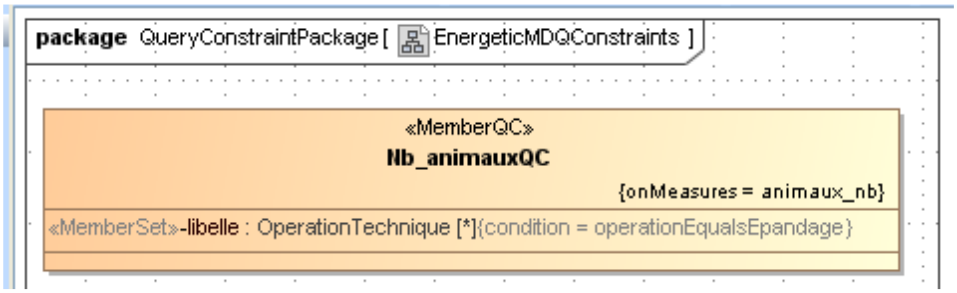


Figure 9.8. An example of a Query IC.

The MDX code implementing this IC is shown in Figure 9.9 and an example of its verification on an OLAP query is shown in Figure 9.10. We can notice that different visual policies are applied to the cells of the JRubik pivot table which are defined by different combinations of the measure "animaux_nb" and members of the dimension "Technical Operations": the invalid cells that involves the member spreading ("Epandage") is displayed with a red colour; the valid cells such as the one involving the member milking operation ("Traire") are displayed with a green colour; and finally the aggregated cells composed of valid and invalid cells such as the cell involving the member "AllOperationsTechniques" are represented with a yellow colour.

```
<!-- MDX Implementation of Query IC -->

<CalculatedMember name="Max_Nb_Animaux_QC" dimension="Measures">
<formula> [Measures].[Max_Nb_Animaux] </formula>
  <CalculatedMemberProperty name="FORMAT_STRING" expression=" IIf((([OperationsTechniques].CurrentMember in
  Filter([OperationsTechniques].Members,[OperationsTechniques].CurrentMember.Name='Epandage')) or
  (([OperationsTechniques].CurrentMember.Level.Ordinal > [OperationsTechniques].[OperationTechnique].Ordinal) AND
    (Ancestor([OperationsTechniques].CurrentMember,[OperationsTechniques].[OperationTechnique]) in
        Filter([OperationsTechniques].[OperationTechnique].Members,
            [OperationsTechniques].[OperationTechnique].CurrentMember.Name='Epandage')
    ))),
  &quot;|#|style=red&quot;,
  IIf([OperationsTechniques].CurrentMember in  Filter([OperationsTechniques].Members,
  Intersect(Descendants([OperationsTechniques].CurrentMember,[OperationsTechniques].[OperationTechnique],SELF),
  Filter([OperationsTechniques].[OperationTechnique].Members,
    [OperationsTechniques].[OperationTechnique].CurrentMember.Name='Epandage')).COUNT>0),
  &quot;|#|style= yellow&quot;,
  &quot;|#|style=green&quot;) )" />
</CalculatedMember>
```

Figure 9.9. The MDX implementation of the Query IC of Figure 9.8.

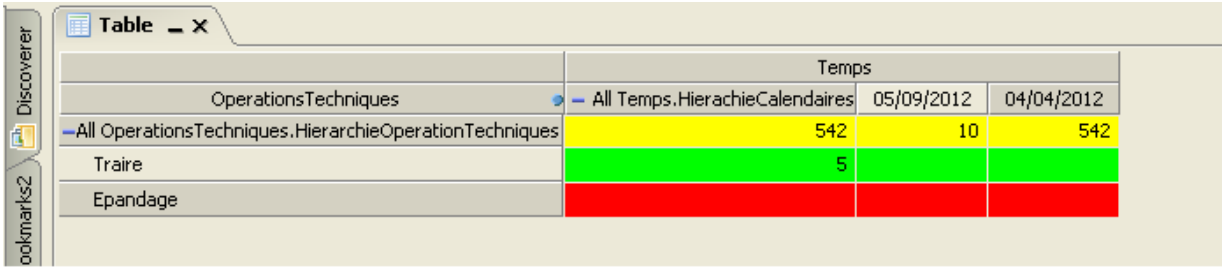


Figure 9.10. An example of execution of the Query IC.

9.7 Conclusions

Quantifying the impacts of human activities has now become an important issue for society, including agriculture. This quantification may eventually reduce these impacts by changing the most polluting practices, which is the reason several diagnoses defining a set of indicators exist to assess the energy performance of agricultural farms. However, these diagnoses present two important limitations: 1) they define indicators only at a global scale, avoiding the finest analysis of energy consumption related to technical operations, and 2) they are based on

classical data storage systems that do not allow interactive cartographic analysis of huge volumes of data.
By contrast, SDW and SOLAP technologies provide tools to interactively analyze massive geo-referenced data sets. Recently, SOLAP has been successfully applied in the agriculture domain. Thus, to overcome the previously described limits, in this chapter, we propose some new indicators to assess agricultural farm performance at a most detailed scale. We also show how it is possible to represent these indicators using a spatial data warehouse and analyze them by means of a classical SOLAP system.

PARTIE V : CONCLUSIONS ET PERSPECTIVES

10 Chapitre 10 : Conclusions et Perspectives

10.1 Rappel de la problématique

Afin de prendre en compte les volumes de données spatiales de plus en plus importants qui sont générées quotidiennement par les nouvelles technologies de l'information dans l'analyse décisionnelle, les systèmes d'Entrepôts de Données et OLAP spatiaux (EDS et SOLAP) ont été introduits [Bédard, 1997]. Ces systèmes étendent les systèmes d'ED et OLAP essentiellement par la définition de nouvelles structures de données multidimensionnelles pour la représentation de l'information spatiale, de nouvelles fonctions pour son agrégation et de nouveaux opérateurs pour son exploration [Malinowski et Zimányi, 2008]. Au niveau architectural, ces systèmes SOLAP comportent généralement quatre couches logicielles [Rivest et al., 2003]: (i) l'ETL spatial qui permet l'intégration et le chargement périodiques des données décisionnelles sources dans l'EDS; (ii) l'EDS qui permet l'historisation et la gestion de ces données; (iii) le serveur SOLAP qui implémente les opérateurs SOLAP; et (vi) le client SOLAP qui définit des interfaces graphiques utilisateurs intuitives pour déclencher les opérateurs SOLAP et afficher les résultats d'analyse sous différents formats.

Comme nous l'avons montré dans ce mémoire (cf. Chapitres 1, 4 et 5), la qualité d'analyse (S)OLAP, dépend de trois types de qualité : (A) qualité des données entreposées, (B) qualité d'agrégation et (C) qualité d'exploration.

La qualité de données (spatiales) [ISO/TC 211, 2002] est essentiellement affectée par trois types de problèmes qui sont l'incohérence logique, l'imprécision et la non exhaustivité (cf. Sections 4.2.1). *Les problèmes d'incohérence logique auxquels nous nous sommes intéressés dans ce travail de thèse se posent essentiellement lors de la phase d'intégration des données et sont dus aux différentes hétérogénéités qui peuvent exister entre les sources de données [Pinet et Schneider, 2010]*.

La qualité d'agrégation (i.e. la validité sémantique et l'exactitude des agrégations) est déminée par deux types de problèmes (cf. Sections 4.2.2 et 4.5) : (i) structurels liés à la présence de hiérarchies de dimension et relations fait-dimension irrégulières (cf. Section 4.5.1); et (ii) sémantiques liés à l'incompatibilité sémantique entre les natures de la mesure, la fonction d'agrégation et la hiérarchie de dimension (cf. Section 4.5.2).

La qualité d'exploration est essentiellement évaluée en termes de consistance des requêtes SOLAP formulées par les utilisateurs (cf. Sections 4.2.3 et 4.6). Comme les utilisateurs de ces systèmes sont sensés ignorer les contraintes sur les données [Inmon, 2005], ils peuvent formuler des requêtes SOLAP inconsistantes. L'interprétation erronée des résultats de ces requêtes peut mener l'utilisateur à prendre de mauvaises décisions. *Ces inconsistances de requêtes peuvent être d'ordre structurel ou sémantique (cf. Section 4.6). Comme les outils (S)OLAP existants garantissent la validité structurelle des requêtes, nous nous sommes focalisés sur les incohérences sémantiques.*

Dans la littérature, différentes approches ont été proposées pour le traitement des problèmes de qualité de données et d'agrégation, essentiellement la définition de Contraintes d'Intégrité (CI) pour vérifier la cohérence logique des données entreposées et des agrégations effectuées. Les CI sont des assertions qui définissent les conditions qui doivent être respectées par les données et éventuellement les conséquences de leur violation [Goertzen et Stausberg, 2007]. D'une manière générale, les approches existantes présentent des limites qui concernent essentiellement le manque d'exhaustivité des classifications de CI proposées, le manque d'utilisation de langages standards pour la spécification des CI au niveau conceptuel et la non implémentation automatique des CI (cf. Section 4.7).

10.2Contributions

Dans ce travail de thèse nous nous sommes focalisés sur les problèmes d'incohérence logique qui peuvent affecter les qualités de données, d'agrégation et d'exploration.
Afin de pallier limitations des travaux existants dans la littérature, nous avons proposé un Framework basé sur les langages standards UML et OCL pour le traitement de ces problèmes. Ce Framework permet la spécification conceptuelle indépendante des plateformes d'implémentation des CI SOLAP et leur implémentation automatisée via des générateurs de code dans différents niveaux de l'architecture SOLAP. Il comporte trois parties :

(I) Une classification des CI SOLAP qui sert de cadre général à l'identification et la définition des CI SOLAP et qui couvre toutes les classifications existantes (cf. Sections 5.3 et 7.5).

(II) Un profil UML implémenté dans l'AGL MagicDraw, pour la représentation conceptuelle des modèles spatio-multidimensionnels et de leurs CI en utilisant les langages de contraintes objet OCL et Spatial OCL. Ce profil définit trois métamodèles interconnectés :

- Le métamodèle EDS, qui permet la spécification des structures de données multidimensionnelles et des CI de données ;
- Le métamodèle d'agrégation, qui permet la spécification du modèle d'analyse SOLAP et des CI d'agrégation;
- Le métamodèle de CI de requêtes SOLAP, qui permet la spécification des CI d'exploration.

Par rapport aux modèles multidimensionnels proposés dans la littérature, notre profil offre un meilleur support pour la définition des CI relatives aux trois types de qualité SOLAP à travers notamment la formalisation de la plupart des concepts/caractéristiques de la modélisation spatio-multidimensionnelle qui sont indispensables à leur spécification et exclusivement la définition de nouveaux stéréotypes pour représenter les CI d'exploration. En dehors de l'aspect CI, notre profil permet aussi la modélisation des aspects spatio-multidimensionnels dynamiques et complexes liés à la définition du modèle d'analyse (S)OLAP (les indicateurs d'analyse et leurs règles de calcul).

(III) Une implémentation automatique basée sur les générateurs de code Spatial OCL2SQL et UML2MDX et permettant de traduire les spécifications conceptuelles en code dans deux niveaux de l'architecture SOLAP, EDS et Serveur SOLAP. Spatial OCL2SQL permet la génération automatique de code SQL pour Oracle Spatial pour l'implémentation du schéma physique de l'EDS et des CI de données. UML2MDX permet la génération automatique du schéma d'analyse OLAP Mondrian et d'implémentations MDX pour les CI de requêtes, et de manière exclusive la génération d'implémentations MDX pour les indicateurs d'analyse complexes. Ces deux générateurs utilisent des patrons de conversion d'UML et OCL vers SQL et vers MDX respectivement. En particulier, pour chaque classe de CI de requêtes

(respectivement type d'indicateurs) nous avons défini un patron MDX générique qui est automatiquement rempli par UML2MDX en remplaçant la partie variable par les éléments impliqués par la contrainte (respectivement l'indicateur).

Nous avons appliqué nos contributions dans le cadre de projets agri-environnementaux nationaux : (i) projets SIE Pesticides et Miriphyque portant sur le développement d'une application OLAP pour l'analyse des résultats de simulations de transfert de pesticides et (ii) projet EnergeTIC portant sur le développement d'une application SOLAP pour l'analyse des dépendances énergétiques des exploitations agricoles.

10.3Discussion et perspectives

Dans cette dernière partie, nous discutons les limites des solutions proposées et présentons les perspectives de recherche.

10.3.1 Limites

10.3.1.1 Limites de modélisation conceptuelle des CI

Dans cette thèse nous avons fait le choix de baser nos solutions pour la spécification des CI (S)OLAP sur les langages standards UML et OCL. Ce choix est essentiellement motivé par la capacité de ces langages à modéliser des aspects complexes dans des domaines spécifiques notamment en raison de leur facilité d'extension et aussi par la possibilité d'automatiser l'implémentation des CI.

10.3.1.1.1 Spécification des CI de données, d'agrégation sémantiques et de requêtes

Pour l'expression de plus de CI SOLAP avec Spatial OCL, il est nécessaire de prendre en compte :

a) **Les relations spatiales métriques et d'ordre**. Spatial OCL permet seulement d'exprimer les relations spatiales topologiques du modèle 9IM entre les géométries simples et complexes [Duboisset, 2007]. Ce langage n'intègre pas les relations spatiales métriques (e.g. surface) et d'ordre (e.g. en face de) [Parent et al., 1999; Salehi, 2009]. Les relations spatiales métriques peuvent par exemple être intéressantes pour exprimer des CI de données (e.g. la surface d'un objet géométrique ou la distance entre deux objets doit être supérieure à une valeur donnée); pondérer l'agrégation de mesures numériques par rapport à l'agrégation de mesures spatiales [Glorio et Trujillo, 2008], etc. L'intégration de ces relations dans OCL pourrait se faire d'une manière similaire aux relations spatiales topologiques 9IM [Duboisset, 2007].

b) **Les fonctions d'agrégation numériques, spatiales, textuelles, booléennes et temporelles**. Spatial OCL n'intègre que la fonction d'agrégation Sum. Il doit donc d'être étendu par les autres fonctions d'agrégation numériques (e.g. Mode), les fonctions d'agrégation spatiales [Shekhar et al., 2001; Silva et al., 2008; Ruiz et Times, 2009], temporelles (e.g. union temporelle), textuelles [Ravat et al., 2008] et booléennes [Golfarelli et al., 1998]. L'intégration des fonctions d'agrégation spatiales par exemple est nécessaire pour l'expression de certaines CI de données (e.g. l'union spatiale des zones d'épandage doit être incluse dans la géométrie de la commune – cf. Section 3.2). L'extension de ce langage par ces différents types de fonctions pourrait se faire en s'inspirant des travaux de [Cabot et al., 2010] qui étendent OCL par un ensemble de fonctions numériques.

c) **Les relations temporelles.** Spatial OCL n'intègre pas les relations temporelles topologiques, métriques et d'ordre définies par exemple dans [Allen, 1983; Hornsby et al., 1999]. L'intégration de ces relations temporelles permettrait par exemple d'exprimer des CI spatio-temporelles [Salehi, 2009].

10.3.1.1.2 Spécification des CI structurelles d'agrégation

Dans cette thèse, nous spécifions les CI structurelles d'agrégation (structures non agrégeables cf. Section 4.5.1) en utilisant un approche basée sur les multiplicités UML et le langage OCL [Boulil et al., 2011] d'une façon similaire à [Mazón et al., 2009; Pinet et Schneider, 2010]. La transformation de ces structures conceptuelles non agrégeables vers des structures logiques agrégeables peut être réalisée en appliquant les règles de transformation proposées dans la littérature (cf. Section 4.5.1). Certaines de ces structures complexes peuvent poser des problèmes lors des agrégations si des *vues matérialisées* sont employées pour l'optimisation des performances de requêtes (e.g. hiérarchies non strictes); d'autres sont indépendantes de cet aspect physique (e.g. relations fait-dimension non strictes). Ces structures qui dépendent de l'implémentation devraient être gérées par les outils (S)OLAP sans nécessiter un traitement particulier de la part des concepteurs. Une étude est donc nécessaire pour distinguer les problèmes structurels d'agrégation indépendants de l'implémentation qui doivent être spécifiées au niveau conceptuel, des problèmes qui sont dépendants de l'implémentation.

10.3.1.1.3 Spécification d'autres classes de CI

Les langages de spécification de CI proposés (Spatial OCL et profil UML) ne permettent pas de spécifier toutes les classes de CI définies par notre classification. Les classes non spécifiées sont : les CI de métadonnées, les CI d'agrégation spécifiques au domaine d'application et les CI de visualisation [Bellatreche et al., 2005]. Pour l'expression de ces classes de CI notre profil UML peut être facilement étendu; par exemple par le rajout de valeurs marquées qui représentent les principales métadonnées en lien avec le standard ISO 19115 [ISO, 2003]; aussi par d'autres stéréotypes qui permettrait à l'utilisateur de définir ces propres contraintes d'agrégation, ainsi qu'une formalisation du profil utilisateur [Villarroel et al., 2006] pour spécifier par exemple des contraintes de visualisation en fonction des préférences utilisateur.

10.3.1.2 Limites d'implémentation des CI

10.3.1.2.1 Architecture d'implémentation non basée sur un serveur SOLAP

Afin de valider nos solutions nous avons opté pour une architecture ROLAP "Spatiale" composée du SGBD Oracle Spatial, du serveur OLAP Mondrian et du client SOLAP JRubik**.** Bien que cette architecture n'intègre pas un serveur SOLAP, elle est suffisante pour le type d'implémentation choisi notamment pour les CI de requêtes. En effet, nous avons défini des implémentations MDX génériques en utilisant entre autres l'élément MDX *NamedSet* qui nous permet de construire les ensembles de membres problématiques (i.e. ceux qui violent la contrainte s'ils sont combinés avec d'autres membres problématiques – cf. Section 5.5). A l'intérieur de cet élément, il est possible de définir des requêtes SQL spatiales pour sélectionner les membres impliqués dans la contrainte; cet SQL (spatial) est interprété par le SGBD spatial. Pour passer à d'autres types d'implémentations (par exemple, intégrant du MDX spatial), on pourrait étendre cette architecture en introduisant un serveur SOLAP (e.g. GeoMondrian).

10.3.1.2.2 Traduction automatique de l'OCL conceptuel en OCL logique

Contrairement aux bases de données; le processus de développement des systèmes ROLAP est caractérisé par la présence de différentes variantes de schémas logiques (en étoile, en flocons de neige et hybrides - cf. Section 2.5.2.1). Afin d'adapter l'outil Spatial OCL2SQL qui est développé pour les bases de données spatiales [Duboisset et al., 2007] à ce processus, nous avons rajouté un module qui transforme le modèle multidimensionnel conceptuel exprimé avec notre profil UML vers l'un des schémas logiques (en étoile ou en flocons de neige) en fonction du choix du concepteur. A noter que ce module ne permet pas de transformer l'OCL conceptuel vers un OCL logique; cette tache est actuellement réalisée de façon manuelle. Il serait donc intéressant pour automatiser tout le processus de définition des CI, de s'attaquer à la définition des règles de mapping de l'OCL conceptuel vers l'OCL logique.

10.3.2 Perspectives

10.3.2.1 Autres composantes de la qualité

Les travaux présentés dans ce mémoire portent sur les problèmes d'incohérence logique qui peuvent se poser au niveau des données, des agrégations et des requêtes. Afin de mettre en place un système de gestion de qualité SOLAP qui tient compte de toutes les composantes de la qualité de données définies par exemple dans le standard ISO 19113 [ISO/TC 211, 2002], il serait intéressant d'élargir ces travaux aux autres types de problèmes de qualité qui sont la non exhaustivité et l'imprécision. Cette perspective de recherche soulève d'abord des questionnements quant aux représentations conceptuelle et physique des données présentant ces types de problèmes (données imprécises et/ou non exhaustives), à leur agrégation (est-il par exemple nécessaire de redéfinir les fonctions d'agrégation SOLAP?) et à leur interrogation (définition de langages de requêtes intégrant des prédicats d'imprécision et d'exhaustivité [Bosc et Pivert, 1995]). Ensuite la question de la définition de langages pour la spécification et l'implémentation des contraintes sur ce type de données, de requêtes (e.g. requêtes vagues [Bejaoui, 2009]) et d'agrégations peut être investiguée. Pour l'imprécision par exemple ceci pourrait se faire en étendant des travaux effectués dans les bases de données spatiales pour la représentation des données spatiales vagues [Bejaoui et al., 2009] et des travaux dans les ED [Pedersen et al., 2001; Burdick et al., 2006; Singh et al., 2007].
Finalement, comme pour l'incohérence logique, il peut être intéressant d'étudier l'implémentation des contraintes de précision et d'exhaustivité tout au long du processus d'entreposage et d'analyse des données.

10.3.2.2 Implémentation des CI au niveau ETL

Dans cette thèse nous proposons une implémentation des CI au niveau des couches EDS et Serveur SOLAP (cf. Section 5. 5). Il serait intéressant d'étendre cette implémentation au niveau de l'ETL. Cette implémentation ETL nécessitera certainement la définition de modèles qui représentent les sources de données; ensuite des techniques de propagation de contraintes vers les premiers niveaux possibles de leur vérification dans le processus ETL pourraient être employées [Liu et al., 2009]. Aussi, une étude plus détaillée [Boulil et al., 2011] pour déterminer quelles sont les couches SOLAP les plus adéquates pour l'implémentation de chaque type de CI serait une perspective intéressante.

10.3.2.3 Politiques de correction

Dans ce travail de thèse, nous nous sommes seulement intéressés à la détection des incohérences au niveau des données, des agrégations et des requêtes. Autrement dit, nous ne

proposons pas de politiques de correction pour les différentes incohérences détectées. Nous laissons le soin à l'utilisateur d'appliquer les politiques de correction qu'il juge adéquates, par exemple interdire le stockage dans le cas de données erronées. Bien que dans certains cas comme par exemple les requêtes définissant des combinaisons insensées de hiérarchies, les politiques de réponse peuvent être triviales, i.e. interdire l'exécution de la requête. En règle générale, différentes politiques peuvent être envisagées notamment concernant les incohérences de données et d'agrégations. Ces politiques peuvent aller de la plus radicale (comme par exemple l'interdiction de l'exécution de la requête ou du stockage de la donnée) à la plus tolérante en relaxant les CI [Decker et Martinenghi, 2006]. Cette relaxation impliquerait le traitement des CI au niveau de l'exploration en se basant par exemple sur des techniques de reformulation et de recommandation de requêtes [Kozmina et Niedrite, 2011; Negre, 2011] Une piste de modélisation possible de ces politiques au niveau conceptuel serait d'intégrer des règles du type "si condition alors actions" dans la modélisation des CI [Prat et al., 2011].
Avec ces politiques de correction, il faudra aussi prévoir des moyens pour informer l'utilisateur sur les corrections effectuées (afin qu'ils les prennent en compte dans son analyse) et/ou la qualité des résultats de requêtes [Devillers et al., 2007]. Concernant ce dernier point, on pourrait se baser sur les métriques de qualité définies par le standard ISO 9126 [ISO/IEC, 2001]. Il faudra particulièrement investiguer des techniques de visualisation des anomalies sur des cartes [Devillers et al., 2007].
Finalement, nous pensons que la définition d'un système de recommandation qui propose différentes alternatives de correction à l'utilisateur [Pedersen et al., 2001] serait une piste intéressante.

10.3.2.4 Mise en place d'une architecture MDA

L'implémentation que nous avons proposée dans cette thèse pour valider nos propositions est une implémentation "ad hoc" car elle considère des plateformes technologiques spécifiques (Oracle Spatial et Mondrian). Afin de permettre l'implémentation facile et automatique dans différentes plateformes d'implémentation nous pensons qu'il est très intéressant de mettre en place une architecture de définition des CI SOLAP suivant les principes MDA. MDA (*Model-Driven Architecture*) [OMG, 2003b] suggère la définition de modèles et de transformations entre modèles à différents niveaux d'abstraction; ceci en utilisant des langages standards compatibles avec MOF[48] tels qu'UML, OCL, QVT et CWM[49]. MDA définit trois niveaux d'abstraction : CIM (*Computation Independent Model*) qui représente les besoins utilisateur via des langages métier; PIM (*Platform Independent Model*) qui utilise des langages informatiques indépendants de plateformes d'implémentation (e.g. UML); et PSM (*Platform Specific Model*) incluant des informations spécifiques à un type plateformes d'implémentation (e.g. CWM Relationnel [OMG, 2003a]).
La mise en place de cette architecture (cf. Figure 10.1) nécessitera la définition de langages pour représenter les CI au niveau CIM (les langages métier CIM améliorent la lisibilité des spécifications vis-à-vis des utilisateurs); au niveau PSM pour représenter les CI et les modèles logiques relatives à différents types d'implémentation (S)OLAP (MOLAP, ROLAP, HOLAP – cf. Section 2.4.1); ainsi que des transformations entre ces modèles notamment pour le cas des implémentations MOLAP et HOLAP.
Ces modèles et transformations devraient être spécifiés en se basant sur des langages standards qui sont compatibles avec MOF (e.g. UML, OCL, CWM) tel que suggéré par MDA (cf. Figure

[48] Meta Object Facility (MOF)[OMG, 2011b]
[49] Common Warehouse Metamodel [OMG, 2003a].

10.1). Notons que la définition d'une architecture MDA pour les systèmes ROLAP a été investiguée dans [Mazon et Trujillo, 2008; Pardillo et al., 2010a]. Ces travaux se sont seulement intéressés à la définition de modèles et de transformations entre modèles pour les modèles de l'ED et d'analyse OLAP; ils ne considèrent pas les CI.

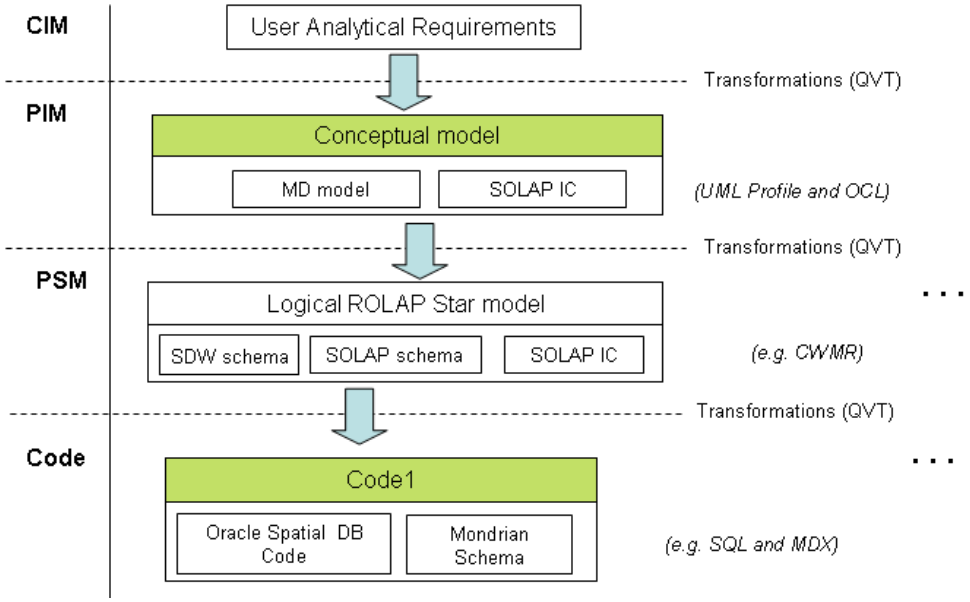


Figure 10.1. Définition d'une architecture MDA pour les CI SOLAP.

10.3.2.5 Complétude de la classification de CI

Dans cette thèse nous avons essayé de définir notre classification en considérant les classes de CI identifiées dans les typologies existantes et dans les travaux sur l'agrégation et l'exploration (S)OLAP (cf. Sections 4.4, 4.5 et 4.7). Il est possible que des classes de CI restent encore à identifier. Cette recherche est une perspective intéressante.

10.3.2.6 Définition de CI de versionning

Dans cette thèse nous ne sommes pas intéressés à la définition de CI pour vérifier la cohérence logique entre plusieurs versions d'ED. Les versions multiples d'un ED peuvent être dues à des changements au niveau schéma ou au niveau instance. Il serait intéressant de définir des CI pour contrôler les incohérences entre versions dues à ces deux types de changements [Turki et al., 2010]. Cette perspective soulève les questions de la représentation conceptuelle et physique des différentes versions d'un ED ainsi que des questions sur les techniques et niveaux d'implémentation de ce type de CI.

10.3.2.7 Satisfiabilité de CI définies

Dans cette thèse nous n'avons pas abordé la problématique du test de satisfiabilité des ensembles de CI définies (i.e. existe-t-il des CI qui se contredisent?). Cette perspective de recherche peut être investiguée en utilisant, entre autres, des travaux tels que [Roussey et Pinet, 2010].

10.3.2.8 Expressivité et optimisation des implémentations

Le but étant de montrer la faisabilité de notre approche, nous ne sommes pas intéressés à délimiter précisément le pouvoir expressif des langages de spécification de CI que nous proposons et à optimiser le code généré.

PARTIE VI : ANNEXES

Annexe A : Langages de contraintes objet OCL et Spatial OCL

Cette annexe présente les langages OCL (cf. Section 1) et Spatial OCL (cf. Section 2).

Ce didacticiel est issu de[Pinet, 2010].

1 Langage de contraintes objet OCL

OCL (*Object Constraint Language*) est un langage formel pour l'expression de contraintes et de requêtes sur des diagrammes UML (diagrammes de classes, états-transitions, etc.)[OMG, 2006]. OCL peut être utilisé soit comme langage de contraintes ou comme langage de requête, i.e. spécifier des expressions, qui une fois évaluées, retournent soit un résultat booléen, soit les valeurs qui satisfont les conditions définies par ces expressions. En tant que langage de contraintes, OCL permet de spécifier entre autres des invariants sur des classes et des types de données dans un modèle UML, exprimer des pré et post conditions sur des opérations et méthodes, définir des invariants sur les stéréotypes d'un profil UML; etc.

Dans nos travaux, nous avons utilisé essentiellement les contraintes OCL de type invariant pour exprimer des contraintes au niveau modèle sur le modèle multidimensionnel (cf. Chapitres 5 et 7,[Boulil et al., 2010b]) et au niveau métamodèle sur les stéréotypes du profil UML SOLAP que nous proposons (cf. Chapitres 5 et 6, et [Boulil et al., 2011]).

Nous présentons donc ici les principaux concepts liés aux invariants OCL à l'aide d'exemples. Les exemples sont basés sur le diagramme de classes UML de la figure suivante (cf. Figure 1).

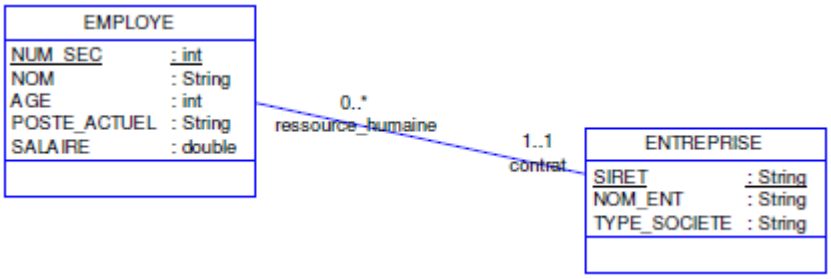


Figure 1. Modèle UML des entreprises et leurs employés.

Un invariant OCL (mot clef *inv*) peut se définir dans le contexte d'une classe et précise les conditions qui doivent être satisfaites par ses instances.
Par exemple, l'invariant OCL suivant exprime le fait que tous les employés dont le poste actuel est chercheur doivent percevoir un salaire supérieur ou égal à 1500€.

```
context Employe inv:
self.poste_actuel = 'chercheur' implies self.salaire >= 1500
```

Cette contrainte OCL est équivalente à la formule logique suivante :
$\forall self \in Employe, (self.poste\_actuel = 'chercheur' \rightarrow self.salaire \geq 1500)$

En OCL, *self* est toujours une variable-itérateur qui correspond à une instance du contexte (ici le contexte est la classe "Employe"). Une contrainte doit obligatoirement être respectée pour toute instance du contexte. "self.poste_actuel" et "self.salaire" sont deux attributs (cf. Figure 1). L'opérateur *implies* correspond à l'implication logique.

1.1 Navigation entre les classes

La navigation permet de fixer des contraintes impliquant des objets de classes différentes. Par exemple, la contrainte suivante signifie que les petites et moyennes entreprises (PME) ne doivent pas avoir plus de 499 salariés.

```
context Entreprise inv:
self.type_societe = 'PME' implies self.ressource_humaine->size() <= 499
```

Cette contrainte OCL est équivalente à la formule logique suivante :
$\forall self \in Entreprise, (self.type\_societe = 'PME' \rightarrow size(self.ressouce\_humaine) \leq 499)$

L'attribut "type_societe" est défini dans la classe "Entreprise", et le nombre de salariés s'obtient à partir des instances de la classe "Employe". Une navigation est donc réalisée entre ces deux classes via l'association "ressource_humaine". Ici, *self* correspond à une entreprise, et "self.ressource_humaine" retourne une collection contenant tous les employés de l'entreprise *self*. L'opération *size()* retourne la taille d'une collection (i.e. le nombre d'objets la composant).

1.2 Opérations sur les éléments d'une collection

La syntaxe d'une opération portant sur les éléments d'une collection est la suivante :

```
<collection> -> <opération> ([<itérateur> |] <expression>)
```

Où l'expression booléenne <expression> est évaluée pour chacun des éléments de la <collection>. <itérateur> est une variable facultative qui sert à faire référence à l'élément courant dans l'expression <expression>. <opération> peut être entre autres l'une des opérations suivantes :
1) *forAll* représente le quantificateur universel qui indique que l'expression booléenne doit être vraie pour toutes les instances de <collection>. Par exemple, on peut spécifier le fait que tous les employés des établissements de recherche doivent avoir au moins 18 ans.

```
context Entreprise inv:
self.type_societe = 'établissement de recherche' implies
self.ressource_humaine->forAll(age>=18)
```

Si nous représentons la contrainte sous une forme logique, nous obtenons :
$\forall self \in Entreprise, (self.type\_societe = \text{'établissement de recherche'} \rightarrow$
$\forall x \in self.ressouce\_humaine, x.age \geq 18)$

2) *exists* représente le quantificateur existentiel qui indique que l'expression booléenne doit être vraie pour au moins un élément de la <collection>. Par exemple, on peut exprimer le fait qu'il doit exister (un ou) plusieurs chercheurs dans un établissement de recherche.

```
context Entreprise inv:
self.type_societe = 'établissement de recherche' implies
self.ressource_humaine->exists(e:Employe|e.poste_actuel='chercheur')
```

La représentation de cette contrainte sous une forme logique est :
$\forall self \in Entreprise, (self.type\_societe = \text{'établissement de recherche'} \rightarrow$
$(\exists e \in self.ressouce\_humaine, e.post\_actuel = \text{'chercheur'}))$

3) *size* retourne le nombre d'éléments (i.e. le nombre d'objets) d'une collection.

4) *select* retourne un sous-ensemble de la <collection> qui ne contient que les éléments qui vérifient l'<expression>. Par exemple, on peut l'utiliser dans une contrainte pour sélectionner les employés majeurs en indiquant :

```
self.ressource_humaine->select(age>=18)
```

On utilise *allInstances* pour retourner toutes les instances d'une classe. "Employe.allInstances()" retourne par exemple la collection de toutes les instances de la classe "Employe". Il existe de très nombreuses autres opérations applicables sur les collections en OCL[OMG, 2006].

2 Langage de contraintes objet Spatial OCL

Pour exprimer les contraintes d'intégrité spatiale, nous avons utilisé l'extension spatiale d'OCL, Spatial OCL, proposée dans le cadre des travaux de thèse de Magali Duboisset[Duboisset, 2007].
Spatial OCL permet d'exprimer les relations spatiales topologiques du modèle 9IM et du modèle CBM (Calculus-Based Model) [Clementini et al., 1993] entre géométries simples et complexes.
Pour ce faire, Spatial OCL étend OCL par :

- les types géométriques simples *Point*, *Ligne* et *Region*. Une géométrie complexe est définie comme un ensemble de géométries simples.
- les relations spatiales topologiques 9IM (cf. Figure 2).
- et les relations spatiales topologiques CBM.

Spatial OCL définit les extensions spatiales suivantes : OCL9IM, OCLCBM, OCLADV et OCLTRCR. Dans nos travaux de thèse nous avons utilisé l'extension OCL9IM qui est décrite dans la section suivante.

2.1 OCL9IM

OCL9IM est le langage OCL auquel ont été rajoutées les 8 relations topologiques du modèle 9IM entre régions simples (cf. Figure 2). Une région simple est définie comme suit.

Définition 1. Région (simple). Une région (simple) est un ensemble de points connectés, sans trou, dans R^2

La syntaxe générale des operateurs spatiaux OCL9IM est décrite comme suit.

```
geoA.opérateur_topo(geoB): Boolean
```

Où *opérateur_topo* ∈ {*inside, contains, covers, coveredBy, disjoint, equal, overlap, meet*}, *geoA* et *geoB* sont de type *Region*.

geoA et *geoB* sont les paramètres de l'opération. Cette opération retourne vrai ou faux (un booléen) suivant que la relation topologique entre *geoA* et *geoB* est vraie ou fausse.

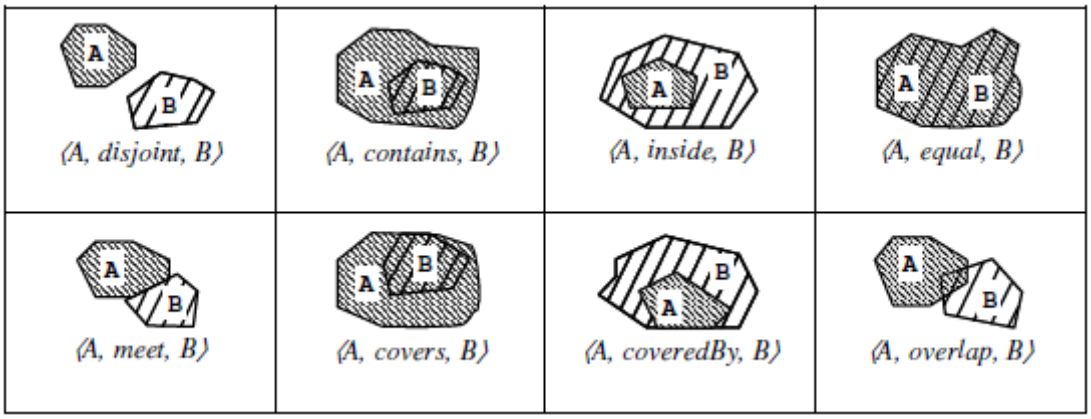


Figure 2. Huit relations topologiques possibles entre deux régions simples[Egenhofer et Herring, 1994].

Ci-après un exemple de contrainte OCL9IM qui spécifie que les bâtiments d'une mairie sont généralement localisés dans la commune qui correspond à leur adresse. Cette contrainte est basée sur le modèle de la figure 3.

```
context Bâtiment_Mairie inv:
(self.geo) .inside (self.code_commune.geo) or
(self.geo) .coveredBy (self.code_commune.geo)
```

La contrainte doit être vérifiée pour chaque instance *self* de la classe "Bâtiment_Mairie". L'expression "self.code_commune" retourne l'instance de "Commune" associée à *self* par l'association "code_commune".

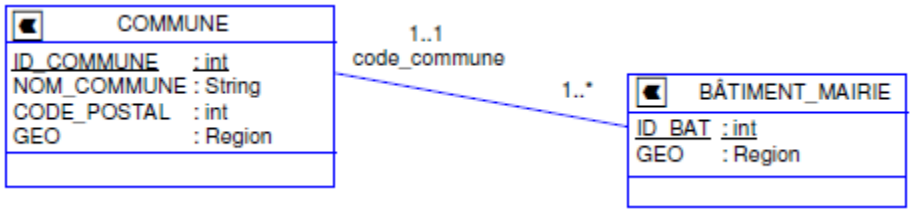


Figure 3. Modèle UML des bâtiments de mairie et leurs communes.

OCL9IM permet aussi d'exprimer des spatiales relations topologiques entre régions composites. Une région composite est vue come une union de régions telle que définie ci-dessous.

Définition 2. Union de régions (région composite). Une union de régions UR est un ensemble fini, non vide, de régions simples R_1, …, R_n tel que :

- $R_i° \cap R_j° = \emptyset, \forall\ i \neq j$

- $\delta R_i \cap \delta R_j = \varnothing, \forall\ i \neq j$

Où R° et δR sont respectivement l'intérieur et la frontière de R.

Nous avons vu que OCL9IM permet la spécification de contraintes entre des objets géométriques simples. Une région composite est un ensemble de régions simples. Par conséquent, les opérations standards d'OCL sur les collections (cf. Section 1.2) peuvent être utilisées en OCL9IM pour accéder aux différentes régions simples d'une région composite. Ainsi, les opérations standards OCL comme *size*, *forAll*, *exists* ou *select* peuvent être appliquées sur des attributs de type *Set(Region)*, c'est-à-dire de type « région composite ».

La syntaxe générale des opérations sur les régions composites est :

```
geo -> opérationSurCollection(...)
```

où *geo* est de type *Set(Region)* et *opérationSurCollection* est une opération OCL sur les collections.

Ainsi, il devient possible d'exprimer des relations topologiques entre les régions composites. Par exemple, dans le modèle de la figure 4, les attributs "bâtiments" de la classe "Îlot_de_Bâtiments" et "geo″ et de la classe "Centre_Ville" sont de type *Set(Region)*. Cela signifie, que chacun de ces attributs stocke une région composite. Chaque attribut "bâtiments" de la classe "Îlot_de_Bâtiments″ stocke un îlot de bâtiments, c'est-à-dire un ensemble de bâtiments proches. Chaque valeur de cet attribut est l'ensemble des régions simples (i.e. une région composite) qui correspondent à l'îlot. Un centre ville est défini comme un ensemble de zones dans la ville. Ces zones sont stockées dans l'attribut "geo" de la classe "Centre_Ville". L'association présente sur le diagramme de la figure 4 indique que les zones d'un centre ville peuvent contenir des îlots de bâtiments. Ces classes sont illustrées sur un exemple à la figure 5. Sur le dessin de cette figure, le centre ville est composé de 2 zones distinctes. Deux exemples d'îlots sont aussi indiqués, le premier îlot fait 6 bâtiments et le second en fait 4. Les opérations d'OCL sur les ensembles peuvent être appliquées aux attributs "bâtiments″ et "geo" des classes, comme illustré ci-dessous.

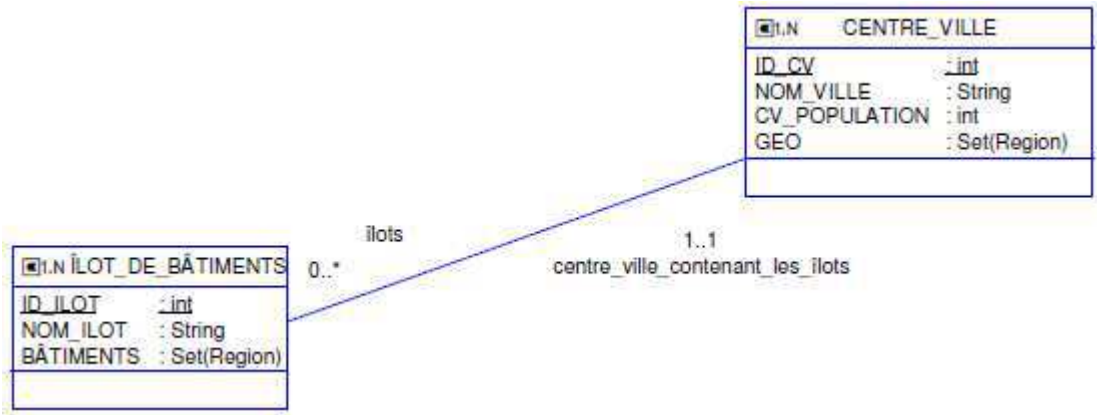


Figure 4. Les centres ville et leurs îlots de bâtiments .

Soit I un îlot et C un centre ville. Si I et C sont associés par "centre_ville_contenant_les_îlots" alors pour chaque bâtiment b de I, il doit exister une partie c du centre ville C tel que b est dans c. Cette contrainte s'exprime en OCL9IM :

```
context Îlot_de_Bâtiments inv:
self.bâtiments -> forAll( b | self.centre_ville_contenant_les_îlots.geo
-> exists(c | (b) .inside (c)))
```

Les attributs "geo" et" bâtiments" sont de type *Set(Region),* et donc b et c sont de type *Region.*

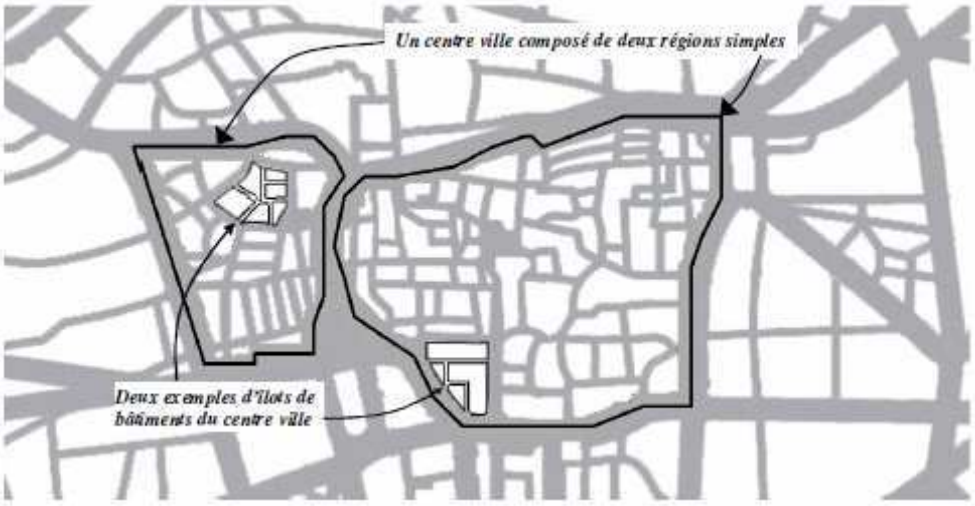


Figure 5. Exemple d'un centre ville et de ses îlots de bâtiments.

Pour faciliter l'expression de contraintes sur les régions composites. OCL9IM a été étendu par les relations topologiques entre régions composites définies par le modèle basé sur des adverbes proposé par[Claramunt, 2000]. Cette extension d'OCL9IM est appelée OCLADV.

Enfin, ces extensions spatiales d'OCL (OCL9IM et OCLADV) ont été implémentées par le générateur de code Spatial OCL2SQL (cf. Section 4.3.2)

2.2 OCLCBM

OCLCBM suit exactement la même approche que OCL9IM mais en se basant sur les relations topologiques de modèle CBM[Clementini et al., 1993].

La syntaxe générale des operateurs spatiaux OCLCBM est décrite ci-dessous.
`geoA.opérateur_topo (geoB):Boolean`

où *opérateur_topo* ∈ { *touch, in, overlap, disjoint, cross* }, *geoA* et *geoB* sont de type *Region*.

geoA et *geoB* sont les paramètres de l'opération. Cette opération retourne vrai ou faux (un booléen) suivant que la relation topologique entre *geoA* et *geoB* est vraie ou fausse.

Pour faciliter l'expression de contraintes sur les régions composites. OCLCBM a été étendu par les relations topologiques entre régions composites définies par le modèle TRCR (*Topological Relationships for Composite Regions*) [Clementini et Felice, 1995] qui est basé sur le modèle CBM. Cette extension d'OCLCBM est appelée OCLADV OCLTRCR.

Annexe B : Contraintes OCL du profil UML proposé

Nous présentons ici les contraintes spécifiées et implémentées en OCL du profil UML.

Le profil UML 2.0 que nous proposons permet la modélisation conceptuelle des systèmes d'EDS et SOLAP. Il est formalisé comme une extension (spécialisation) du métamodèle UML 2.0 et définit un ensemble de stéréotypes, de valeurs marquées et de contraintes OCL. Les stéréotypes sont des spécialisations de metaclasses UML (e.g. class) qui sont représentées par <<nom_du_stéreotype>> et/ou une icône. Les valeurs marquées sont des propriétés de stéréotypes. Les contraintes OCL permettent de préciser la sémantique des stéréotypes et des valeurs marquées dans le domaine d'étude. Elles formalisent leurs définitions et permettent d'éviter leurs mauvaises utilisations par les concepteurs.

Ce profil, qui est destiné à être utilisé par les concepteurs des systèmes d'EDS et SOLAP, est implémenté dans l'AGL MagicDraw. Le choix de cet outil (MagicDraw) parmi tant d'autres outils (e.g. Objecteering, Rational Rose, etc.) est principalement motivé par le fait que cet outil est le seul qui permette la définition des contraintes OCL au niveau métamodèle et leur vérification au niveau modèle sur les instances du profil (modèles conceptuels) et ceci sans aucun effort de programmation.
Dans ce qui suit, nous présentons les principales contraintes OCL du métamodèle EDS (cf. Section 1) et du métamodèle d'agrégation (cf. Section 2).

1 Contraintes du métamodèle EDS

Le métamodèle EDS (cf. Figure 1) permet de spécifier les structures multidimensionnelles d'EDS (e.g. faits, mesures, dimensions) et du modèle d'analyse SOLAP (e.g. hypercubes, hiérarchies, niveaux d'agrégation, etc.). Ces structures sont nécessaires pour la spécification des CI SOLAP.

Nous présentons ici les principales contraintes associées à chacun des stéréotypes définis par ce métamodèle.

1.1 SMDModel

a) Un modèle spatio-multidimensionnel doit avoir au moins un hypecube.

```
context SDModel inv:
self.ownedMember->select (m | m.oclIsTypeOf(Hypercube))->size()>=1
```

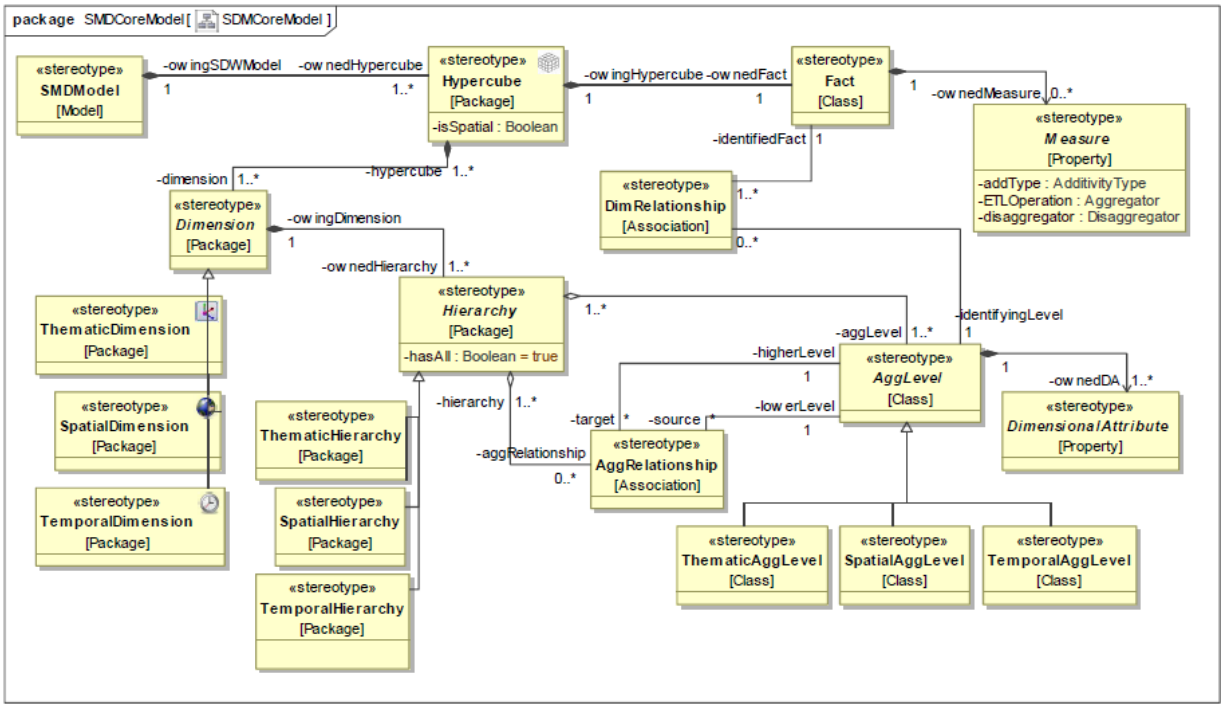


Figure 1. Partie Métamodèle EDS du profil UML.

1.2 Hypercube

a) Un hypercube doit avoir au moins deux dimensions.

```
context Hypercube inv :
self.ownedMember->select (m | m.oclIsKindOf(Dimension))->size()>=2
```

b) Un hypercube doit avoir exactement une classe de fait.

```
context Hypercube inv :
self.ownedMember->select (m | m.oclIsTypeOf(Fact))->size()=1
```

c) Un hypercube spatial doit avoir au moins une dimension spatiale ou une mesure spatiale.

```
context Hypercube inv :
self. isSpatial implies (
self.ownedMember->select (m | m.oclIsTypeOf(SpatialDimension))->size()>0 or
(self.ownedMember->exists (m | m.oclIsTypeOf(Fact) and
   m.ownedAttribute-> select (a | a.oclIsTypeOf(SpatialMeasure))->size()>0)
))
```

d) Un hypercube doit avoir au moins un indicateur de base.

```
context Hypercube inv:
self.ownedMember->select (m | m.oclIsTypeOf(BaseIndicator))->size()>=1
```

1.3 Dimension

a) Une dimension doit avoir au moins une hiérarchie de dimension.

```
context Dimension inv:
self.ownedMember->select (m | m.oclIsKindOf(Hierarchy))->size()>=1
```

b) Une dimension thématique (non spatiale et non temporelle) ne doit avoir que des hiérarchies de dimension thématiques (non spatiales et non temporelles).

```
context ThematicDimension inv:
self.ownedMember->select       (m       |       m.oclIsKindOf(Hierarchy))->
forAll(oclIsTypeOf(ThematicHierarchy)
```

c) Une dimension spatiale doit avoir au moins une hiérarchie de dimension spatiale.

```
context SpatialDimension inv:
self.ownedMember->select (m | m.oclIsTypeOf(SpatialHierarchy))->size()>=1
```

d) Une dimension temporelle ne doit avoir que des hiérarchies de dimension temporelles.

```
context TemporalDimension inv:
self.ownedMember->select       (m       |       m.oclIsKindOf(Hierarchy))->
forAll(oclIsTypeOf(TemporalHierarchy))
```

1.4 Hiérarchie de dimension

a) Une hiérarchie de dimension doit avoir au moins un niveau d'agrégation.

```
context Hierarchy inv:
self.ownedMember->select (m | m.oclIsKindOf(AggLevel))->size()>=1
```

b) Une hiérarchie de dimension thématique ne doit avoir que des niveaux d'agrégation thématiques (non spatiaux et non temporels).

```
context ThematicHierarchy inv:
self.ownedMember->select       (m       |       m.oclIsKindOf(AggLevel))->
forAll(oclIsTypeOf(ThematicAggLevel)
```

c) Une hiérarchie de dimension spatiale doit avoir au moins un niveau d'agrégation spatial.

```
context SpatialHierarchy inv:
self.ownedMember->select (m | m.oclIsTypeOf(SpatialAggLevel))->size()>=1
```

d) Une hiérarchie de dimension temporelle ne doit avoir que des niveaux d'agrégation temporels.

```
context TemporalHierarchy inv:
self.ownedMember->select       (m       |       m.oclIsKindOf(AggLevel))->
forAll(oclIsTypeOf(TemporalAggLevel))
```

1.5 Niveau d'agrégation

a) Un niveau d'agrégation doit avoir au moins un identifiant.

```
context AggLevel inv:
self.ownedMember->select (m | m.oclIsTypeOf(IdAttribute))->size()>=1
```

b) Un niveau d'agrégation peut avoir zéro ou plusieurs attributs descriptifs.

```
context AggLevel inv:
self.ownedMember->select    (m    |    m.oclIsTypeOf(DescriptiveAttribute))-
>size()>=0
```

c) Un niveau d'agrégation spatial doit avoir exactement un attribut spatial.

```
context SpatialAggLevel inv:
self.ownedMember->select (m | m.oclIsTypeOf(LevelGeometry))->size()=1
```

d) Un niveau d'agrégation temporel doit avoir exactement un attribut temporel.

```
context TemporalAggLevel inv:
self.ownedMember->select (m | m.oclIsTypeOf(LevelTemporality))->size()=1
```

e) Un niveau d'agrégation thématique ne doit pas voir d'attributs stéréotypés LevelGeometry ou LevelTemporality.

```
context ThematicAggLevel inv:
self.ownedMember->select (m | m.oclIsTypeOf(LevelGeometry) or
m.oclIsTypeOf(LevelTemporality))->isEmpty()
```

1.6 Attribut de dimension

a) Un identifiant doit avoir un type de données alphanumérique.

```
context IdAttribute inv:
self.type.oclIsKindOf(NumericalDType)or self.type.oclIsKindOf(TextualDType)
```

b) Un attribut descriptif peut avoir n'importe quel type de données.

c) Un attribut de dimension stéréotypé LevelGeometry doit avoir un type de données spatial.

```
context LevelGeometry inv:
self.type.oclIsKindOf(SpatialDType)
```

d) Un attribut de dimension stéréotypé LevelTemporality doit avoir un type de données temporel.

```
context LevelTemporality inv:
self.type.oclIsKindOf(TemporalDType)
```

1.7 Relation d'agrégation

a) Une relation d'agrégation doit lier deux niveaux d'agrégation dans une hiérarchie de dimension.

```
context AggRelationship inv:
self.memberEnd->size() = 2 and
self.memberEnd->forAll(m|m.oclIsKindOf(AggLevel))
```

1.8 Classe de fait

a) Une classe de fait doit appartenir à un hypercube.

```
context Fact inv:
self.owner.oclIsTypeOf(Hypercube)
```

b) Une classe de fait peut avoir zéro ou plusieurs mesures.

```
context Fact inv:
self.ownedMember->select (m | m.oclIsKindOf(Measure))->size()>=0
```

c) Un fait ne peut être lié à un même niveau d'agrégation que par au plus une relation de dimensionnement.

```
context Fact inv:
self.ownedEnd->select (m | m.oclIsKindOf(AggLevel))->collect(m.type)->
asSet()->size() = self.ownedEnd->select (m | m.oclIsKindOf(AggLevel))-
>size()
```

d) La classe de fait doit être liée à toutes les dimensions de l'hypercubes par des relations de dimensionnement (au moins une relation pour chaque dimension).

```
context Fact inv:
self.ownedEnd->select (m | m.oclIsKindOf(AggLevel))-
>collect(m.hierarchy.dimension)-> asSet()->size() = self.owner.
ownedMember->select (m | m.oclIsKindOf(Dimension))->size()
```

1.9 Mesure

a) Le type additif d'une mesure doit être "stock", "flow" ou "value per unit".

```
context Measure inv:
addType= AdditivityType::Flow or addType= AdditivityType::Stock or addType=
AdditivityType::ValuePerUnit
```

b) Une mesure numérique doit avoir un type de données numérique.

```
context NumericMeasure inv:
self.type.oclIsKindOf(NumericDType)
```

c) Une mesure spatiale doit avoir un type de données spatial.

```
context SpatialMeasure inv:
self.type.oclIsKindOf(SpatialDType)
```

d) Une mesure temporelle doit avoir un type de données temporel.

```
context TemporalMeasure inv:
self.type.oclIsKindOf(TemporalDType)
```

e) Une mesure textuelle doit avoir un type de données textuel.

```
context TextualMeasure inv:
self.type.oclIsKindOf(TextualDType)
```

f) Une mesure booléenne doit avoir un type de données booléen.

```
context BooleanMeasure inv:
self.type.name = 'Boolean'
```

g) Les mesures à différentes granulites doivent avoir une fonction de désagrégation si leur fonction d'agrégation ETL n'est pas la fonction Sum.

```
context Measure inv:
if self.owingFact.owingHypercube.ownedIndicator->exists(i|
    i.oclIsTypeOf(BaseIndicator) and
    i.aggregateAttribute = self and
    i.ownedAggRule->exists(r| r.aggregator <> self.ETLOperation))
then not self.disaggregator.oclIsUndefined()
```

h) Chaque mesure de l'hypercube doit avoir au moins un indicateur de base (cf. Section 2) qui est lui associé.

```
context Measure inv:
self.owingFact.owingHypercube.ownedIndicator->select(i|
i.oclIsTypeOf(BaseIndicator))->exists(i|i.aggregatedAttribute = self)
```

1.10 Relation de dimensionnement

a) Une relation de dimensionnement doit lier une classe de fait à un niveau d'agrégation dans un hypercube

```
context DimRelationship inv:
self.memberEnd->size() = 2 and
self.memberEnd->exits(m|m.oclIsKindOf(AggLevel))and
self.memberEnd->exits(m|m.oclIsTypeOf(Fact))
```

2 Contraintes du métamodèle d'agrégation

Le métamodèle d'agrégation (cf. Figure 2) permet de représenter comment les mesures doivent être agrégées pour répondre aux besoins des décideurs en termes d'analyse. Il permet de spécifier les structures multidimensionnelles d'agrégation du modèle d'analyse SOLAP (e.g.

indicateurs, règles d'agrégation, etc.). Ces structures sont nécessaires pour la spécification des CI SOLAP d'agrégation et de requêtes.

Nous présentons ici les principales contraintes associées à chacun des stéréotypes définis par ce métamodèle.

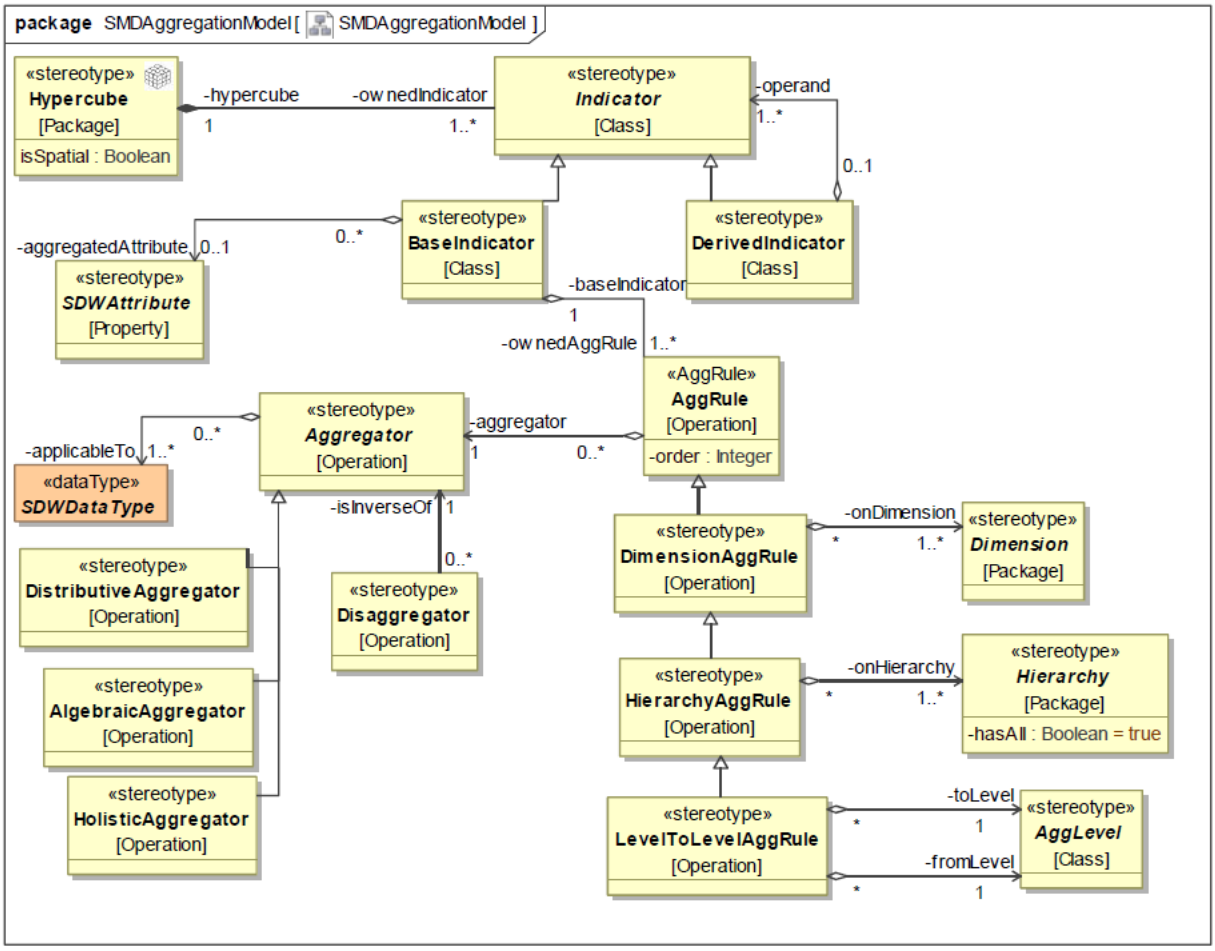


Figure 2. Partie métamodèle d'agrégation du profil UML.

2.1 Indicateur d'analyse

a) Un indicateur d'analyse doit appartenir à un seul hypercube.

```
context Indicator inv:
self.owner.oclIsTypeOf(Hypercube)
```

b) Un indicateur peut être de base ou dérivé.

```
context Indicator inv:
self.oclIsTypeOf(BaseIndicator)xor self.oclIsTypeOf(DerivedIndicator)
```

c) Un indicateur de base doit porter sur zéro ou une mesure; le zéro est utile dans le cas d'un comptage des instances de fait.

```
context BaseIndicator inv:
self.aggregatedAttribute->oclIsUndefined()                    xor
self.aggregatedAttribute.oclIsKindOf(Measure)
```

d) Un indicateur de base doit avoir au moins une règle d'agrégation.

```
context BaseIndicator inv:
self.ownedMember->select (m | m.oclIsKindOf(AggRule))->size()>=1
```

e) Un indicateur de base contient au maximum une seule règle d'agrégation de type AggRule.

```
context BaseIndicator inv:
self.ownedMember->select (m | m.oclIsTypeOf(AggRule))->size()<=1
```

f) Un indicateur de base contient au maximum une seule règle d'agrégation de type DimensionAggRule pour chaque dimension.

```
context BaseIndicator inv:
self.ownedMember->select (m | m.oclIsTypeOf(DimensionAggRule))-> forAll(r1,
r2| (r1 != r2) implies  r1.onDimension != r2.onDimension)
```

g) Un indicateur de base contient au maximum une seule règle d'agrégation de type HierarchyAggRule pour chaque hiérarchie.

```
context BaseIndicator inv:
self.ownedMember->select (m | m.oclIsTypeOf(HierarchyAggRule))-> forAll(r1,
r2| (r1 != r2) implies r1.onHierarchy != r2. onHierarchy)
```

h) Un indicateur de base contient au maximum une seule règle d'agrégation de type LevelToLevelAggRule entre deux niveaux identiques.

```
context BaseIndicator inv:
self.ownedMember->select    (m    |    m.oclIsTypeOf(LevelToLevelAggRule))->
forAll(r1,  r2|  (r1  !=  r2)  implies  ((r1.fromLevel  !=  r2.fromLevel)  and
(r1.toLevel != r2. toLevel)) )
```

i) Un indicateur dérivé doit avoir une formule de dérivation de type OCL OpaqueExpression.

```
context DerivedIndicator inv:
self.ownedAttribute->select(a|     a.name='formula'     and     a.type.name=
'OpaqueExpression')->size()=1
```

j) Dans un indicateur de base, les fonctions d'agrégation Avg, Max et Min doivent être appliquées après la fonction d'agrégation Sum.

```
context BaseIndicator
def: SumRules:Set{AggRule} = self.ownedAggRule->select(r|
r.aggregator.name = 'Sum')
def: AvgMaxMinRules:Set{AggRule}= self.ownedAggRule->select(r|
r.aggregator.name = 'Avg' or r.aggregator.name = 'Max' or
r.aggregator.name = 'Min')
inv SumBeforeAvgMinMax: SumRules->forAll(rSum|
AvgMaxMinRules->forAll(rAvgMaxMin| rSum.order < rAvgMaxMin.order))
```

2.2 Règle d'agrégation

a) Une règle d'agrégation doit appartenir à un indicateur de base.

```
context Indicator inv:
self.owner.oclIsTypeOf(BaseIndicator)
```

b) Une règle d'agrégation de type DimensionAggRule doit être définie sur un ensemble non vide de dimensions de l'hypercube.

```
context DimensionAggRule inv:
self.onDimension-> notEmpty() and
self.indicator.hypercube.ownedMember->select              (m              |
m.oclIsKindOf(Dimension))->includesAll(self.onDimension)
```

c) Une règle d'agrégation de type HierarchyAggRule doit être définie sur un ensemble non vide de hiérarchies de dimension de l'hypercube.

```
context HierarchyAggRule inv:
self.onHierarchy-> notEmpty() and
self.indicator.hypercube.ownedDimension.ownedMember->select (m |
m.oclIsKindOf(Hierachy))->includesAll(self. onHierarchy)
```

d) Une règle d'agrégation de type LevelToLevelAggRule doit être définie sur deux niveaux différents appartenant à une même hiérarchie dimension.

```
context LevelToLevelAggRule inv:
(self.fromLevel-> notEmpty() and self.toLevel-> notEmpty())
(self.fromLevel != self.toLevel)
(self.fromLevel.owner = self.toLevel.owner)
```

e) Compatibilité de types entre l'attribut agrégée et la fonction d'agrégation utilisée. La fonction d'agrégation doit être applicable au type de données de l'attribut agrégé.

```
context AggRule inv :
not(baseIndicator.aggregatedAttribute.OclIsUndefined()) implies
aggregator.applicableTo->exists(dt|
  baseIndicator.aggregatedAttribute.oclIsTypeOf(dt))
```

f) Dans le cas de fait sans mesures, les instances de fait ne peuvent être comptées par la fonction d'agrégation Count ou DistinctCount.

```
context AggRule inv:
if (baseIndicator.aggregatedAttribute->oclIsUndefined())
then (aggregator.name = 'Count' or aggregator.name = 'DistinctCount')
```

g) Les mesures non additives de type valeur par unité ne peuvent pas être sommées.

```
context AggRule inv:
if (baseIndicator.aggregatedAttribute.OclIsKindOf(Measure) and
baseIndicator.aggregatedAttribute.addType = AdditivityType::ValuePerUnit)
then aggregator.name <> 'Sum'
```

h) Les mesures semi-additives de type stock ne peuvent pas être sommées selon la dimension temporelle.

```
context DimensionAggRule inv:
if (baseIndicator.aggregatedAttribute.OclIsKindOf(Measure)and
baseIndicator.aggregatedAttribute.addType = AdditivityType::Stock and
onDimension->exists(d|d.oclIsTypeOf(TemporalDimension)))
then aggregator.name <>'Sum'
```

Références

Abdullah, A., S. Brobst, M. Umer and M. F. Khan (2004). The Case for an Agri Data Warehouse: Enabling Analytical Exploration of Integrated Agricultural Data. Databases and Applications**:** 139-144.

Abelló, A., J. Samos and F. Saltor (2006). "YAM^2: a multidimensional conceptual model extending UML." Information Systems **31**(6): 541-567.

Adamson, C. (2006). Mastering Data Warehouse Aggregates: Solutions for Star Schema Performance, John Wiley & Sons.

Ahmed, T. O. and M. Miquel (2005). Multidimensional Structures Dedicated to Continuous Spatiotemporal Phenomena. BNCOD**:** 29-40.

Akehurst, D. H. and B. Bordbar (2001). On Querying UML Data Models with OCL. Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools, Springer-Verlag**:** 91-103.

Ali, W., B. Moulin, Y. Bédard, M. J. Proulx and S. Rivest (2007). "coupling Multiagent geosimulation and Spatial olap for Better geosimulation data analysis." URISA-WASHINGTON DC- **19**(2): 23.

Allen, J. F. (1983). "Maintaining knowledge about temporal intervals." Commun. ACM **26**(11): 832-843.

Bailey, A. P., W. D. Basford, N. Penlington, J. R. Park, J. D. H. Keatinge, T. Rehman, R. B. Tranter and C. M. Yates (2003). "A comparison of energy use in conventional and integrated arable farming systems in the UK." Agriculture, Ecosystems & Environment **97**(1–3): 241-253.

Basta, A. and M. Zgola (2011). Database Security, Delmar Cengage Learning.

Batini, C. and M. Scannapieca (2006). Data quality: Concepts, methodologies and techniques, Springer-Verlag New York Inc.

Bédard, Y. (1997). Spatial OLAP.

Bédard, Y. (2009). "Web site "Spatial OLAP"." <http://www.spatialbi.com/>.

Bédard, Y., E. Bernier, S. Larrivée, M. Nadeau, M.-J. Proulx and S. Rivest. (2009). "Spatial OLAP." from http://www.spatialbi.com.

Bédard, Y. and J. Han (2009). Fundamentals of Spatial Data Warehousing for Geographic Knowledge Discovery. Geographic Data Mining and Knowledge Discovery. H. J. Miller and J. Han, Taylor & Francis.

Bédard, Y., S. Larrivee, M. J. Proulx and M. Nadeau (2004). "Modeling geospatial databases with plug-ins for visual languages: A pragmatic approach and the impacts of 16 years of research and experimentations on perceptory." Lecture Notes in Computer Science **3289**: 17-30.

Bédard, Y., T. Merrett and J. Han (2001). "Fundamentals of spatial data warehousing for geographic knowledge discovery." Geographic data mining and knowledge discovery **2**.

Bédard, Y., S. Rivest and M.-j. Proulx (2006). Spatial On-Line Analytical Processing (SOLAP): Concepts, Architectures, and Solutions from a Geomatics Engineering Perspective. Data Warehouses and OLAP: Concepts, Architecture, and Solutions, Idea Group Publishing.

Bédard, Y., S. Rivest and M. Proulx (2007). "Spatial. Online. Analytical. Processing.(SOLAP): Concepts,. Architectures,. and. Solutions. from." Data warehouses and OLAP: concepts, architectures, and solutions: 298.

Bejaoui, L. (2009). Qualitative topological relationships for objects with possibly vague shapes: implications on the specification of topological integrity constraints in

transactional spatial databases and in spatial data warehouses, Université Blaise Pascal.

Bejaoui, L., F. Pinet, Y. Bedard and M. Schneider (2009). "Qualified topological relations between spatial objects with possible vague shape." International Journal of Geographical Information Science **23**(7): 877-921.

Bellatreche, L. (2000). Une méthodologie pour la fragmentation dans les entrepôts de données. INFORSID**:** 246-267.

Bellatreche, L., A. Giacometti, P. Marcel, H. Mouloudi and D. Laurent (2005). A personalization framework for OLAP queries. Proceedings of the 8th ACM international workshop on Data warehousing and OLAP. Bremen, Germany, ACM**:** 9-18.

Bellatreche, L., M. Schneider, H. Lorinquer and M. Mohania (2004). Bringing Together Partitioning, Materialized Views and Indexes to Optimize Performance of Relational Data Warehouses

Data Warehousing and Knowledge Discovery. Y. Kambayashi, M. Mohania and W. Wöß, Springer Berlin / Heidelberg. **3181:** 15-25.

Bernier, E., P. Gosselin, T. Badard and Y. Bédard (2009). "Easier Surveillance Of Climate-Related Health Vulnerabilities Through A Web-Based Spatial Olap Application." International Journal of Health Geographics **8**(18).

Berson, A. and S. Smith (1997). Data Warehousing, Data Mining, and OLAP (Data Warehousing/Data Management), Computing Mcgraw-Hill.

Bimonte, S. (2007). Intégration de l'information géographique dans les entrepôts de données et l'analyse en ligne : de la modélisation à la visualisation, INSA-Lyon**:** 207.

Bimonte, S. (2010). "A Web-Based Tool for Spatio-Multidimensional Analysis of Geographic and Complex Data." International Journal of Agricultural and Environmental Information Systems **1**(2): 42-67.

Bimonte, S. and F. Pinet (2010). "When Spatial Analysis Meets OLAP: Multidimensional Model and Operators." to appear in: International Journal of Data Warehousing and Mining.

Bimonte, S., A. Tchounikine and M. Miquel (2005). Towards a spatial multidimensional model. Proceedings of the 8th ACM international workshop on Data warehousing and OLAP. Bremen, Germany, ACM**:** 39-46.

Bimonte, S., A. Tchounikine and M. Miquel (2007). Spatial OLAP: Open Issues and a Web Based Prototype. International Conference on Geographic Information Science 1-11.

Bimonte, S., M. Villanova-Oliver and J. Gensel (2009). "A Multidimensional Model for Correct Aggregation of Geographic Measures." Evolving Application Domains of Data Warehousing and Mining: Trends and Solutions: 162.

Böhnlein, M., M. Plaha and A. Ulbrich-vom Ende (2002). "Visual Specification of Multidimensional Queries based on a Semantic Data Model." Vom Data Warehouse zum Corporate Knowledge Center (DW): 379-397.

Booch, G., J. Rumbaugh and I. Jacobson (1999). The Unified Modeling Language User Guide, Addison-Wesley.

Bosc, P. and O. Pivert (1995). "SQLf: a relational database language for fuzzy querying." Fuzzy Systems, IEEE Transactions on **3**(1): 1-17.

Boulil, K., S. Bimonte, François Pinet, N. Carluer, C. Lauvernet, B. Cheviron, A. Miralles and J.-P. Chanet (2012a). "Using Data Warehouses for Storage and Visualization of Simulation Model Results." Environmental Modelling & Software: 20.

Boulil, K., S. Bimonte, H. Mahboubi and F. Pinet (2010a). Towards the definition of spatial data warehouses integrity constraints with spatial OCL. Proceedings of the ACM 13th

international workshop on Data warehousing and OLAP. Toronto, ON, Canada, ACM**:** 31-36.

Boulil, K., S. Bimonte, H. Mahboubi and F. Pinet (2010b). "Vers la définition des contraintes d'intégrité d'entrepôts de données spatiales avec OCL." Revue des Nouvelles Technologies de l'Information **B6**: 121-136.

Boulil, K., S. Bimonte and F. Pinet (2011). "Un modèle UML et des contraintes OCL pour les entrepôts de données spatiales. De la représentation conceptuelle à l'implémentation." Ingénierie des Systèmes d'Information **16**(6): 11-39.

Boulil, K., S. Bimonte and F. Pinet (2012b). A UML & Spatial OCL based approach for handling quality issues in SOLAP systems. Proceedings of the14th International Conference on Enterprise Information Systems (ICEIS 2012). Wroclaw, Poland**:** 6.

Boulil, K., S. Bimonte and F. Pinet (2012c). "A UML Profile and OCL-based Constraints for Spatial Data Cubes." Information Systems (submitted).

Boulil, K., S. Bimonte and F. Pinet (2012d). Un cadre conceptuel basé sur UML et Spatial OCL pour la définition des contraintes d'intégrité dans les systèmes SOLAP. Actes des 8èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 2012). Bordeaux, France**:** 15.

Burdick, D., P. M. Deshpande, T. Jayram, R. Ramakrishnan and S. Vaithyanathan (2006). Efficient allocation algorithms for OLAP over imprecise data, VLDB Endowment.

Cabibbo, L. and R. Torlone (2004). On the integration of autonomous data marts. 16th International Conference on Scientific and Statistical Database Management. Santorini Island,Greece**:** 223-234.

Cabot, J., J.-N. Mazón, J. Pardillo and J. Trujillo (2010). Specifying aggregation functions in multidimensional models with OCL. Proceedings of the 29th international conference on Conceptual modeling. Vancouver, BC, Canada, Springer-Verlag**:** 419-432.

Camptocamp. (2012). "Spatial Data Integrator version 1.3.0." from http://www.camptocamp.com/fr/news/89-release-de-spatial-data-integrator-version-130.

Carluer, N., V. Gouy, C. Lauvernet, A. Miralles, F. Pinet, C. Gascuel-Odoux, C. Grimaldi, Y. Coquet, P. Benoit, B. Réal, N. Marquet and C. Guyot (2011). Building risk indicators of surface water contamination by pesticides at the smallcatchment scale. Taking in account spatial and temporal dimensions. Support for risk assessment and management : MIRIPHYQUE project. PEER Euraqua, Montpellier, France.

Carpani, F. and R. Ruggia (2001). An Integrity Constraints Language for a Conceptual Multidimensional Data Model.

Chaker, W., M. Proulx, B. Moulin and Y. Bédard (2009). "Modélisation, Simulation et Analyse d'Environnements Urbains Peuplés." Revue Internationale de Géomatique.

Chaudhary, S., V. Sorathia and Z. Laliwala (2004). Architecture of Sensor based Agricultural Information System for Effective Planning of Farm Activities. Proceedings of the 2004 IEEE International Conference on Services Computing, IEEE Computer Society**:** 93-100.

Chaudhuri, S., U. Dayal and V. Ganti (2001). "Database technology for decision support systems." Computer **34**(12): 48-55.

Chimiak-Opoka, J. D. and C. Lenz (2007). "Use of OCL in a Model Assessment Framework: An experience report." Electronic Communications of the EASST **5**(0).

Choong, Y. W., D. Laurent and P. Marcel (2003). "Computing appropriate representations for multidimensional data." Data Knowl. Eng. **45**(2): 181-203.

Chuffart, F., N. Dumoulin, T. Faure and G. Deffuant (2010). "SimExplorer: programming experimental design on models and managing quality of modelling process."

International Journal of Agricultural and Environmental Information Systems **1**(1): 55-68.

Claramunt, C. (2000). Extending Ladkin's algebra on non-convex intervals towards an algebra on union-of regions. Proceedings of the 8th ACM international symposium on Advances in geographic information systems. Washington, D.C., United States, ACM**:** 9-14.

Clementini, E. and P. D. Felice (1995). "A comparison of methods for representing topological relationships." Inf. Sci. Appl. **3**(3): 149-178.

Clementini, E., P. D. Felice and P. v. Oosterom (1993). A Small Set of Formal Topological Relationships Suitable for End-User Interaction. Proceedings of the Third International Symposium on Advances in Spatial Databases, Springer-Verlag**:** 277-295.

Clements, A. C. A., D. U. Pfeiffer, M. J. Otte, K. Morteo and L. Chen (2002). "A global livestock production and health atlas (GLiPHA) for interactive presentation, integration and analysis of livestock data." Preventive Veterinary Medicine **56**(1): 19-32.

Codd, E. F., S. B. Codd, C. T. Salley, Codd and I. Date (1993). Providing OLAP (On-line Analytical Processing) to User-analysts: An IT Mandate, Codd amp; Associates.

da Silva, J., A. G. de Oliveira, R. N. Fidalgo, A. C. Salgado and V. C. Times (2010). "Modelling and querying geographical data warehouses." Information Systems **35**(5): 592-614.

Damiani, M. L. and S. Spaccapietra (2006). Spatial Data Warehouse Modelling. Processing and Managing Complex Data for Decision Support. J. Darmont and O. Boussaid. Hershey, IGI Global**:** 1-27.

Daniel L, M. (2005). "Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions." Data & Knowledge Engineering **55**(3): 243-276.

Decker, H. and D. Martinenghi (2006). A relaxed approach to integrity and inconsistency in databases. Proceedings of the 13th international conference on Logic for Programming, Artificial Intelligence, and Reasoning. Phnom Penh, Cambodia, Springer-Verlag**:** 287-301.

Demuth, B. and H. Hussmann (1999). Using UML/OCL Constraints for Relational Database Design. «UML»'99 — The Unified Modeling Language**:** 751-751.

Demuth, B., H. Hussmann and S. Loecher (2001). OCL as a Specification Language for Business Rules in Database Applications. «UML» 2001 — The Unified Modeling Language. Modeling Languages, Concepts, and Tools**:** 104-117.

Devillers, R., Y. Bedard, R. Jeansoulin and B. Moulin (2007). "Towards spatial data quality information analysis tools for experts assessing the fitness for use of spatial data." International Journal of Geographical Information Science **21**(3): 261-282.

Devillers, R. and R. Jeansoulin (2005). Qualité de l'information géographique, Hermès Science Publications.

do Nascimento Fidalgo, R., V. C. Times, J. da Silva, F. d. F. de Souza and A. C. Salgado (2010). "Revisiting "Providing Multidimensional and Geographical Integration Based on a GDW and Metamodels"." JIDM **1**(1): 107-110.

Duboisset, M. (2007). Un Système de Contraintes d'Intégrité OCL pour les Bases de Données Spatiales: Application à un Système d'Information pour l'Epandage Agricole, Université Blaise Pascal.

Duboisset, M., F. Pinet, M. A. Kang and M. Schneider (2005). "Precise modeling and verification of topological integrity constraints in spatial databases: from an expressive power study to code generation principles." Lecture Notes in Computer Science (ER) **3716**: 465-482.

Duboisset, M., F. Pinet, M. A. Kang and M. Schneider (2007). "A general framework to implement topological relations on composite regions." Lecture notes in computer science **4653**: 823-833.

Egenhofer, M. and J. Herring (1994). "Categorizing binary topological relations between regions, lines, and points in geographic databases." The **9**: 94-91.

Fernández-Quiruelas, V., J. Fernández, A. S. Cofiño, L. Fita and J. M. Gutiérrez (2011). "Benefits and requirements of grid computing for climate applications. An example with the community atmospheric model." Environmental Modelling & Software **26**(9): 1057-1069.

Fidalgo, R., V. Times, J. Silva and F. Souza (2004). GeoDWFrame: A Framework for Guiding the Design of Geographical Dimensional Schemas. Data Warehousing and Knowledge Discovery. Y. Kambayashi, M. Mohania and W. Wöß, Springer Berlin / Heidelberg. **3181:** 26-37.

Franconi, E. and A. Kamble (2004). The GMD Data Model and Algebra for Multidimensional Information. Advanced Information Systems Engineering. A. Persson and J. Stirna, Springer Berlin / Heidelberg. **3084:** 97-116.

Galhardas, H., D. Florescu, D. Shasha, E. Simon and C.-A. Saita (2001). Declarative Data Cleaning: Language, Model, and Algorithms. Proceedings of the 27th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc.**:** 371-380.

Ghozzi, F., F. Ravat, O. Teste and G. Zurfluh (2003a). Constraints and Multidimensional Databases, Citeseer.

Ghozzi, F., O. Teste and G. Zurfluh (2003b). "Modèle multidimensionnel à contraintes." Revue d'Intelligence Artificielle **17**(1-3): 43-55.

Glorio, O. and J. Trujillo (2008). An MDA Approach for the Development of Spatial Data Warehouses. Data Warehousing and Knowledge Discovery. I.-Y. Song, J. Eder and T. Nguyen, Springer Berlin / Heidelberg. **5182:** 23-32.

Goertzen, R. and J. Stausberg (2007). "A grammar of integrity constraints in medical documentation systems." Computer Methods and Programs in Biomedicine **86**(1): 93-102.

Golfarelli, M., D. Maio and S. Rizzi (1998). Conceptual Design of Data Warehouses from E/R Schema. Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences-Volume 7 - Volume 7, IEEE Computer Society**:** 334.

Golfarelli, M. and S. Rizzi (2009). Data Warehouse Design: Modern Principles and Methodologies, McGraw-Hill.

Gras, R., M. Benoît, J. P. Deffontaines, M. Duru, M. Lafarge, A. Langlet and P. L. Osty (1989). Le Fait technique en agronomie: activité agricole, concepts et méthodes d'étude, Institut national de la Recherche agronomique.

Hackney, D. (2002). "Architectures and Approaches for Successful Data Warehouses." Oracle White Papers **2**.

Hadzilacos, T. and N. Tryfona (1992). A Model for Expressing topological Integrity Constraints in Geographic Databases. Proceedings of the International Conference GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning on Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, Springer-Verlag**:** 252-268.

Hahn, K., C. Sapia and M. Blaschka (2000). Automatically generating OLAP schemata from conceptual graphical models. Proceedings of the 3rd ACM international workshop on Data warehousing and OLAP. McLean, Virginia, United States, ACM**:** 9-16.

Han, J., N. Stefanovic and K. Koperski (1998). Selective materialization: An efficient method for spatial data cube construction. Research and Development in Knowledge Discovery

and Data Mining. X. Wu, R. Kotagiri and K. Korb, Springer Berlin / Heidelberg. **1394:** 144-158.
Hirabayashi, S., C. N. Kroll and D. J. Nowak (2011). "Component-based development and sensitivity analyses of an air pollutant dry deposition model." Environmental Modelling & Software **26**(6): 804-816.
Horner, J. and I.-Y. Song (2005). A taxonomy of inaccurate summaries and their management in OLAP systems. Proceedings of the 24th international conference on Conceptual Modeling. Klagenfurt, Austria, Springer-Verlag**:** 433-448.
Horner, J., I.-Y. Song and P. P. Chen (2004). An analysis of additivity in OLAP systems. Proceedings of the 7th ACM international workshop on Data warehousing and OLAP. Washington, DC, USA, ACM**:** 83-91.
Hornsby, K., M. J. Egenhofer and P. J. Hayes (1999). Modeling Cyclic Change. Proceedings of the Workshops on Evolution and Change in Data Management, Reverse Engineering in Information Systems, and the World Wide Web and Conceptual Modeling, Springer-Verlag**:** 98-109.
Hurtado, C. A., C. Gutierrez and A. O. Mendelzon (2005). "Capturing summarizability with integrity constraints in OLAP." ACM Transactions on Database Systems **30**(3): 854-886.
Hurtado, C. A. and A. O. Mendelzon (2001). Reasoning about Summarizability in Heterogeneous Multidimensional Schemas. Proceedings of the 8th International Conference on Database Theory, Springer-Verlag**:** 375-389.
Husemann, B., J. Lechtenbörger and G. Vossen (2000). Conceptual Data Warehouse Design. In Proc. of the International Workshop on Design and Management of Data Warehouses (DMDW 2000).
Iftikhar, N. and T. Pedersen (2010). Schema Design Alternatives for Multi-granular Data Warehousing. Database and Expert Systems Applications. P. Bringas, A. Hameurlain and G. Quirchmayr, Springer Berlin / Heidelberg. **6262:** 111-125.
Inmon, W. H. (2005). Building the Data Warehouse, 4th Edition, Wiley.
ISO (2000). ISO 9000: Quality Management Systems:Fundamentals and Vocabulary.
ISO (2003). ISO 19115:2003 -- Geographic information -- Metadata.
ISO/IEC (2001). ISO/IEC 9126-1:2001 -- Software engineering -- Product quality -- Part 1: Quality model.
ISO/IEC (2003). ISO/IEC TR 9126-3:2003 -- Software engineering -- Product quality -- Part 3: Internal metrics.
ISO/IEC (2004). ISO/IEC TR 9126-4:2004 -- Software engineering -- Product quality -- Part 4: Quality in use metrics.
ISO/TC 211 (2002). ISO 19113:2002: Geographic information -- Quality principles
Jarke, M., M. Lenzerini, Y. Vassiliou and P. Vassiliadis (2003). Fundamentals of data warehouses, Springer-Verlag New York Inc.
Jeltsch, F., S. J. Milton, W. R. J. Dean and N. Van Rooyen (1996). "Tree spacing and coexistence in semiarid savannas." Journal of Ecology **84**(4): 583-595.
Jeltsch, F., S. J. Milton, W. R. J. Dean, N. Van Rooyen and K. A. Moloney (1998). "Modelling the impact of small-scale heterogeneities on tree-grass coexistence in semi-arid savannas." Journal of Ecology **86**(5): 780-793.
Jeltsch, F., K. A. Moloney and S. J. Milton (1999). "Detecting Process from Snap-Shot Pattern: Lessons from Tree Spacing in the Southern Kalahari." Oikos **85**.
Jensen, C. S., A. Kligys, T. B. Pedersen and I. Timko (2004). "Multidimensional data modeling for location-based services." The VLDB Journal **13**(1): 1-21.
JRubik. (2012). "Introduction to JRubik." from http://rubik.sourceforge.net/jrubik/intro.html.

Kamble, A. S. (2008). A conceptual model for multidimensional data. Proceedings of the fifth Asia-Pacific conference on Conceptual Modelling - Volume 79. Wollongong, NSW, Australia, Australian Computer Society, Inc.**:** 29-38.

Kelly, S. (2007). Data warehousing in action, Wiley-India.

Kimball, R. (1996). The data warehouse toolkit: practical techniques for building dimensional data warehouses, John Wiley & Sons.

Kimball, R. and M. Ross (2002). The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd Edition, Wiley.

Klasse Objecten. (2009). "OCL Tools and Services Web Site. http://www.klasse.nl/ocl." from http://www.klasse.nl/ocl.

Kotidis, Y. and N. Roussopoulos (1998). "An alternative storage organization for ROLAP aggregate views based on cubetrees." SIGMOD Rec. **27**(2): 249-258.

Kozmina, N. and L. Niedrite (2011). "Research Directions of OLAP Personalizaton." Information Systems Development: 345-356.

Larsbo, M. and N. J. Jarvis (2003). MACRO 5.0. A model of water flow and solute transport in macroporous soil. Technical description. Studies in the Biogeophysical Environment.

Lechtenbörger, J. and G. Vossen (2003). "Multidimensional normal forms for data warehouse design." Inf. Syst. **28**(5): 415-434.

Lehner, W., J. Albrecht and H. Wedekind (1998). Normal Forms for Multidimensional Databases. Proceedings of the 10th International Conference on Scientific and Statistical Database Management, IEEE Computer Society**:** 63-72.

Lenz, H.-J. and A. Shoshani (1997). Summarizability in OLAP and statistical data bases, IEEE.

Levesque, M., Y. Bédard, M. Gervais and R. Devillers (2007). Towards managing the risks of data misuse for spatial datacubes.

Li, Z. and X.-z. Mao (2011). "Global multiquadric collocation method for groundwater contaminant source identification." Environmental Modelling & Software **26**(12): 1611-1621.

Liu, J., S. Liang, D. Ye, J. Wei and T. Huang (2009). ETL Workflow Analysis and Verification Using Backwards Constraint Propagation. Proceedings of the 21st International Conference on Advanced Information Systems Engineering. Amsterdam, The Netherlands, Springer-Verlag**:** 455-469.

Lujan-Mora, S., J. Trujillo and I.-Y. Song (2006). "A UML profile for multidimensional modeling in data warehouses." Data & Knowledge Engineering **59**(3): 725-769.

Ma, H., K.-D. Schewe and B. Thalheim (2009). Geometrically Enhanced Conceptual Modelling

Conceptual Modeling - ER 2009. A. Laender, S. Castano, U. Dayal, F. Casati and J. de Oliveira, Springer Berlin / Heidelberg. **5829:** 219-233.

MacEachren, A. M., M. Gahegan, W. Pike, I. Brewer, G. Cai, E. Lengerich and F. Hardisty (2004). "Geovisualization for Knowledge Construction and Decision Support." IEEE Comput. Graph. Appl. **24**(1): 13-17.

Mahboubi, H., S. Bimonte and G. Deffuant (2011). Analyzing demographic and economic simulation model results: a semi-automatic spatial OLAP approach. Proceedings of the 2011 international conference on Computational science and its applications - Volume Part I. Santander, Spain, Springer-Verlag**:** 17-31.

Mahboubi, H., S. Bimonte, T. Faure and F. Pinet (2010). "Data warehouse and OLAP for Environmental Simulation Data." International Journal of Agricultural and Environmental Systems **1**(2).

Malinowski, E. and E. Zimanyi (2006). "Hierarchies in a rnultidimensional model: From conceptual modeling to logical representation." Data & Knowledge Engineering **59**(2): 348-377.
Malinowski, E. and E. Zimanyi (2008). Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications, Springer.
Malinowski, E. and E. Zimányi (2004a). OLAP Hierarchies: A Conceptual Perspective
Advanced Information Systems Engineering. A. Persson and J. Stirna, Springer Berlin / Heidelberg. **3084:** 19-35.
Malinowski, E. and E. Zimányi (2004b). Representing spatiality in a conceptual multidimensional model. GIS**:** 12-22.
Malinowski, E. and E. Zimányi (2005). Spatial Hierarchies and Topological Relationships in the Spatial MultiDimER Model. BNCOD**:** 17-28.
Malinowski, E. and E. Zimányi (2006). "Hierarchies in a multidimensional model: From conceptual modeling to logical representation." Data & Knowledge Engineering **59**(2): 348-377.
Malinowski, E. and E. Zimányi (2008). "Advanced Data Warehouse Design." Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications, Data-Centric Systems and Applications, Volume. ISBN 978-3-540-74404-7. Springer Berlin Heidelberg, 2008 **1**.
Mandel, L. and M. Cengarle (1999). On the Expressive Power of OCL
FM'99 — Formal Methods. J. Wing, J. Woodcock and J. Davies, Springer Berlin / Heidelberg. **1708:** 713-713.
Mansmann, S. and M. H. Scholl (2006). Extending visual OLAP for handling irregular dimensional hierarchies. Proceedings of the 8th international conference on Data Warehousing and Knowledge Discovery. Krakow, Poland, Springer-Verlag**:** 95-105.
Martin, R. (2008). Data Warehouse 100 Success Secrets - 100 most Asked questions on Data Warehouse Design, Projects, Business Intelligence, Architecture, Software and Models, Emereo Pty Ltd.
Mazón, J.-N., J. Lechtenbörger and J. Trujillo (2008). Solving summarizability problems in fact-dimension relationships for multidimensional models. Proceedings of the ACM 11th international workshop on Data warehousing and OLAP. Napa Valley, California, USA, ACM**:** 57-64.
Mazón, J.-N., J. Lechtenbörger and J. Trujillo (2009). "A survey on summarizability issues in multidimensional modeling." Data & Knowledge Engineering **68**(12): 1452-1469.
Mazon, J.-N. and J. Trujillo (2008). "An MDA approach for the development of data warehouses." Decis. Support Syst. **45**(1): 41-58.
Mazón, J.-N., J. Trujillo and J. Lechtenbörger (2006). A set of QVT relations to assure the correctness of data warehouses by using multidimensional normal forms. Proceedings of the 25th international conference on Conceptual Modeling. Tucson, AZ, Springer-Verlag**:** 385-398.
Mazón, J.-N., J. Trujillo and J. Lechtenbörger (2007). "Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms." Data Knowl. Eng. **63**(3): 725-751.
McHugh, R., S. Roche and Y. Bédard (2007). Vers une solution SOLAP comme outil participatif. Comptesrendus de la Conférence québéco-française pour le développement de la géomatique-CQFD.
Microsoft. (2012). "Multidimensional Expressions (MDX) Reference." from http://msdn.microsoft.com/en-us/library/ms145506.aspx.

Miralles, A., F. Pinet, N. Carluer, F. Vernier, S. Bimonte, C. Lauvernet and V. Gouy (2011). EIS pesticide: an information system for data and knowledge capitalization and analysis. PEER Euraqua, Montpellier, France.

Mitchell, G., A. May and A. McDonald (1995). "PICABUE: a methodological framework for the development of indicators of sustainable development." International Journal of Sustainable Development & World Ecology **2**(2): 104-123.

Mũoz, L., J. N. Mazón and J. Trujillo (2009). Automatic generation of ETL processes from conceptual models.

Negre, E. (2011). "Quand la recommandation rencontre la personnalisation. Ou comment générer des recommandations (requêtes MDX) en adéquation avec les préférences de l'utilisateur."

Nguyen, T., A. Tjoa and R. Wagner (2000a). Conceptual Multidimensional Data Model Based on MetaCube. Advances in Information Systems. T. Yakhno, Springer Berlin / Heidelberg. **1909:** 24-33.

Nguyen, T. B., A. M. Tjoa and R. Wagner (2000b). An Object Oriented Multidimensional Data Model for OLAP. Proceedings of the First International Conference on Web-Age Information Management, Springer-Verlag**:** 69-82.

Niedrite, L., M. S. Treimanis and L. Grundmane (2009). Development of Data Warehouse Conceptual Models: Method Engineering Approach. Progressive Methods in Data Warehousing and Business Intelligence: Concepts and Competitive Analytics, IGI Global.

Niemi, T., J. Nummenmaa and P. Thanisch (2001). Logical Multidimensional Database Design for Ragged and Unbalanced Aggregation Hierarchies. in Proceedings of 3rd International Workshop on Design and Management of Data Warehouses**:** 7.

Nilakanta, S., K. Scheibe and A. Rai (2008a). "Dimensional issues in agricultural data warehouse designs." Computers and Electronics in Agriculture **60**(2): 263-278.

Nilakanta, S., K. Scheibe and A. Rai (2008b). "Dimensional issues in agricultural data warehouse designs." Comput. Electron. Agric. **60**(2): 263-278.

No Magic. (2012). "MagicDraw web site ", from https://www.magicdraw.com.

Olivé, A. (2006). "A method for the definition of integrity constraints in object-oriented conceptual modeling languages." Data and Knowledge Engineering **59**(3 SPEC. ISS.): 559-575.

OMG (2003a). Common Warehouse Metamodel (CWM) Specification, Version 1.1, Volume 1, Object Management Group.

OMG (2003b). MDA Guide, *Version 1.0.1*, Object Management Group.

OMG (2006). Object Constraint Language OMG, Available Specification, *Version 2.0*, Object Management Group.

OMG (2007). MOF 2.0/XMI Mapping, Version 2.1.1, Object Management Group.

OMG (2009). Production Rule Representation (PRR), *Version 1.0*, Object Management Group.

OMG (2011a). Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, Object Management Group.

OMG (2011b). OMG Meta Object Facility (MOF) Core Specification - Version 2.4.1, Object Management Group.

OMG (2011c). Unified Modeling Language™ (OMG UML), Infrastructure, *Version 2.4*, Object Management Group.

Oracle. (2012). "Oracle Spatial & Oracle Locator - Location Features for Oracle Database 11g." from http://www.oracle.com/technetwork/database/options/spatial/index.html.

Papajorgji, P., F. Pinet, A. Miralles, E. Jallas and P. M. Pardalos (2010). "Modeling: A Central Activity for Flexible Information Systems Development in Agriculture and Environment." 1-25.

Pardillo, J., J.-N. Mazon and J. Trujillo (2010a). "Designing OLAP schemata for data warehouses from conceptual models with MDA." Decision Support Systems **In Press**: 1-13.

Pardillo, J., J.-N. Mazón and J. Trujillo (2010b). "Extending OCL for OLAP querying on conceptual multidimensional models of data warehouses." Information Sciences **180**(5): 584-601.

Parent, C., S. Spaccapietra and E. Zimányi (1999). Spatio-temporal conceptual models: data structures + space + time. Proceedings of the 7th ACM international symposium on Advances in geographic information systems. Kansas City, Missouri, United States, ACM**:** 26-33.

Park, B.-K., H. Han and I.-Y. Song (2005). XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses. Data Warehousing and Knowledge Discovery. A. Tjoa and J. Trujillo, Springer Berlin / Heidelberg. **3589:** 32-42.

Pedersen, T. B. and C. S. Jensen (2001). "Multidimensional database technology." Computer **34**(12): 40-46.

Pedersen, T. B., C. S. Jensen and C. E. Dyreson (1999). Extending Practical Pre-Aggregation in On-Line Analytical Processing. Proceedings of the 25th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc.**:** 663-674.

Pedersen, T. B., C. S. Jensen and C. E. Dyreson (2001). "A foundation for capturing and querying complex multidimensional data." Inf. Syst. **26**(5): 383-423.

Pedersen, T. B. and N. Tryfona (2001). Pre-aggregation in Spatial Data Warehouses. Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases, Springer-Verlag**:** 460-480.

Pentaho. (2012a). "Pentaho Analysis Services (Mondrian)." from http://mondrian.pentaho.com.

Pentaho. (2012b). "Pentaho Mondrian Project." from http://mondrian.pentaho.com.

Pervanchon, F., C. Bockstaller and P. Girardin (2002). "Assessment of energy use in arable farming systems by means of an agro-ecological indicator: the energy indicator." Agricultural Systems **72**(2): 149-172.

Pestana, G., M. M. da Silva and Y. Bedard (2005). Spatial OLAP modeling: an overview base on spatial objects changing over time. Computational Cybernetics, 2005. ICCC 2005. IEEE 3rd International Conference on.

Pinet, F. (2010). HDR - Modélisation des contraintes d'intégrité dans les systèmes d'information environnementaux**:** 112.

Pinet, F., M. Duboisset and V. Soulignac (2007). "Using UML and OCL to maintain the consistency of spatial data in environmental information systems." Environmental modelling & software **22**(8): 1217-1220.

Pinet, F., A. Miralles, S. Bimonte, F. Vernier, N. Carluer, V. Gouy and S. Bernard (2010). The use of UML to design agricultural data warehouses. International Conference on Agricultural Engineering, AGENG 2010.

Pinet, F. and M. Schneider (2009). "A Unified Object Constraint Model for Designing and Implementing Multidimensional Systems." Journal on Data Semantics **13**: 37-71.

Pinet, F. and M. Schneider (2010). "Precise design of environmental data warehouses." Operational Research **10**(3): 349-369.

Pizano, A., A. Klinger and A. Cardenas (1989). "Specification of spatial integrity constraints in pictorial databases." Computer **22**(12): 59-71.

Pogson, M., A. Hastings and P. Smith (2012). "Sensitivity of crop model predictions to entire meteorological and soil input datasets highlights vulnerability to drought." Environmental Modelling & Software **29**(1): 37-43.

Pokorný, J. (2006). "Database architectures: Current trends and their relationships to environmental data management." Environmental Modelling & Software **21**(11): 1579-1586.

PostgreSQL. (2012). "Web site PostgreSQL ", from http://www.postgresql.org.

Pradel, M. and D. Boffety (2012). "Quels indicateurs et solutions technologiques adaptés pour évaluer finement les performances énergétiques des exploitations agricoles?" Sciences Eaux et Territoires(7): 16-29.

Prat, N. and J. Akoka (2002). "From UML to ROLAP multidimensional databases using a pivot model." 18th journées Bases de données avancées (BDA02), Évry, France: 171-195.

Prat, N., J. Akoka and I. Comyn-Wattiau (2006). "A UML-based data warehouse design method." Decis. Support Syst. **42**(3): 1449-1473.

Prat, N., I. Comyn-Wattiau and J. Akoka (2011). "Combining objects with rules to represent aggregation knowledge in data warehouse and OLAP systems." Data and Knowledge Engineering.

Prat, N., I. Wattiau and J. Akoka (2010). Representation of aggregation knowledge in OLAP systems. The 18th European Conference on Information Systems.

Proulx, M. J., E. Bernier, Y. Bédard and P. Gosselin (2007). "Revue systématique en santé environnementale."

Rafanelli, M. and A. Shoshani (1990). STORM: a statistical object representation model. Proceedings of the fifth international conference on Statistical and scientific database management. Charlotte, North Carolina, United States, Springer-Verlag New York, Inc.**:** 14-29.

Rai, A., V. Dubey, K. K. Chaturvedi and P. K. Malhotra (2008). "Design and development of data mart for animal resources." Computers and Electronics in Agriculture **64**(2): 111-119.

Ravat, F., O. Teste and R. Tournier (2007). OLAP aggregation function for textual data warehouse. 9th International Conference on Enterprise Information Systems, Funchal, Madeira - Portugal.

Ravat, F., O. Teste, R. Tournier and G. Zurfluh (2008). Top_Keyword: An Aggregation Function for Textual Document OLAP. Data Warehousing and Knowledge Discovery. I.-Y. Song, J. Eder and T. Nguyen, Springer Berlin / Heidelberg. **5182:** 55-64.

Ravat, F., O. Teste, R. Tournier and G. Zurfluh (2010). Algebraic and Graphic Languages for OLAP Manipulations. Strategic Advancements in Utilizing Data Mining and Warehousing Technologies**:** 60-90.

Ribeiro, L. d. S., R. R. Goldschmidt, M. Cláudia and M. C. Cavalcanti (2011). Complementing data in the ETL process. Proceedings of the 13th international conference on Data warehousing and knowledge discovery. Toulouse, France, Springer-Verlag**:** 112-123.

Richards, L. A. (1931). "Capillary conduction of liquids through porous mediums." Journal of Applied Physics **1**(5): 318-333.

Rivest, S. (2000). "Investigation des modes d'intégration physique entre un serveur de base de données multidimensionnelle et un SIG." Rapport de DEA Informatique, Laval: Université Laval, Canada.

Rivest, S., Y. Bedard, M. Proulx, M. Nadeau, F. Hubert and J. Pastor (2005). "SOLAP technology: Merging business intelligence with geospatial technology for interactive spatio-temporal exploration and analysis of data." ISPRS Journal of Photogrammetry and Remote Sensing **60**(1): 17-33.

Rivest, S., Y. Bédard, M. J. Proulx and M. Nadeau (2003). SOLAP: a new type of user interface to support spatio-temporal multidimensional data exploration and analysis.

Rizzi, S., A. Abelló, J. Lechtenbörger and J. Trujillo (2006). Research in data warehouse modeling and design: dead or alive? Proceedings of the 9th ACM international workshop on Data warehousing and OLAP. Arlington, Virginia, USA, ACM**:** 3-10.

Roussey, C. and F. Pinet (2010). DL based automated consistency checking of spatial relationships. International Conference on Spatial Analysis and GEOmatics (SAGEO 2010°, Toulouse, France.

Rubik. (2012). "Introduction to JRubik." from http://rubik.sourceforge.net/jrubik/intro.html.

Ruiz, C. V. and V. C. Times (2009). "A Taxonomy of SOLAP Operators." XXIV Simpósio Brasileiro de Banco de Dados, Fortaleza, CE.

Salehi, M. (2009). Developing a Model and a Language to Identify and Specify the Integrity Constraints in Spatial Datacubes. Laval, Faculté des études supérieures de l'Université Laval**:** 2002.

Salehi, M., Y. Bédard, M. Mostafavi and J. Brodeur (2007). On Languages for the Specification of Integrity Constraints in Spatial Conceptual Models. Advances in Conceptual Modeling – Foundations and Applications. J.-L. Hainaut, E. Rundensteiner, M. Kirchberget al, Springer Berlin / Heidelberg. **4802:** 388-397.

Salehi, M., Y. Bédard and S. Rivest (2010). " Formal Conceptual Model and Definition Framework for Spatial Datacubes." Geomatica **64**(3): 313-326.

Salka, C. (1998). Ending the ROLAP/MOLAP debate: usage based aggregation and flexible HOLAP. Data Engineering, 1998. Proceedings., 14th International Conference on.

Sampaio, M. C., A. G. d. Sousa and C. d. S. Baptista (2006). Towards a logical multidimensional model for spatial data warehousing and OLAP. Proceedings of the 9th ACM international workshop on Data warehousing and OLAP. Arlington, Virginia, USA, ACM**:** 83-90.

Sapia, C. (1999). On modeling and predicting query behavior in OLAP systems.

Sapia, C., M. Blaschka, G. Höfling and B. Dinter (1998). Extending the E/R Model for the Multidimensional Paradigm. Proceedings of the Workshops on Data Warehousing and Data Mining: Advances in Database Technologies, Springer-Verlag**:** 105-116.

Sboui, T., M. Salehi and Y. Bédard (2010). "A systematic approach for managing the risk related to semantic interoperability between geospatial datacubes." International Journal of Agricultural and Environmental Information Systems **1**(2): 20-41.

Schulze, C., J. Spilke and W. Lehner (2007a). "Data modeling for Precision Dairy Farming within the competitive field of operational and analytical tasks." Computers and Electronics in Agriculture **59**(1): 39-55.

Schulze, C., J. Spilke and W. Lehner (2007b). "Data modeling for Precision Dairy Farming within the competitive field of operational and analytical tasks." Computers and Electronics in Agriculture **59**(1–2): 39-55.

Shekhar, S., C. T. Lu, S. Chawla and R. R. Vatsavai (2001). Map Cube: A Visualization Tool for Spatial Data Warehouses.

Silva, J. d., V. C. Times, A. C. Salgado, C. Souza, R. d. N. Fidalgo and A. G. d. Oliveira (2008). A set of aggregation functions for spatial measures. Proceeding of the ACM 11th international workshop on Data warehousing and OLAP. Napa Valley, California, USA, ACM**:** 25-32.

Singh, M., S. Parvinder and Suman (2007). "Conceptual Multidimensional Model." World Academy of Science, Engineering and Technology 6.

SOLAGRO (2007). Energie dans les exploitations agricoles : état des lieux en Europe et éléments de réflexion pour la France. Etude ADEME/ MAP 0671C0036 (2007)**:** 34.

Stefanovic, N., J. Han and K. Koperski (2000). "Object-based selective materialization for efficient implementation of spatial data cubes." IEEE Transactions on Knowledge and Data Engineering **12**(6): 938-958.

Talend. (2012a). "Talend community Wiki- sdi:mainpage- Data Integration has never been easier." from http://www.talendforge.org/wiki/doku.php?id=sdi:MainPage.

Talend. (2012b). "Web site Talend." from www.talend.com.

Thornsbury, S., K. Davis and T. Minton (2003). "Adding value to agricultural data: a golden opportunity." Review of Agricultural Economics **25**(2): 550-568.

Torlone, R. (2003). Conceptual multidimensional models. Multidimensional databases, IGI Publishing**:** 69-90.

Trolle, D., D. P. Hamilton, C. A. Pilditch, I. C. Duggan and E. Jeppesen (2011). "Predicting the effects of climate change on trophic status of three morphologically varying lakes: Implications for lake restoration and management." Environmental Modelling & Software **26**(4): 354-370.

Tryfona, N., F. Busborg and J. G. B. Christiansen (1999). starER: a conceptual model for data warehouse design. Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP. Kansas City, Missouri, United States, ACM**:** 3-8.

Turki, I. Z., F. G. Jedidi and R. Bouaziz (2010). Multiversion data warehouse constraints. Proceedings of the ACM 13th international workshop on Data warehousing and OLAP. Toronto, ON, Canada, ACM**:** 11-18.

Ubeda, T. and M. J. Egenhofer (1997). Topological Error Correcting in GIS. Proceedings of the 5th International Symposium on Advances in Spatial Databases, Springer-Verlag**:** 283-297.

Vassiliadis, P. and T. Sellis (1999). "A survey of logical models for OLAP databases." SIGMOD Rec. **28**(4): 64-69.

Vilain, L. (2008). La méthode IDEA: indicateurs de durabilité des exploitations agricoles, Educagri éditions.

Villarroel, R., E. Fernández-Medina, M. Piattini and J. Trujillo (2006). "A UML 2.0/OCL Extension for Designing Secure Data Warehouses." Journal of Research and Practice in Information Technology **38**(1).

Viswanathan, G. and M. Schneider (2010). BigCube: A Metamodel for Managing Multidimensional Data. SEDE**:** 237-242.

Viswanathan, G. and M. Schneider (2011). On the Requirements for User-Centric Spatial Data Warehousing and SOLAP. Database Systems for Adanced Applications. J. Xu, G. Yu, S. Zhou and R. Unland, Springer Berlin / Heidelberg. **6637:** 144-155.

WHO. (2009). "The World Health Organization Global InfoBase (data warehouse)." from http://apps.who.int/infobase.

Xi-Qian, C., C. Zhong-Xian and C. Xiu-Kun (2004). Applying DP to ETL of spatial data warehouse. Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on.