Chapitre 7

Implantation d'un modèle conceptuel avec multi-représentation

7.1. Introduction

Le chapitre précédent a proposé plusieurs modèles de données pour décrire des données spatiales ou spatio-temporelles sous plusieurs représentations. Notamment le modèle MADS [VAN 02] a été défini pour la conception des bases de données spatio-temporelles avec multi-représentation. Il offre une approche générale qui permet au concepteur d'associer à chaque entité, lien ou propriété du monde réel plusieurs descriptions selon le point de vue, la résolution sémantique ou spatiale, ou selon toute autre caractéristique générant des descriptions différentes. Cependant, une fois défini le schéma de la base de données selon ce modèle, la conception de la base de données n'est pas terminée. Il reste alors à implanter ce schéma, qui est de niveau conceptuel, sur un SIG ou SGBD existant. Les modèles de données de ces systèmes sont de niveau logique. Les concepts qu'ils offrent sont moins génériques que ceux d'un modèle conceptuel en général et de MADS en particulier. L'éventail de concepts est plus restreint.

Par exemple, le plus souvent pour la dimension temporelle seul le type de données temporel Date est offert, alors que MADS en offre un jeu complet allant des types simples Instant et Intervalle, au type générique Temps qui permet de décrire toute valeur temporelle qu'elle soit un instant, un intervalle ou un ensemble d'instants et/ou d'intervalles. De même, les possibilités de multi-représentation offertes par les systèmes actuels se résument aux possibilités de multi-instanciation des systèmes de type orienté-objets, qui permettent de représenter une même entité du monde réel par des instances de plusieurs classes et sous-classes. Mais aucun système n'offre un mécanisme générique de multi-représentation comme celui de MADS. Enfin, les modèles de données des SIG pour la dimension spatiale sont souvent fortement contraints par les techniques d'implantation interne propres au système. De plus, les SIG n'offrent qu'un jeu limité de concepts spatiaux : soit une vue discrète, soit une vue continue, et si les deux vues sont offertes ce ne sera pas

---

Chapitre rédigé par Christine PARENT, Esteban ZIMANYI, Mohammed MINOUT, ET Ali AISSAOUI

dans le cadre du même modèle. La dimension spatiale n'est pas orthogonale à la dimension structurelle : seuls certaines structures (soit le type d'objet, soit l'attribut) peuvent avoir une caractéristique spatiale...

Toutes ces limites des modèles de données des systèmes existants font qu'il faut transformer le schéma conceptuel – qu'il soit exprimé en MADS ou en tout autre modèle conceptuel – pour pouvoir l'implanter sur l'un d'eux. Cette transformation d'un schéma MADS consiste à remplacer chaque occurrence d'un concept *C* de MADS qui n'a pas d'équivalent dans le modèle du système visé (que nous appellerons dorénavant modèle cible) par une construction (un morceau de schéma), toujours en MADS, mais qui n'emploie plus le concept *C*. Par exemple, un attribut MADS de domaine Intervalle sera remplacé par deux attributs de domaine Instant (qui est équivalent au domaine Date du modèle cible) décrivant respectivement le début et la fin de l'intervalle de temps.

Une fois le schéma conceptuel de départ complètement transformé on obtient un nouveau schéma, toujours exprimé en MADS, mais qui n'utilise que des concepts de MADS qui ont un équivalent dans le modèle cible. Ce nouveau schéma est appelé schéma MADS-, parce que pratiquement c'est un schéma correct, mais "faible" au sens où il n'emploie pas efficacement le modèle ; ce qui en l'occurrence est voulu. Il ne reste plus qu'à réécrire ce schéma MADS- en utilisant la syntaxe du modèle cible.

L'intérêt de passer par un schéma MADS- est d'obtenir un outil évolutif. Les modèles cibles se regroupent en familles qui partagent des concepts communs, par exemple les SIG basés sur le relationnel ou ceux basés sur le relationnel-objet. Les règles de transformation seront ainsi communes à plusieurs modèles cibles, selon le principe suivant : chaque transformation qui remplace un concept *C* par une structure employant les concepts C_1, C_2, ... et C_n pourra être utilisée par (au moins) tous les modèles cibles qui n'offrent pas le concept *C*, mais qui offrent les concepts C_1, C_2, ... et C_n. Rajouter un nouveau modèle cible revient ainsi le plus souvent à réaliser le module de réécriture spécifique de la cible, sans avoir à ajouter de nouvelle règle de transformation.

Les règles de transformation structurelle d'un schéma conceptuel vers un schéma logique sont bien connues (voir par exemple [HAI 91]), et couramment employées dans des outils d'aide à la conception tels que Merise [TAR 00] et UML [BOO 98]. Quelques outils proposent une extension pour la dimension spatiale ou temporelle [PER 02]. Dans ce chapitre nous définissons une approche globale et cohérente de la transformation d'un schéma conceptuel spatio-temporel avec multi-représentation en un schéma logique. L'approche est globale car elle prend en compte les quatre dimensions, structurelle, spatiale, temporelle et de multi-représentation. Elle est cohérente, car les quatre dimensions sont traitées de façon similaire, la même règle pouvant s'appliquer au cas du temps, de l'espace ou de la multi-représentation. Ce

sont les principes d'orthogonalité et de similitude du modèle MADS qui ont permis de définir une approche cohérente et par là simple. En effet, les quatre dimensions sont orthogonales, ce qui signifie que tout concept structurel (objet, association, attribut ou méthode) peut ou non être spatial, temporel et/ou multi-représenté, le choix d'une caractéristique sur une dimension étant indépendant des choix sur les autres. Cela simplifie le modèle et en conséquence les transformations des schémas. Les dimensions spatiale, temporelle et de multi-représentation sont traitées de façon semblable, avec le même type de concepts, ce qui là encore simplifie l'apprentissage du modèle et minimise le nombre des règles de transformation nécessaires.

Dans le cadre des projets européens MurMur [MUR 02] et Cobalt [COB 02], un jeu d'outils d'aide à la conception de schémas MADS avec multi-représentation a été développé. Parmi ces outils il y a un éditeur visuel de schémas MADS et un traducteur dont le rôle est d'implanter les schémas MADS sur divers systèmes existants tels qu'Oracle 9i, ArcView ou MapInfo.

Dans ce chapitre, le paragraphe 7.2 présente l'architecture du module de traduction. Les règles de transformation sont ensuite exposées dans le paragraphe 7.3 : règles structurelles, de multi-représentation, spatiales et temporelles. Le paragraphe 7.4 présente les transcripteurs, modules qui réécrivent les schémas MADS- dans la syntaxe des systèmes cibles. Le paragraphe 7.5 présente un exemple de traduction d'un schéma de gestion d'un réseau fluvial. Finalement la conclusion rappelle les résultats et ouvre des perspectives.

7.2. Architecture du module de traduction

Le module de traduction (appelé aussi le *traducteur*) a pour objectif de traduire les schémas conceptuels MADS en schémas logiques des systèmes existants. Il est composé de deux niveaux de modules :

- le module de *transformation* qui transforme les schémas MADS en schémas MADS- adaptés à un modèle cible X (par exemple, Oracle 9i);
- un jeu de modules spécialisés, un par modèle cible, les *transcripteurs*. Le transcripteur correspondant au système X traduit les schémas MADS- adaptés au modèle X en schémas exprimés avec la syntaxe de la plate-forme X (par exemple, en scripts SQL pour Oracle 9i). Cette traduction est du type réécriture.

Le module de transformation est lui-même composé de plusieurs modules :

- le module des *règles* qui constitue le noyau du traducteur. Il contient l'ensemble des règles de transformation structurelle, spatiale, temporelle et de multi-représentation ;

- un module *sélecteur* par modèle cible, dont le rôle est de choisir en fonction du modèle cible les règles de transformation à appliquer et de définir dans quel ordre. En effet, comme le montrent notamment les figures 2 à 6, le même élément peut être successivement transformé par plusieurs règles. L'ordre global est le suivant : en premier les règles de multi-représentation, ensuite celles relatives aux dimensions temporelle et spatiale, et pour finir les règles structurelles.

7.3. Le module des règles de transformation

Comme les quatre dimensions, structurelle, spatiale, temporelle et de multi-représentation ont été définies dans le modèle MADS de façon orthogonale, les transformations d'un élément du schéma (par exemple un type d'objet spatio-temporel avec multi-représentation) selon chacune des dimensions sont indépendantes des caractéristiques de l'élément sur les autres dimensions. Les règles ont donc pu être définies pour chaque dimension, indépendamment des autres.

Les paragraphes suivants présentent quelques règles de traduction pour chacune des dimensions. Dans les figures décrivant des schémas MADS les caractéristiques spatiales, temporelles et de multi-représentation y sont visuellement représentées par les symboles décrits dans la table ci-dessous. Nous renvoyons à [VAN 02] pour la liste complète des types spatiaux et temporels disponibles dans MADS.

Icône	Signification
	Type abstrait spatial générique : Géo
•	Type abstrait spatial : Point
	Type abstrait spatial: Ligne orientée
	Type abstrait spatial: Surface complexe
	Type d'association spatial : Adjacence
	Type abstrait temporel: Instant
	Type abstrait temporel: Intervalle
e1, e2, …	Estampilles qui définissent une représentation

Table 1. *Icônes représentant les caractéristiques spatiales, temporelles et de multi-représentation.*

7.3.1. *Règles de transformations structurelles*

Le modèle structurel de MADS offre les concepts habituels des modèles entité-association étendus : type d'objet, type d'association, attribut, méthode et lien de généralisation/spécialisation (ou IsA). Les attributs peuvent être simples ou complexes (c'est-à-dire composés d'attributs) à n'importe quelle profondeur. Ils peuvent être facultatifs ou obligatoires, monovalués ou multivalués (à valeurs multiples). Un attribut peut être dérivé par une expression à partir d'autres attributs. Les associations peuvent être n-aires et cycliques. Les liens IsA supportent le raffinement, la redéfinition et la surcharge des attributs et des méthodes. MADS offre également des types d'association spécifiques comme l'agrégation.

Etant donné que les règles de traduction pour la dimension structurelle sont bien connues, nous présentons un seul exemple de règle de transformation de ce type.

7.3.1.1. *Transformation d'un attribut multivalué*

Dans de nombreux modèles cibles, dont ceux basés sur le relationnel, les attributs ne peuvent prendre qu'une seule valeur (ils sont monovalués). Les attributs multivalués (qui peuvent prendre plusieurs valeurs) n'existent pas. Une règle de transformation structurelle consiste donc à remplacer les attributs multivalués par une structure n'employant que des attributs monovalués.

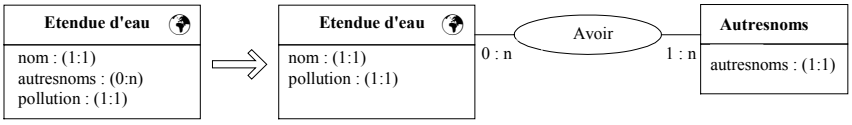

Figure 1a. *Transformation de l'attribut autresnoms en type d'objet.*

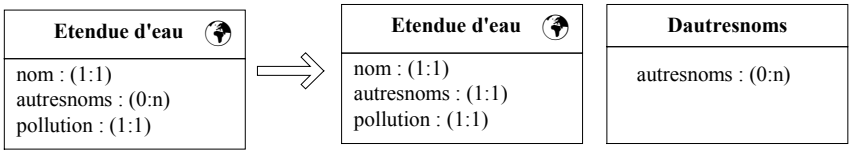

Figure 1b. *Transformation de l'attribut autresnoms en domaine.*

Selon la cible, relationnelle ou relationnel-objet, du modèle logique, un attribut multivalué se transforme de plusieurs façons. Dans le cas d'un modèle relationnel (Figure 1a), l'attribut *autresnoms*, qui est multivalué (de cardinalités 0:n), est transformé en un type d'objet *Autresnoms*. Cette règle ajoute également une association nommée *Avoir* qui relie le *Etendue d'eau* au type d'objet *Autresnoms.*

Par contre, dans un modèle relationnel-objet (tel qu'Oracle 9i), pour l'attribut *autresnoms* on crée un domaine multivalué *Dautresnoms* (Figure 1b) et l'attribut *autresnoms* devient monovalué et est défini sur ce nouveau domaine.

7.3.2. *Règles de transformation de la multi-représentation*

7.3.2.1. *Introduction*

Dans une base de données avec multi-représentation, chaque élément du schéma (type d'objet, type d'association, attribut et méthode) a une estampille définissant sa visibilité. Cette estampille est définie soit explicitement par le concepteur de la base de données, soit implicitement comme étant la même que celui de l'élément dont il fait partie (un type d'objet fait partie de son schéma, un attribut composant fait partie de son attribut complexe…).

7.3.2.2. *Génération explicite des estampilles d'un type d'objet ou d'association*

Considérons un schéma multi-représenté avec un type d'objet ou d'association qui ne possède pas d'estampilles explicites. Le but de cette règle est de rendre explicite les estampilles (c'est-à-dire la visibilité) du type. La figure 2 illustre l'application de cette règle au type d'objet *Rivière*, en supposant que le schéma auquel ce type appartient est défini pour les deux estampilles e1 et e3. Dans le type d'objet initial (à gauche dans la figure), les estampilles de *Rivière* ne sont pas définies explicitement : ce sont par défaut celles du schéma. Dans le type d'objet transformé (à droite dans la figure), la ligne sous le nom du type contient la liste de ses estampilles, maintenant définies explicitement.

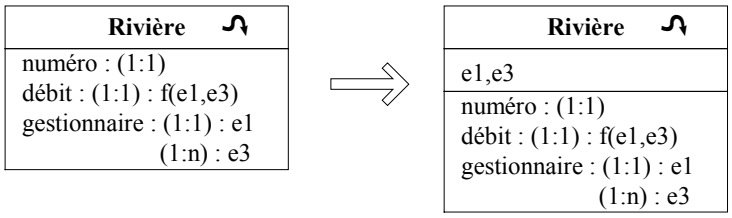

Figure 2. *Génération explicite des estampilles du type d'objet Rivière.*

7.3.2.3. *Ajout d'un attribut estampilles aux types d'objet et d'association multi-représentés*

Les instances d'un type d'objet ou d'association sont aussi estampillées. Cela veut dire qu'une instance peut être visible dans une représentation et pas dans une autre. Des lors, cette règle de transformation consiste à ajouter un attribut multivalué

de type ensemble, nommé *estampilles,* qui garde trace des estampilles de chaque instance. Dans la figure 3, *Rivière, adjacence* et *Ville* ont subit cette transformation.

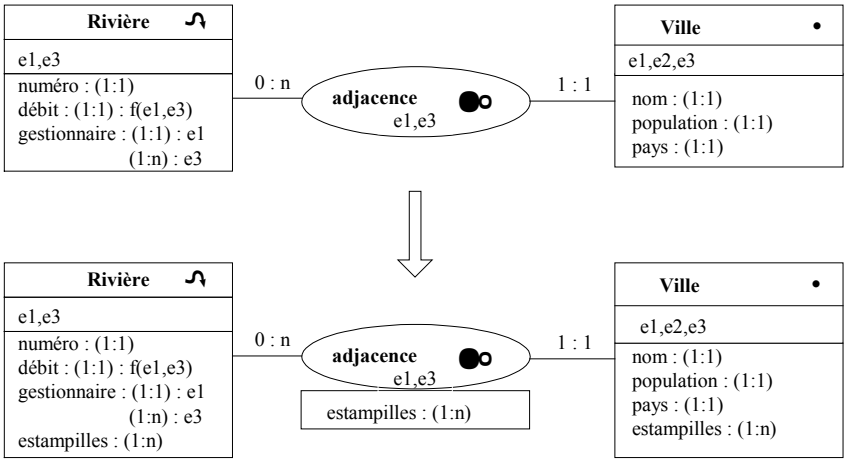

Figure 3. *Ajout de l'attribut estampilles.*

Cette règle génère également une contrainte d'intégrité qui spécifie que la valeur prise par l'attribut *estampilles* doit être un sous-ensemble de (ou être égale à) l'ensemble d'estampilles associées au type d'objet ou d'association.

7.3.2.4. *Génération explicite des estampilles des attributs et méthodes*

D'une manière analogue aux types d'objet et d'association (paragraphe 7.3.2.2), on applique cette règle pour rendre explicite les estampilles des attributs et des méthodes. La figure 4 montre l'application de cette transformation sur l'attribut *numéro* (dans les diagrammes les estampilles de visibilité d'un attribut sont indiquées à la fin de la ligne qui le décrit).

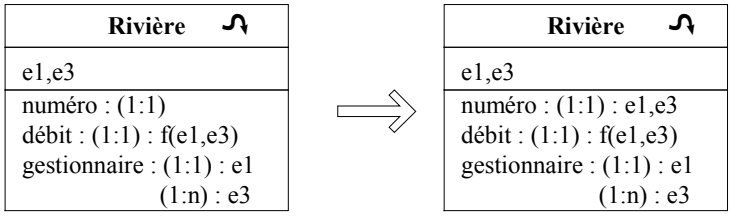

Figure 4. *Génération explicite des estampilles de l'attribut numéro.*

7.3.2.5. *Transformation d'un attribut (ou méthode) qui possède des définitions différentes selon les estampilles*

L'objet de cette règle est de transformer un attribut ou méthode à plusieurs définitions (chacune est associée à une estampille) en plusieurs attributs (un attribut par définition). En effet, les systèmes cibles ne permettent pas qu'un type ait deux attributs ou méthodes du même nom, ni qu'un attribut ou qu'une méthode ait deux définitions différentes. La figure 5 montre l'application de cette transformation sur l'attribut *gestionnaire* qui a deux définitions dans le schéma initial : monovalué pour l'estampille e1, multivalué pour l'estampille e3.

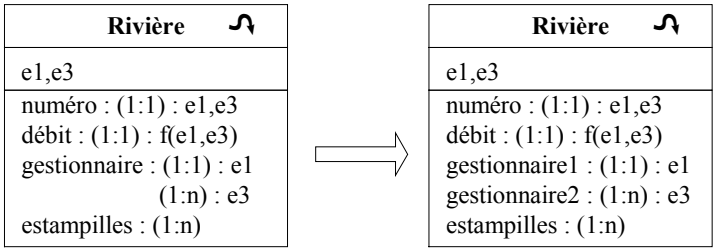

Figure 5. *Transformation de l'attribut gestionnaire.*

7.3.2.6. *Transformation d'un attribut qui a des valeurs différentes selon les estampilles*

Un attribut à valeurs estampillées est un attribut dont la valeur change selon l'estampille (point de vue et/ou résolution) utilisée pour y accéder. Cette règle consiste à transformer un tel attribut dont les valeurs sont estampillées en un attribut multivalué de cardinalité comprise entre 0 et le nombre d'estampilles de l'attribut. Cet attribut est complexe, composé de couples (*estampille, valeur*) où *estampille* a comme domaine l'ensemble d'estampilles de l'attribut. Cette transformation est montrée pour l'attribut *débit* de l'objet *Rivière* dans la figure 6, où la notation f(e1,e3) signale que la valeur de l'attribut change selon l'estampille, e1 ou e3.

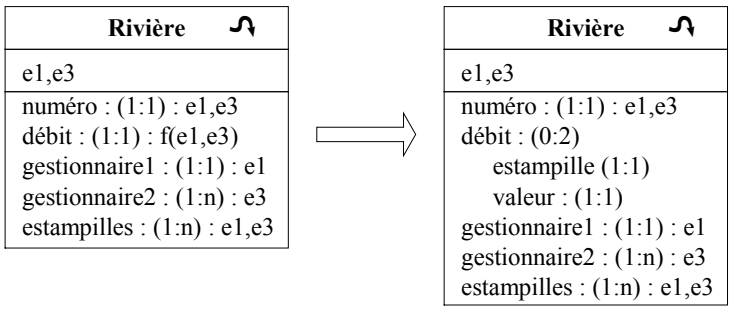

Figure 6. *Transformation de l'attribut à valeurs estampillées débit.*

Cette règle génère également une contrainte d'intégrité exprimant que la valeur de l'attribut composant *débit.estampille* doit être un élément de l'ensemble d'estampilles de l'attribut en question.

7.3.2.6. *Transformation des estampilles des attributs, des méthodes et des types*

Etant donné que les systèmes cibles ne connaissent pas les estampilles de multi-représentation, cette règle consiste à supprimer du schéma les estampilles qui définissent la validité d'un attribut, d'une méthode, d'un type d'objet ou d'un type d'association. Comme l'information véhiculée par ces estampilles est constante pour toutes les valeurs de l'attribut (ou pour toutes les occurrences du type), elle est de niveau meta; ce pourquoi elle est transformée en un attribut du meta-schéma. Le meta-schéma décrit les types d'objet et d'association ainsi que leurs propriétés, attributs et méthodes.

Par exemple, l'application de cette règle de transformation aux estampilles du type d'objet *Rivière* supprime les estampilles e1, e3 du type d'objet et complète l'instance *Rivière* du meta-type d'objet T-Objet de la façon suivante:

```
UPDATE T-Objet
SET estampilles={e1,e3}
WHERE nom="Rivière"
```

7.3.3. *Règles de transformations spatiales*

MADS fournit un ensemble de types abstraits spatiaux [SCO 96], organisés en une hiérarchie de généralisation [PAR 97]. A chaque type spatial est associé un ensemble de méthodes permettant de définir et de manipuler les instances de ce type. Le type géo, le plus générique, signifie uniquement "ce type est spatial" sans

préciser aucunement son emprise. La définition du type précis de chaque occurrence se fera lors de sa création.

7.3.3.1. *Transformation d'un objet spatial*

MADS permet au concepteur d'attacher une spatialité aux types d'objets comme aux attributs. Un type d'objet sera caractérisé par une spatialité lorsqu'il représente une classe d'entités dont la référence spatiale est pertinente pour l'application.

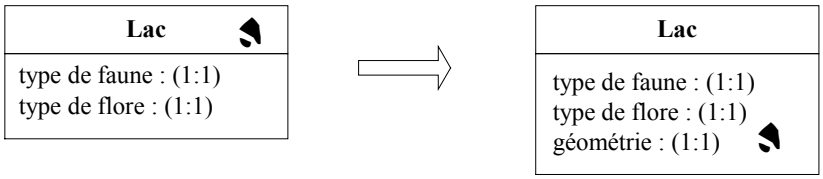

Figure 7. *Transformation d'un objet spatial en objet non spatial.*

Pour les systèmes cibles qui n'ont pas le concept d'objet spatial, mais qui ont celui d'attribut spatial, comme par exemple Oracle 9i, un type d'objet spatial est transformé en créant un attribut spatial monovalué, appelé *géométrie*, qui a le même domaine spatial que le type d'objet initial (voir figure 7) [1].

7.3.3.2. *Transformation d'un type de données spatial spécialisé en type générique*

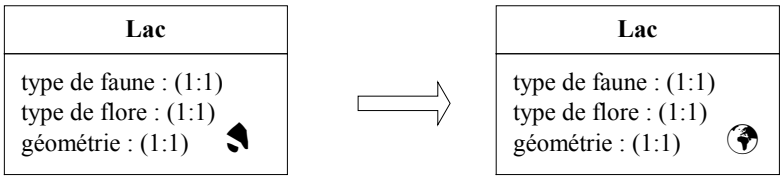

Figure 8. *Transformation d'un type de données spatial spécialisé en type générique géo.*

Cette règle permet de transformer un type de données spatial spécialisé (par exemple surface complexe pour l'attribut *géométrie* dans la figure 8) en le type générique géo. Cette transformation est nécessaire pour tous les modèles, tels qu'Oracle 9i, qui n'ont qu'un seul type spatial, qui est générique; la définition du type spécifique (point, ligne, surface…) se faisant lors de la création des instances [2].

---

[1] Pour les systèmes qui à l'inverse offrent les types d'objets spatiaux mais pas les attributs spatiaux, la règle inverse existe aussi dans le transformateur.

[2] Pour les systèmes qui à l'inverse offrent les types spatiaux spécialisés, comme point, ligne et surface, mais pas de type spatial générique, la règle inverse existe aussi dans le transformateur.

Par conséquent, cette règle génère aussi une contrainte d'intégrité exprimant que la valeur spatiale doit appartenir au type spécifié initialement dans le schéma MADS.

7.3.3.3. *Transformation d'un attribut variable dans l'espace*

En MADS, un attribut est dit variable dans l'espace si sa valeur dépend de l'emplacement considéré, comme c'est le cas par exemple pour l'altitude ou la couverture du sol. Un attribut variable est une fonction définie sur une étendue spatiale qui peut être tout l'espace couvert par la base de données ou au contraire une zone, voire une ligne donnée, et qui est discrétisée si nécessaire, puis partitionnée en un ensemble d'éléments spatiaux du même type, c'est-à-dire en un ensemble de points, de surfaces élémentaires ou de lignes élémentaires. La fonction (l'attribut variable) associe à chacun de ces éléments spatiaux une valeur du domaine de l'attribut. Cette dernière peut être, selon le type de l'attribut, une valeur simple (cas d'un attribut simple) ou une valeur complexe (cas d'un attribut complexe), une valeur unique (cas d'un attribut monovalué) ou un ensemble de valeurs (cas d'un attribut multivalué).

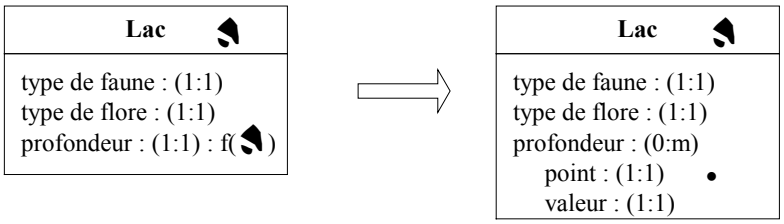

Figure 9. *Transformation de l'attribut variable profondeur.*

Pour les systèmes qui ne possèdent pas la notion d'attribut variable dans l'espace, la transformation consiste à remplacer l'attribut variable par un attribut complexe multivalué regroupant un élément spatial et une valeur. La figure 9 présente cette transformation pour l'attribut variable *profondeur* [3]. Le premier attribut composant est l'attribut *point* de type spatial point. Le deuxième est la *valeur* de l'attribut en ce point.

[3] Dans les diagrammes la notation f() désigne un attribut variable.

7.3.4. *Règles de transformation temporelles*

MADS permet d'associer la temporalité à tout concept du modèle (objet, association, et attribut quelque soit son niveau). La temporalité exprime le cycle de vie des objets ou des associations temporels et la validité des valeurs des attributs temporels.

7.3.4.1. *Transformation du cycle de vie d'un objet ou d'une association*

En MADS, un type d'objet (ou d'association) temporel garde trace du cycle de vie de ses instances, défini par les événements de création, de suspension, de réactivation et de destruction. Pour chaque instance, cette information reste disponible après sa destruction ainsi que sa valeur (valeur courante pour les attributs non temporels, historique des valeurs pour les attributs temporels), de manière à pouvoir associer l'information sur le cycle de vie à l'objet correspondant du monde réel.

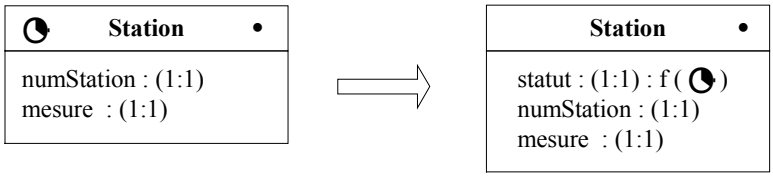

Figure 10. *Transformation du cycle de vie du type d'objet Station.*

De façon similaire aux transformations pour la spatialité, le cycle de vie d'un type d'objet temporel est transformé en créant un attribut temporel monovalué appelé *statut*, qui peut prendre comme valeur un des quatre états : prévu, actif, suspendu ou détruit (voir figure 10).

Cette règle génère également un ensemble de contraintes sur la temporalité associée aux états. Par exemple, une instance ne peut pas être active et suspendue en même temps. Une autre contrainte d'intégrité précise quel type temporel associer à la période de validité de la valeur active : un instant pour un objet temporel décrivant un évènement instantané (et identifié comme tel par l'icône d'instant pour son cycle de vie), un intervalle pour un objet temporel ayant un cycle de vie simple sans suspension, un ensemble d'intervalles pour un objet temporel qui peut être suspendu…

7.3.4.2. *Transformation d'un attribut temporel*

Un attribut temporel est un attribut dont l'historique, voire le futur, des valeurs est conservé. C'est, à l'instar des attributs variables dans l'espace, un attribut variable dans le temps. Il est décrit par une fonction définie sur un intervalle de temps, qui peut couvrir la durée de validité globale de la base de données. Cette fonction associe à chaque élément temporel (instant ou intervalle) de l'intervalle de temps, la valeur de l'attribut à ce moment là. Comme les systèmes existants n'ont pas de concept d'attribut temporel variable dans le temps, l'objet de cette règle est de transformer un attribut temporel, comme par exemple *statut* dans la figure 11, en un attribut multivalué et complexe portant le même nom, dont les composants sont un élément temporel (instant ou intervalle) et une valeur. Cette transformation permet d'exprimer la sémantique des attributs temporels en utilisant une structure de données existante dans le système cible.

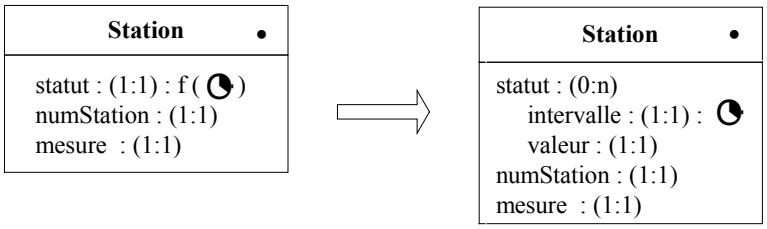

Figure 11. *Transformation de l'attribut temporel statut.*

Cette règle génère une contrainte d'intégrité exprimant que les intervalles doivent être disjoints et que les valeurs doivent appartenir au domaine du *statut*, à savoir un des états : prévu, actif, suspendu ou détruit.

7.3.4.3. *Transformation d'un attribut de type intervalle*

La plupart des systèmes existants ont un domaine de type instant (par exemple le type Date), mais pas de domaine de type intervalle de temps. Cette règle transforme donc un attribut de type intervalle en un attribut complexe portant le même nom, de même cardinalité, et dont les attributs composants sont, comme le montre la figure 12, de type instant. Cette règle génère une contrainte d'intégrité exprimant que pour chaque valeur de l'attribut *intervalle*, *intervalle.début* doit être inférieur ou égal à *intervalle.fin*.

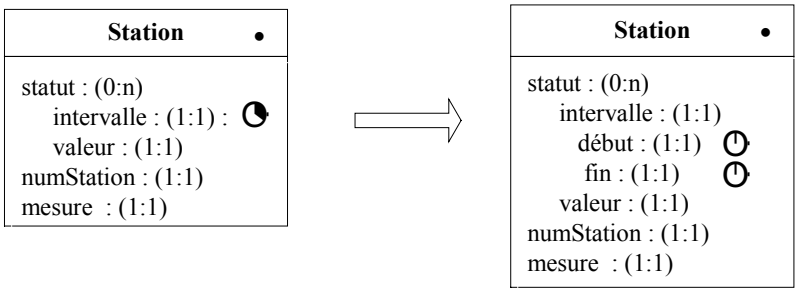

Figure 12. *Transformation de l'attribut intervalle.*

7.4. Modules spécifiques des systèmes cibles

Pour chaque SIG et SGBD cible, il existe un transcripteur qui re-écrit le schéma MADS- fourni par le transformateur, et qui génère un schéma exprimé dans le langage du système cible.

Par exemple, la création des scripts SQL pour Oracle 9i (relationnel-objet) se fait par la re-écriture du schéma MADS-. Lors de la re-écriture pour Oracle 9i, les types d'objet MADS- génèrent des tables d'objets et les domaines multivalués MADS- génèrent soit des tables imbriquées (nested tables), soit des types VARRAY d'Oracle 9i.

Pour que le schéma MADS initial et celui implanté sur le système cible soient sémantiquement équivalents, il reste encore à compléter ce dernier par des clauses d'intégrité (par exemple des triggers, des checks ou des procédures) qui implémentent les contraintes d'intégrité générées lors des transformations.

7.5. Exemple

Nous présentons, figure 13, un schéma MADS décrivant une partie d'un réseau fluvial. Ce schéma a été traduit par le traducteur afin d'obtenir un schéma MADS- pour la cible Oracle 9i, en utilisant le modèle objet- relationnel. Le résultat est donné dans la figure 14.

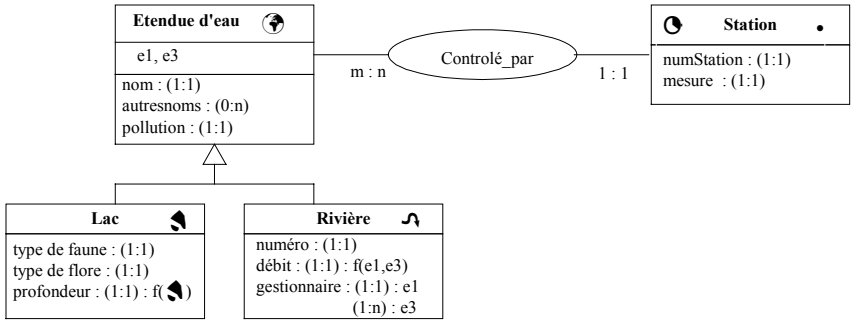

Figure 13. *Schéma MADS d'application de gestion d'un réseau fluvial.*

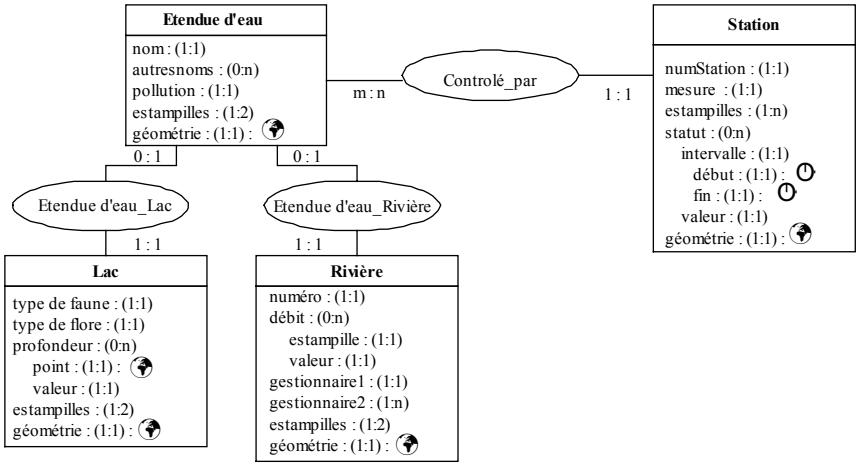

Figure 14. *Traduction de l'exemple en MADS- pour Oracle 9i objet-relationnel.*

Finalement, le transcripteur traduira ce schéma MADS- en un ensemble de commandes SQL. Nous montrons dans la suite uniquement une partie de ces commandes :

- CREATE OR REPLACE TYPE DStationRef AS TABLE OF REF DStation ;
- CREATE OR REPLACE TYPE DEtendue_d_eau AS OBJECT (id Did, geometrie MDSYS.SDO_GEOMETRY, nom VARCHAR2(255), autresnoms DEtendue_d_eau_autresnoms, pollution INTEGER, estampilles DEtendue_d_eau_estampilles, StationRef DStationRef, LacRef REF DLac, RiviereRef REF DRiviere) ;
- CREATE OR REPLACE TYPE DStation AS OBJECT (id Did, geometrie MDSYS.SDO_GEOMETRY, statut DStation_statut, numStation

VARCHAR2(255), mesure INTEGER, estampilles DStation_estampilles, Etendue_d_eauRef REF DEtendue_d_eau) ;

- CREATE OR REPLACE TYPE DLac AS OBJECT (id Did, geometrie MDSYS.SDO_GEOMETRY, type_de_faune VARCHAR2(255), type_de_flore VARCHAR2(255), estampilles DLac_estampilles, profondeur DLac_profondeur, Etendue_d_eauRef REF DEtendue_d_eau) ;
- CREATE OR REPLACE TYPE DRiviere AS OBJECT (id Did, geometrie MDSYS.SDO_GEOMETRY, numero INTEGER, debit DRiviere_debit, estampilles DRiviere_estampilles, Etendue_d_eauRef REF DEtendue_d_eau)
- CREATE TABLE Etendue_d_eau OF DEtendue_d_eau NESTED TABLE StationRef STORE AS Etendue_d_eau_NT;
- CREATE TABLE Station OF DStation NESTED TABLE statut STORE AS Station_NT;
- CREATE TABLE Lac OF DLac NESTED TABLE profondeur STORE AS Lac_NT;
- CREATE TABLE Riviere OF DRiviere NESTED TABLE debit STORE AS Riviere_NT;

Comme on peut le voir, le transcripteur doit prendre en compte les restrictions propres à la cible Oracle 9i. Notamment les caractères accentués, les apostrophes et les espaces ne sont pas acceptés dans les noms. Dans les tables ci-dessus *geometrie* décrit la spatialité des types *Etendue d'eau* et de *Station*. De même, *statut* décrit la temporalité de *Station*. L'attribut *id* décrit l'identificateur de chaque type d'objet. Tout attribut qui commence par un nom de type d'objet et qui termine par *Ref* exprime une référence à un objet type (par exemple, *LacRef* est une référence au *Lac*).

Le traducteur génère également un ensemble de domaines. Dans le cas de notre exemple, l'attribut variable dans l'espace *profondeur* se transforme en attribut simple ayant un domaine *DLac_profondeur* qui est complexe et multivalué (en l'occurrence une table imbriquée). L'attribut *débit* qui varie en fonction de la représentation se transforme de la même façon en attribut simple ayant un domaine *DRiviere_debit.*

7.5. Conclusions et perspectives

Ce chapitre décrit un outil qui permet la traduction d'un schéma conceptuel spatio-temporel avec multi-représentation en un schéma logique, puis en un schéma physique implanté sur une plate-forme cible (SIG ou SGBD). L'outil est composé de deux modules. Un traducteur générique utilise un ensemble de règles de transformation, chacune permettant d'exprimer un concept sémantiquement riche du

modèle MADS avec les mécanismes disponibles dans un système cible. Ensuite, un transcripteur exprime ce schéma transformé dans la syntaxe propre de la cible. Il y a autant de transcripteurs que de plate-formes cibles.

L'outil décrit dans ce chapitre fait partie d'un ensemble d'outils, développés notamment dans le cadre des projets de recherche européens MurMur et Cobalt, qui permettent la définition et la manipulation de bases de données spatio-temporelles en utilisant la modélisation conceptuelle. L'ensemble de ces modules a été réalisé en Java.

Un éditeur de schéma MADS permet à l'utilisateur de concevoir le schéma conceptuel de la base de données de manière graphique et conviviale. Lorsque l'utilisateur choisit la cible souhaitée, le traducteur prend comme entrée le schéma conceptuel MADS (exprimé en XML) et génère le schéma associé MADS- (également exprimé en XML) qui est ensuite envoyé au transcripteur pour générer les scripts propres à la cible choisie. Parmi les cibles qui ont déjà été réalisées figurent Oracle 9i, ArcView et MapInfo, d'autres cibles sont en cours de développement.

D'autres modules permettent la définition visuelle des requêtes exprimées sur le schéma conceptuel et un autre traducteur génère la requête correspondante qui est adressée à la cible (par exemple, requête en SQL pour Oracle 9i).

Concernant les travaux futurs du traducteur, en vue de l'optimisation du schéma physique il serait intéressant de rajouter une couche de transformations qui, à l'aide d'un dialogue avec l'administrateur de la base de données à propos des caractéristiques des applications (types de requêtes, fréquence…), préparerait les choix internes, tels que le choix des index ou du type d'implémentation (couches topologiques, spaghetti…).

Bibliographie

[BOO 98] BOOCH G., JACOBSON I., RUMBAUGH J., *The Unified Modeling Language: User Guide*, Addison-Wesley, 1998

[COB 02] Cobalt : Conception de Bases de Données Localisées Temporelles, Rapport final INTEREG II, Laboratoire THEMA, Université de Franche-Comté, Besançon et EPFL-LBD, Lausanne, mars 2002.

[HAI 91] HAINAUT J.L., « Entity-Generating Schema Transformation for Entity Relationship Models », *Proceedings of the 10th Int. Conference on the Entity Relationship Approach*, San Mateo, CA, USA, octobre 1991.

[MUR 02] http://lbd.epfl.ch/e/MurMur/

[PAR 97] PARENT C., SPACCAPIETRA S., ZIMANYI E., DONINI P., PLAZANET C., VANGENOT C., ROGNON N., CRAUSAZ P.A., « MADS : Un modèle conceptuel pour des applications spatio-temporelles», *Revue internationale de Géomatique*, vol. 7, no 3-4, 1997.

[PER 02] http://sirs.scg.ulaval.ca/perceptory/

[SCO 96] SCOLLI M., VOISARD A., PELOUX J.P., RIGAUX P., *SGBD Géographiques – Spécificités*, International Thomson Publishing, 1996.

[TAR 00] TARDIEU H., ROCHFELD A., COLLETTI R., *La Méthode Merise : Principes et outils*, Ed. Organisation, 2000

[VAN 02] VANGENOT C., PARENT C., SPACCAPIETRA S., Modélisation et manipulation de données spatiales avec multi-représentation dans le modèle MADS, dans ce volume.

Remerciements

Les auteurs remercient vivement tous ceux qui ont travaillé à la conception et réalisation de cet outil, notamment Louis Jacomet, Olivier Samyn, Malek Ben Youssef et Souleymane Thiam. Nous remercions également la Commission Européenne qui a soutenu les projets MurMur ("Multiple Representations - Multiple Resolutions" - IST 10723) et COBALT (programme Interreg II).