Construction et manipulation de présentations spatio-temporelles multimédias à partir de serveurs d'objets répartis : applications aux données sur le Web

José-Luis Zechinelli-Martini

Université Joseph Fourier - Grenoble I

Thèse

pour obtenir le grade de

Docteur de l'Université Joseph Fourier

(arrêtés ministériels du 5 juillet 1984 et du 30 mars 1992)

Discipline : Informatique

présentée et soutenue publiquement par

José Luis Zechinelli Martini

Le 27 avril 2001

Construction et manipulation de présentations spatio-temporelles multimédias à partir de serveurs d'objets répartis : applications aux données sur le Web

Composition du jury

Président :	Jean-Pierre Peyrin
Rapporteurs :	Anne Doucet
	Jacques Kouloumdjian
Examinateur :	Christine Collet
	Andrzej Duda
	Cécile Roisin
Directeur de thèse :	Michel Adiba

Thèse préparée au sein du laboratoire Logiciels, Systèmes et Réseaux, IMAG

Je viens d'un petit village au centre du Mexique appelé Chipilo. Il a été fondé le 7 octobre 1882 par 40 familles italiennes provenant de la région du Veneto. À Chipilo, nous sommes des gens simples, sédentaires pour la plupart et je pense que personne n'a jamais dédié une thèse à son petit univers. Je tiens donc à dédier ce travail, mon effort et ma réussite à Chipilo, une patrie dans ma grande patrie qui est le Mexique.

Je tiens à remercier :

Jean Kouloumdjian, Professeur à l'Institut National de Sciences Appliquées de Lyon et Anne Doucet, Professeur à l'Université Pierre et Marie Curie, les rapporteurs de ma thèse, pour le temps qu'ils ont consacré à la lecture de ce document et pour leurs commentaires qui ont beaucoup servi à l'améliorer. Andrej Duda Professeur à l'Institut Polytechnique de Grenoble, Jean-Pierre Peyrin, Professeur à l'Universtie Joseph Fourier, Cécile Roisin, Maître de Conférences à l'Université Pierre Mendès France pour avoir accepté de faire partie du jury de cette thèse.

Christine Collet, Professeur à l'Institut Polytechnique de Grenoble et responsable de l'équipe STORM-Bases de données, pour m'avoir accueilli dans son équipe et d'avoir accepté de participer au jury de ma thèse. Je remercie surtout son soutien, ses conseils, ses commentaires et ses regards toujours parlants qui m'ont toujours encouragé à continuer avec ce difficile challenge qui est la réalisation d'une thèse.

Prof. Michel Adiba, mon directeur de thèse, pour son soutien et sa patience lors de ces années. Il est à l'origine de tout ce que j'ai appris pendant ma formation doctorale. Je le remercie en particulier de m'avoir guidé avec sagesse pendant ces quatre années.

Claudia Roncancio, à qui je dois en grande partie mon diplôme de DEA et Elizabeth Perez-Cortez pour les lectures qu'elle a fait de quelques chapitres de ma thèse. Merci aussi pour leurs conseils et les discussions qui ont contribué à la préparation de ma présentation de thèse.

Merci à Pierre Habraken pour le temps qu'il m'a consacré pour discuter avec moi sur d'OQLiST, ses commentaires pertinents, mesurés et clairs ont contribué grandement à la spécification de ce langage.

Merci aux membres et collaborateurs du projet NODS. À mes collègues de bureau Patricia Serrano-Alvarado et Tanguy Nedelec pour leurs conseils et leurs réponses à mes questions. À Khalid Belhajjam, qui sait toujours être disponible pour aider ses amis. À Thierry Coupaye, Stéphane Drapeau, Olivier Guyotot, Luciano Garcia-Banuelos, Gennaro Bruno, Tuyet-Trin Vu, Phuong-Quynh Duong pour le travail et l'amitié que nous partageons. Merci aussi pour leur aide précieuse lors de la préparation de ma soutenance.

Edgard Benitez-Guerrero et Carmen Mezura-Godoy pour avoir partagé de bons et de mauvais moments. Pour leur compagnie et encouragements, en gros pour être de très bons amis. Merci à Edgard, expert en entrepôt de données, pour le temps qu'il a consacré à de longues discussions sur mon travail entre autres.

Merci aux amis Marlon, Rafa, Olivier Lobry, Nicolas Henry, Helena Grazziotin et

Alexandre Ribeiro et Tony qui ont rendu mon séjour en France plus agréable. À Marlon et Rafa pour leur compagnie et leur amitié qui est au delà des frontières lointaines. À Nicolas Henry mon premier moniteur de ski.

Rachid Echahed, Professeur à l'Institut National Polytechnique de Grenoble pour être un professeur modèle. Merci de son aide et du temps qu'il a consacré à m'aider avec les durs concepts de la logique.

Paul Jacquet, Professeur à l'Institut National Polytechnique de Grenoble et directeur du Laboratoire Logiciels, Systèmes, Réseaux (LSR-IMAG).

Merci aux membres des équipes du LSR : STORM, SIGMA, DRAKKAR, SCOP, PFL, ADELE et PLIAGE. À Marie-Christine Fauvet et Pierre-Claude Scholl, membres de l'équipe STORM. Je remercie Dominique Decouchant, Jésus Favela, Hervé Martin pour leur aide dans la préparation de la présentation de ma soutenance. Merci à Dominique Rieu pour m'avoir guidé dans la spécification de mon premier schéma UML. Merci à Jacquie Stublier et en particulier à Jean-Marie Favre pour sa direction et son soutien lors de mon stage de DEA.

Merci à Liliane, Martine, Solange, Sonia pour leur aide dans les aspects administratifs, mais surtout pour leur gentillesse. François Challier pour son aide dans les aspects techniques qui constituent le support matériel des résultats de mon travail.

Je remercie les membres de l'équipe OPERA, INRIA Rhône Alpes, pour l'intérêt qu'ils ont porté à mon travail lors de mes visites chez eux. Leurs questions et remarques ont contribué grandement à la dernière mise au point de mon travail.

Consejo Nacional de Ciencia y Tecnologia (CONCACyT) du gouvernement mexicain pour son soutien économique. J'espère, en échange, pouvoir contribuer au développement de mon pays.

Merci à Ofelia Cervantes-Villagomez, Professeur à l'Universidad de las Américas et Warren Greiff, Information Technology Center qui ont contribué à ce que mes études en France soient possibles. À Cristina Loyo, directeur du LANIA pour l'intérêt qu'elle a porté sur mon travail.

Ma Mère qui s'est toujours battue pour que je sois quelqu'un. Je la remercie pour son amour tendre, son courage sans mesure et son exemple silencieux.

Mon Père qui a su transmettre sa joie de vivre, son optimisme et son amitié. Merci pour ton soutien inconditionnel, *popa*.

Je remercie mes sœurs et frères Gaby, Tere, Domingo et Pedro qui peuplent toujours les beaux souvenirs de mon enfance.

Ma sœur aînée, Gaby, à mon beau frère Omar et à mes neveux et nièces Maritsa, Omarcito, Karina et Iván.

Ma sœur Tere, à mon beau frère Vichis et mes neveux et nièces neveux Lorena, Alejandra et José Luis.

Domingo, mon frère que je porte dans mon cœur. Tous les mots que je sais en Français ne suffisent pas pour exprimer ce que je ressens pour lui.

Pedro, le plus jeune de ma famille, pour sa joie et sa bonne humeur contagieuse.

Mes grand-parents Dominga et Onofre, même s'ils sont des souvenirs lointains dans ma mémoire.

Mes grand parents Luis et Pierinna, *Nono*, je serai toujours un "macaco" qui garde les plus beaux souvenirs de ta bicyclette magique.

Comme toute famille italo-mexicaine (quelle combinaison !) nous sommes nombreux et très unis. Mes tantes et oncles : *barba* Miguel, *gegia* Nicoletta, *gegia* Tere, i me Santoli, *gegia* Cecilia et *barba* Pablo, *barba* José et *gegia* Angelina. Je remercie en particulier mon *barba* Miguel grâce à qui j'ai pu faire des études. Il a toujours été l'ange gardien de ma famille. Ma *gegia* Tere, pour s'être toujours inquiétée pour moi, pour ses lettres tendres et ses prières effectives. Je n'oublie pas ma *gegia* Dominga et Felipe. Un merci spécial à ma *gegia* Dominga qui a été l'esprit moteur qui m'a encouragé à choisir mon avenir. Ma *gegia* Nena et *barba* Carlos, *gegia* Tata, *gegia* Luisa, *gegia* Eugenia, *gegia* Maria et *barba* Vidal, *barba* Geres, *barba* Pedro, *barba* Francisco et *gegia* Carmen. Je remercie également tous mes cousins et cousines qui font toujours grandir la famille !

Merci à Alicia Gomez-Juarez d'être venue nous accompagner dans un des moments les plus significatifs de ma vie.

Maestra Guadalupe Solar-Quiroz y Maga, amigas consejeras y desde hace unos cinco años parte de mi familia (amies, conseillères et depuis quelques années membres de ma famille).

Genoveva, mon grand amour, qui est mon ange protecteur et toute ma vie.

Table des matières

1 Introduction **1**

1.1 Systèmes et environnements multimédias 1

1.2 Présentation multimédia et SGBD 2

1.3 Problématique et objectifs du travail 4

1.3.1 Modèle et langage 5

1.3.2 Vers une infrastructure d'intégration d'applications et de données multimédias 7

1.4 Validation expérimentale 8

1.5 Organisation du document 9

2 Données et systèmes multimédias **11**

2.1 Données multimédias 11

2.1.1 Texte 12

2.1.2 Image 15

2.1.3 Son 17

2.1.4 Vidéo 18

2.1.5 Discussion 20

2.2 Documents et présentations multimédias 21

2.2.1 Présentation de médias 21

2.2.2 Modèles de présentations temporelles 22

2.2.3 Modèles de présentations spatiales 22

2.2.4 Relations 23

2.2.5 Discussion 25

2.3 Environnements multimédias 27

2.3.1 Module de communication 27

2.3.2 Système d'exploitation multimédia 29

2.3.3 Boîte à outils multimédias 30
2.3.4 Discussion 31
2.4 SGBD multimédia 31
2.4.1 Fonctionnalités 32
2.4.2 Architecture d'un SGBDM 33
2.4.3 Principes de construction 34
2.4.4 Systèmes existants 35
2.4.5 Discussion 36
2.5 Les standards 37
2.5.1 Langages de programmation de présentations 37
2.5.2 Environnements standards 38
2.5.3 Discussion 39
2.6 Conclusion 39

3 Modèle spatio-temporel 41
3.1 Concepts de base 41
3.1.1 Espace de présentation 41
3.1.2 Ombre spatio-temporelle 42
3.1.3 Types d'objets 44
3.2 Présentations spatio-temporelles 45
3.2.1 Présentation atomique 46
3.2.2 Présentation composite 47
3.3 Relations spatio-temporelles 47
3.3.1 Relations d'intervalles 48
3.3.2 Relations topologiques et directionnelles 49
3.3.3 Positionnement des présentations 51
3.4 Composition de présentations 54
3.4.1 Description inter-média 54
3.4.2 Description du contenu d'objets 55
3.5 Comparaison avec d'autres modèles 57
3.5.1 Modèles spatiaux 57
3.5.2 Modèles temporels 58
3.5.3 Modèles spatio-temporels 59
3.6 Conclusion 60

4 Langage OQLiST **63**

4.1 Concepts de base 63
4.1.1 Présentation de OQL 63
4.1.2 Objets 65
4.1.3 Présentations 68
4.2 Constructeurs 72
4.2.1 Présentations atomiques et composites 72
4.2.2 Décor des présentations 72
4.2.3 Positionnement et taille des présentations 74
4.2.4 Présentations prédéfinies pour des collections d'objets 75
4.3 Ordonnancement spatial et/ou temporel 79
4.3.1 Opérateurs spatiaux 79
4.3.2 Opérateurs temporels 86
4.3.3 Ordre et profondeur des présentations 87
4.4 Interrogation et construction 88
4.4.1 Taxonomie des requêtes spatio-temporelles 88
4.4.2 Présentations comme résultats de requêtes 90
4.4.3 Description du contenu des objets 91
4.5 Conclusion 92

5 Infrastructure pour la spécification de gestionnaires de présentations **95**

5.1 Intégration d'objets dans des présentations 95
5.1.1 Gestionnaire de présentations 97
5.1.2 Fonctionnement 98
5.2 Données d'un gestionnaire 99
5.2.1 Modèle spatio-temporel et schéma de données d'une application 100
5.2.2 Contexte de présentation 101
5.2.3 Interaction avec les applications 102
5.2.4 Utilisation d'un gestionnaire 102
5.3 Transformation et génération de présentations 103
5.3.1 Transformation 103
5.3.2 Génération de présentations 104
5.4 Construction de présentations 108
5.4.1 Récupération de données 108

5.4.2 Intégration de présentations 109
5.5 Mise en place d'un gestionnaire 109
5.5.1 Modèle de JAGUAR 110
5.5.2 Spécification des contextes de présentation 112
5.5.3 Spécification d'un schéma de données 113
5.5.4 Génération 114
5.6 Conclusion 116

6 Implantation d'un gestionnaire de présentations 117
6.1 Architecture 118
6.1.1 Fonctions implantées 119
6.1.2 Schéma de données spatio-temporel 119
6.1.3 Contextes de présentation 121
6.2 Mécanisme de médiation 122
6.2.1 Extracteurs de données 122
6.2.2 Serveur de présentations 123
6.3 Adaptateurs 123
6.3.1 Générateur JMF 125
6.3.2 Générateur SMIL 127
6.4 Expérimentation 129
6.4.1 Application touristique 129
6.4.2 Support à la présentation des données pour le décisionnel . . . 134
6.5 Conclusion 139

7 Conclusions et perspectives 141
7.1 Bilan du travail effectué 141
7.1.1 Infrastructure de spécification des gestionnaires de présentations 141
7.1.2 Modèle spatio-temporel de présentations 142
7.1.3 Le langage OQLiST 143
7.1.4 Implantation et expérimentation 143
7.1.5 Contributions majeures 144
7.2 Perspectives 145
7.2.1 Modèle 145
7.2.2 Langage 145
7.2.3 Indexation 146

7.2.4 Vers des services multimédias adaptables 146

Bibliographie **148**

A Définitions des opérateurs OQLiST **161**

B Présentations **187**

Table des figures

1.1 Présentation spatio-temporelle 2

2.1 Sélection de documents pertinents [Sub98] 13
2.2 Structure R-tree 16
2.3 Relations d'intervalles 24
2.4 Relations topologiques 25
2.5 Relations directionnelles 26
2.6 Modules principaux d'un environnement multimédia 27
2.7 Architecture générale d'un SGBDM 33

3.1 Ombre 43
3.2 Ombre spatio-temporelle 43
3.3 Présentations atomiques 47
3.4 Configuration **beginning** 52
3.5 Configuration **centered** 53
3.6 Configuration **end** 53
3.7 Configuration **middle** 53
3.8 Configuration **meet** 53
3.9 Description inter-média 54
3.10 Description intra-média 56

4.1 Objets 65
4.2 Présentation par défaut de deux collections 66
4.3 Vidéo découpée 67
4.4 Schéma de présentations 68
4.5 Éléments des présentations 69
4.6 Sélection des éléments 70
4.7 Relations dans une présentation composite 71

4.8 Opérateur border 73
4.9 Les opérateurs border et background 74
4.10 Opérateur table fit 76
4.11 Opérateur arrange all 78
4.12 Présentation référence et présentation cible 79
4.13 Présentations disjointes de deux images 81
4.14 Présentations superposées de deux images 84
4.15 Présentations superposées de deux vidéos 85
4.16 Présentation inverse 87
4.17 Types de requêtes 88

5.1 Applications multimédias 96
5.2 Intégration d'applications et de sources 97
5.3 Architecture générale d'un gestionnaire de présentations 98
5.4 Schéma de données d'une application 100
5.5 Présentation par défaut d'une image 101
5.6 Utilisation d'un gestionnaire 102
5.7 Adaptateur 104
5.8 Spécification d'une présentation 104
5.9 Arbre d'une présentation 106
5.10 Adaptation de présentations 108
5.11 Récupération de données 108
5.12 Architecture de JAGUAR 110
5.13 Modèle de présentations de JAGUAR 111
5.14 Types prédéfinis dans le modèle de JAGUAR 113
5.15 Processus de génération 115

6.1 Architecture du gestionnaire 118
6.2 Schéma de données du gestionnaire 120
6.3 Extracteurs de données 123
6.4 Serveur de présentations 123
6.5 Architecture générale d'un adaptateur 124
6.6 Générateur JMF 125
6.7 Générateur SMIL 127
6.8 Description de la carte de France 130
6.9 Noms des régions de la France 131

6.10 Stations de ski et activités 132
6.11 Présentations enchaînées 133
6.12 Schéma en étoile des données météorologiques 135
6.13 Cube METEO 135
6.14 Schéma de présentations d'un cube 136
6.15 Présentations des éléments d'un cube 136
6.16 Présentations des plans d'un cube 138
6.17 Opération roll-up 139
6.18 Opération pivot 139

B.1 Espace de configurations 189
B.2 Stations de ski et activités 190
B.3 Composition de présentations avec des cadres 195
B.4 Le soleil et ses planètes 196
B.5 Le système solaire 197
B.6 Composition des tableaux 197

Liste des tableaux

3.1 Types des présentations 46
3.2 Prédicats 48
3.3 Relations topologiques 50
3.4 Relations directionnelles 50
3.5 Ombre et relations spatiales 51

4.1 Prédicats temporels 70
4.2 Prédicats spatiaux 71
4.3 Organisation de collections 77
4.4 Premier groupe d'opérateurs spatiaux 80
4.5 Deuxième groupe d'opérateurs spatiaux 82
4.6 Troisième groupe d'opérateurs spatiaux 84
4.7 Relations temporelles caractérisées 86
4.8 Relations temporelles non caractérisées 87

Chapitre 1

Introduction

1.1 Systèmes et environnements multimédias

La manipulation digitale du texte, des images, du son et de la vidéo (*médias*) sur des ordinateurs a changé la nature d'un grand nombre d'applications. En particulier, les systèmes et les applications habituellement dédiés à la gestion de données classiques (chaîne de caractères, entier, réel) trouvent dans ces nouveaux types de données des nouvelles possibilités : modélisation, stockage, récupération, visualisation, etc.

Un système est dit multimédia lorsqu'il fournit les mécanismes nécessaires pour manipuler plusieurs médias (par ex. son et vidéo) et pour les intégrer avec des données classiques. L'information multimédias est plus riche en ce sens qu'un média a un contenu, une structure et une sémantique bien particulière qui est perçue par un utilisateur humain. De plus, ces médias peuvent aussi être intégrés dans des documents pouvant d'ailleurs être reliés entre eux pour constituer des réseaux à l'intérieur desquels on peut naviguer par des hyperliens comme dans le Web. Une présentation ou document multimédia intègre et synchronise des médias avec des données classiques.

Les médias et les présentations sont souvent traités sous l'angle de la représentation de leur contenu (structure logique, temporelle, spatiale), du placement et de la synchronisation. Des travaux ont également abordé leur exploitation à travers des mécanismes permettant d'assurer l'exécution des présentations, leur réutilisation, leur interrogation efficace et leur stockage.

Selon les besoins applicatifs, les utilisateurs veulent récupérer ces données, les transformer (réduire, changer la palette de couleurs), les combiner, réutiliser tout ou partie des médias pour construire des présentations.

La gestion des documents multimédias implique aussi la spécification de mécanismes pour gérer la récupération et la synchronisation d'objets distribués, le partage de l'information, etc. Les couches logicielles qui supportent les outils et les applications multimédias (i.e., les bases de données, le réseau) doivent être adaptées pour transporter et stocker les données selon leurs caractéristiques propres en prenant en compte les besoins des applications.

À l'heure actuelle, il existe des produits logiciels de tout genre pour spécifier, traiter et gérer des données multimédias. Le spectre de logiciels est large et varie entre des produits qui permettent la composition de médias [CC96, VKS99], l'édition de documents [Lay97], la reconnaissance du contenu [SM97] jusqu'à des systèmes de gestion de bases de données multimédias (SGBDM) supportant le stockage, la récupération efficace, l'interrogation des données [TVS96, Mar96a, LOSO97, OTCH98, LM98].

1.2 Présentation multimédia et SGBD

La notion de présentation est centrale à notre travail. Elle est introduite ici au moyen d'un exemple. Considérons une présentation de sites historiques de France, qui consiste en plusieurs images affichées de manière séquentielle synchronisées avec une explication vocale. La carte de France sert de repère pour positionner les images à des endroits précis de l'écran. La figure 1.1 décrit cette présentation en utilisant Grenoble comme exemple. Les aspects spatiaux sont décrits à gauche et les aspects temporels à droite.

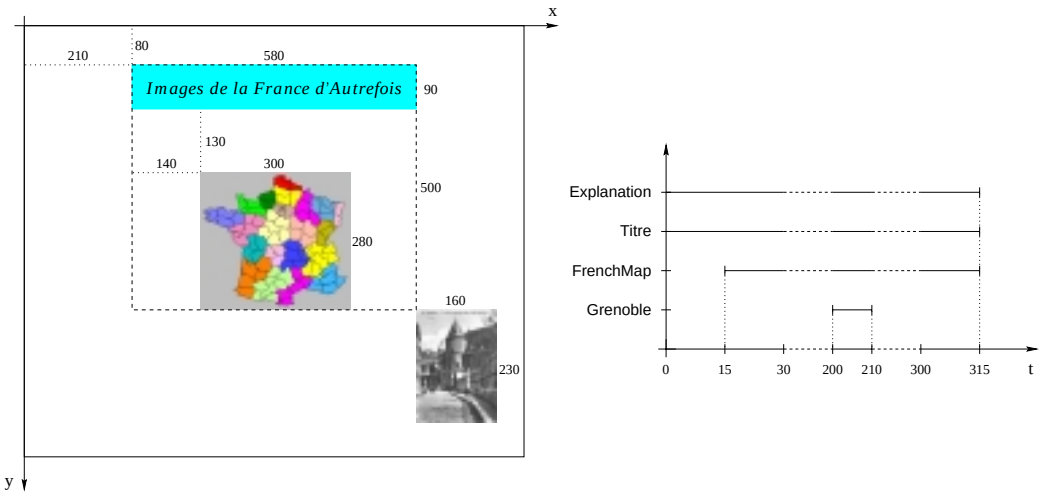


FIG. 1.1 – *Présentation spatio-temporelle*

Un titre est affiché à la position (210, 80), sa présentation démarre en même temps que l'explication. La carte est affichée 15 secondes plus tard, à 140 pixels du côté gauche du titre et à 130 pixels de son côté bas. À la seconde 30, une présentation séquentielle des images historiques de la France démarre. La position d'une image est déterminée par la région concernée. Par exemple l'image de Grenoble est présentée à la position (790, 580) qui correspond au sud-est de la France.

Ainsi, une présentation est une application interactive qui doit être exécutée en respectant des contraintes spatiales et temporelles. Pour définir une présentation, il faut (1) spécifier la configuration spatiale et temporelle des objets qui la composent et (2) décrire la façon dont ces objets doivent être manipulées : afficher, modifier, parcourir, ...

Pour mettre en oeuvre une présentation, il faut représenter ses attributs (structure, configuration spatiale et temporelle) et son contenu (structure des objets composants, format, source, taille, volume). Ces objets peuvent soit être référencés directement à travers une adresse URL, soit être le résultat d'une requête ou encore être obtenus en temps-réel. A priori, il n'existe pas de moyen simple pour connaître le format des données qui composent une présentation. C'est alors la responsabilité de l'utilisateur de prévoir dans son application la mise en forme des données avant de les présenter.

Depuis plusieurs années, des standards tels que MHEG [EMM+98], PREMO [ISO94], SMIL [W3C98b] sont apparus. Il s'agit de langages de spécification de présentations multimédias qui sont associés à des environnements de compilation et d'exécution. Ces environnements permettent d'obtenir des documents pouvant être joués, par exemple sur des navigateurs. La présence de différents standards a aussi provoqué une diversité de documents et de présentations, qui sont, de notre point de vue, des données multimédias. Ces données sont souvent stockés dans des bases spécialisées, et il serait souhaitable de pouvoir les récupérer, les combiner pour construire des nouvelles présentations.

Des travaux dans le domaine de bases de données ont proposé des extensions pour la construction de SGBDM [OSEV95, Vaz96, Adi96, MS96]. En particulier, notre travail est basé sur le SGBDM STORM qui est une extension du SGBD O_2. Une première version a été réalisée dans le cadre des travaux de [Moc97, Loz00] en utilisant le modèle temporel proposé par [Adi96].

1.3 Problématique et objectifs du travail

La diversité et l'hétérogénéité de sources de données multimédias met en évidence la nécessité de proposer des modèles et des stratégies d'intégration permettant d'exploiter leur contenu et leur structure. Ceci permet d'une part la réutilisation de ces données, d'autre part, ceci facilite leur gestion.

La problématique de notre travail s'articule alors autour de deux problèmes :

1. la modélisation de présentations intégrant des médias et des données classiques. Il s'agit de caractériser les aspects spatiaux et/ou temporels des médias pour les représenter de manière homogène. Il faut également caractériser la composition des médias dans l'espace et/ou le temps au travers de relations. Il faut enfin disposer d'un langage *ad-hoc* pour tout cela.

2. la spécification d'une infrastructure et de mécanismes nécessaires pour la gestion de présentations (définition, stockage, récupération, réutilisation). Ces mécanismes doivent fournir un support pour modéliser et construire des applications qui ont besoin d'accéder et d'exploiter des données stockées dans des sources hétérogènes et réparties.

Nous nous focalisons sur les modèles et les systèmes de bases de données à objets pour les présentations spatio-temporelles multimédias. Notre objectif est d'en définir une représentation générale en considérant les aspects spatiaux et temporels. Cette représentation est indépendante des langages de définition et des types de données. Nous proposons une représentation où ces aspects peuvent être combinés pour résoudre différents besoins, à savoir :

- La manipulation et la réutilisation des présentations spatio-temporelles composées de médias et de documents hétérogènes.

- L'intégration de données indépendamment des formats de représentation.

- La construction de présentations indépendamment des plates-formes d'exécution.

Nous abordons le problème de l'intégration des objets multimédias, à la fois du point de vue modèle, représentation, langage et architecture logicielle. Nous avons

construit des outils pour stocker et pour interroger les objets et les présentations multimédias, en introduisant trois aspects novateurs :

- homogénéisation de la représentation des aspects spatiaux et temporels : nous avons spécifié un modèle pour représenter les objets et les présentations multimédias comme des objets de première classe.

- définition et intégration d'opérateurs spatio-temporels à travers un langage nommé OQLiST (*Object Query Language integrating Spatial and Temporal aspects*) qui est une extension d'OQL pour manipuler les objets et les présentations.

- proposition d'une infrastructure pour intégrer applications et données hétérogènes distribuées. Nous utilisons une approche à objets pour intégrer dans des présentations spatio-temporelles, des objets multimédias provenant de sources hétérogènes et réparties sur un réseau général (Internet, par exemple).

1.3.1 Modèle et langage

Une présentation multimédia est à la fois pilotée par les données qui la composent (médias), par les enchaînements temporels et spatiaux et par les flux de contrôle (interaction, navigation). La modélisation des présentations doit donc prendre en compte :

- les caractéristiques des médias, au-delà des formats et des standards sous lesquels ils sont digitalisés. Un modèle doit permettre une vision homogène des médias et prendre en compte aussi bien des médias comme l'audio qui sont purement temporels, ainsi que des images qui sont intrinsèquement spatiales. Enfin, la modélisation doit aussi prendre en compte des attributs faisant référence au contenu, couleur, texture, etc.

- les opérations associées à chaque type de média : elles concernent à la fois la modification des caractéristiques spatiales et temporelles (réduire, agrandir, couper), la récupération d'une partie (récupérer un segment ou une région), mais aussi leur exécution, par exemple jouer, afficher, mettre en attente.

- la représentation des caractéristiques de l'espace où les médias sont affichés : les dimensions de l'espace 1D, 2D, 3D, 4D et les relations temporelles et spatiales entre les médias.

Modèle spatio-temporel

Nous avons spécifié un modèle qui propose des concepts pour caractériser les attributs spatiaux et temporels des objets (par ex. la taille, la durée) et leur position dans l'espace et dans le temps. La position est caractérisée de manière absolue ou de manière relative en utilisant des relations spatio-temporelles [AZ99a, AZ99b].

Dans notre modèle, les dimensions sont considérées comme un continuum où chaque objet est caractérisé par ses projections sur les dimensions spatiale et/ou temporelle. Les projections sont caractérisées par des intervalles décrivant la position et la taille d'un objet sur une dimension. Dans un espace à n dimensions, un objet est représenté comme une combinaison de n intervalles.

Une présentation peut être atomique (par ex. afficher une image) ou composite (par ex. afficher un ensemble d'images de Paris et Grenoble séquentiellement à la coordonnée 5, 80). La composition permet de traiter à la fois des présentations inter-médias mais aussi des descriptions intra-médias. Pour ces dernières, un objet peut être considéré comme une présentation composite qui décrit le contenu de l'objet.

Langage de construction et d'interrogation de présentations spatiales et temporelles

Nous avons spécifié OQLiST comme une extension du langage OQL en intégrant des aspects spatiaux et temporels pour la construction, l'interrogation et la manipulation de présentations. Ce langage fournit :

- des constructeurs de présentations spatiales, temporelles et spatio-temporelles d'objets (image, son, vidéo, texte, document).

  Les constructeurs sont associés à des opérateurs tels que *crop* permettant de récupérer des segments et des régions d'un objet pour créer une présentation.

- des opérateurs pour exprimer des présentations spécifiant les positions absolues et relatives des objets.

Les présentations sont aussi construites comme des résultats de requêtes avec des caractéristiques spatio-temporelles implicites ou explicites. OQLiST permet de réutiliser ces caractéristiques pour définir des résultats de requêtes avec des nouvelles dispositions spatiales et temporelles. Par exemple récupérer un ensemble d'images de sites historiques français et les présenter sous forme de tableau où chaque image est affichée pendant 25 secondes.

1.3.2 Vers une infrastructure d'intégration d'applications et de données multimédias

Les médias (texte, image, son, vidéo) sont d'aujourd'hui présents dans la plupart des applications. Cependant, ils sont souvent traités comme des "gros" objets sans véritablement prendre en compte toutes leurs caractéristiques ni toute la sémantique dont ils sont porteurs. Par exemple l'accès par le contenu reste un problème fondamental qui demande encore des efforts de recherche et de développement. De plus, les caractéristiques spatiales et/ou temporelles des données de ce type ne sont pas toujours prises en compte par les serveurs de données.

Nous avons vu apparaître les systèmes de type médiateur [Wie92] et des mécanismes spécialisés pour l'interrogation de données [SL90, BBE99, CG99, MRJ99, BCD+99]. L'objectif général de ces systèmes est de fournir un moyen pour accéder de manière unifiée à des données/informations stockées dans diverses sources d'information. Ces sources peuvent être accédées via des systèmes relationnels ou à objets, des moteurs de recherche (par ex. AltaVista, NorthernLight, etc.) ou, bien directement (pages HTML ou XML). Nous sommes donc aujourd'hui en présence de sources d'information fortement distribuées, hétérogènes et autonomes que l'on veut interroger.

La diversité des solutions proposées permet d'avoir des capacités de traitement et de gestion des données de plus en plus complexes et efficaces. Néanmoins, l'intégration des données et des applications devient plus compliquée : chaque outil repose sur son propre modèle de représentation des données et d'exécution de présentations. Face à cette diversité, il semble nécessaire d'avoir des infrastructures servant d'intermédiaires entre les données et les applications multimédias. Leur rôle est de fournir un modèle de représentation et de programmation général pour cacher l'hétérogénéité des données et des applications.

Nous avons spécifié et implanté une infrastructure permettant la spécification de gestionnaires de présentations. Un gestionnaire est un médiateur entre des données et des applications multimédias. Il est configuré pour accéder aux données des sources prédéfinies (bases de données, Web, des serveurs, des fichiers, etc.) et pour interagir avec des applications qui ont leurs propres langages de définition. Le gestionnaire intègre les données dans des présentations qui sont ensuite évaluées et mises en forme selon les caractéristiques des plates-formes d'exécution utilisées pour les applications.

Pour configurer un tel gestionnaire, une application doit spécifier les types d'objets qu'elle utilise à travers un schéma de données. Ces données sont associées à des pré-

sentations par défaut, décrites dans un contexte. Une même application peut utiliser plusieurs contextes de présentation selon la façon dont elle souhaite visualiser les données stockées dans des sources. Les interactions d'un gestionnaire avec les applications et les sources doivent aussi être spécifiées.

À partir d'un schéma de données, des spécifications d'interactions et au moins un contexte de présentation, notre infrastructure (1) génère une interface et une librairie de classes permettant aux applications d'interagir avec le gestionnaire ; (2) configure le gestionnaire pour qu'il prenne en compte les données utilisées par les applications.

1.4 Validation expérimentale

Pour valider notre contribution, nous avons implanté une gestionnaire prototype pour l'intégration de données et applications [ZA00, AZ00]. Le prototype est un gestionnaire de présentations adapté pour les environnements d'exécution de présentations multimédias JMF (*Java Media Framework* [SI]) et SOJA [Fou] (basé sur le standard SMIL). Nous avons utilisé et étendu un SGBD multimédia basé sur O_2 [Adi96, ALMM97, Mar97, LM98]. Nous avons implanté notre modèle spatio-temporel dans un schéma bases de données ainsi que des fonctions pour permettre la gestion (définition, interrogation et exécution de présentations) de présentations spatio-temporelles sur ce SGBD.

Le gestionnaire est un environnement de type client (Web) et serveur pour construire, stocker, interroger et exécuter des présentations multimédias incluant les types d'objets texte, image, son et vidéo. Outre le serveur de données qui est O_2, le prototype est basé sur des langages et composants normalisés (SMIL, Java, Perl, ImageMagick, JMF, SOJA). Par ces choix techniques, l'ouverture sur le Web est obtenue d'emblée et nous pouvons construire des présentations spatio-temporelles multimédias en y intégrant des objets qui sont identifiés par une adresse URL, ce qui permet une grande généralité de notre approche.

L'expérimentation de notre travail s'est faite dans deux directions :

- Utilisation de notre prototype pour valider la spécification de présentations avec le langage OQLiST. Nous avons implanté une application touristique qui permet de visualiser des données et des documents multimédias.

- Visualisation de données pour le décisionnel. Nous avons implanté un système OLAP pour l'interrogation et la visualisation interactive des données multimé-

dias stockées dans un entrepôt et concernant la météo. En particulier, nous avons défini des constructeurs adaptés pour la présentation de données multi-dimensionnelles [Cod93] (cubes).

1.5 Organisation du document

La suite de ce document est organisée de la manière suivante :

- Le chapitre 2 est une analyse de l'état de l'art concernant les données multimédias. Nous présentons tout d'abord les caractéristiques des types texte, image, son et vidéo. Nous analysons la représentation et les techniques d'interrogation associés aux bases de médias. Nous abordons les différents aspects à considérer pour construire des systèmes intégrant ces données avec des données classiques. Nous introduisons ensuite les différentes couches logicielles nécessaires pour construire des applications multimédias. Nous décrivons, également les caractéristiques et fonctionnalités des SGBD multimédia. Enfin, nous analysons les standards en mettant en évidence les aspects de données et des fonctions associées qui sont standardisées.

- Le chapitre 3 présente notre modèle qui fournit des concepts pour décrire les attributs temporels et spatiaux des médias. Le modèle permet aussi de représenter des relations entre objets pour décrire des présentations. D'autres attributs plus spécifiques à un média (par ex. la couleur) peuvent également être représentés. Notre modèle permet de représenter des présentations atomiques et composites (i.e., description inter-média). Cette approche permet à la fois de décrire les liens inter et intra-médias.

- Le chapitre 4 décrit le langage spatio-temporel OQLiST qui permet la spécification et l'interrogation déclaratives de présentations spatio-temporelles. Nous présentons les opérateurs du langage : constructeurs, opérateurs temporels et spatiaux. Une taxanomie des requêtes pouvant être exprimées avec ce langage est également présentée.

- Au chapitre 5, nous présentons une infrastructure de spécification de gestionnaires de présentations JAGUAR. Un gestionnaire de présentations implante une couche médiatrice entre des applications et des sources hétérogènes et réparties.

Tout d'abord, nous introduisons la gestion de présentations en utilisant un gestionnaire qui intègre des données hétérogènes pour construire des présentations pouvant être exécutées sur des plates-formes différentes. Nous présentons ensuite ses fonctions principales : la transformation, la construction et la génération des présentations. Nous décrivons également la mise en place d'un gestionnaire avec JAGUAR. En effet, notre infrastructure permet de configurer et de générer des gestionnaires pour un ensemble d'applications et des sources. Enfin, nous discutons les avantages et limitations de notre approche.

- Le chapitre 6 décrit une implantation de JAGUAR qui a été faite. Nous présentons tout d'abord la plate-forme logicielle adoptée pour construire le gestionnaire. Nous présentons l'architecture de notre infrastructure et des gestionnaires, ainsi que les fonctions implantées. Ensuite, nous décrivons une implantation d'un médiateur entre des sources hétérogènes et des applications tournant sur des plates-formes JMF et SMIL. Finalement, nous présentons les expérimentations que nous menons concernant d'une part, l'implantation des applications de visualisation de données et, d'autre part, la spécification de présentations pour le décisionnel (cubes de données).

- Le chapitre 7 conclut ce document en mettant l'accent sur les apports de notre travail et en donnant quelques perspectives pour des recherches futures : utilisation de valeurs indéterminées pour la modélisation de l'interaction dans les présentations, spécification d'algorithmes et de mécanismes d'indexation adaptés pour les données multimédias et les présentations, extension du langage pour la définition de présentations dans des espaces à quatre dimensions, et spécification d'un service multimédias adaptable.

Chapitre 2

Données et systèmes multimédias

Dans ce chapitre, nous caractérisons tout d'abord les données multimédias et les opérations qui y sont associées. Nous nous intéressons aussi aux techniques utilisées pour les représenter et les exploiter. Ensuite nous étudions les systèmes multimédias proposés dans les domaines bases de données, systèmes ainsi que les standards disponibles à l'heure actuelle.

L'objectif de notre étude est d'analyser la façon dont les données multimédias sont traitées dans la technologie existante pour dégager les approches utilisées, leurs particularités et leurs limitations. La section 2.1 caractérise les données multimédias et les compare par rapport aux données classiques. La section 2.2 décrit les modèles de présentation des données et des documents multimédias. Des environnements pour la programmation d'applications multimédias sont décrits dans la section 2.3. Ensuite, la section 2.4 présente les systèmes de gestion de bases de données multimédias. La section 2.5 décrit les différents standards proposés à l'heure actuelle. Enfin, la section 2.6 conclut ce chapitre.

2.1 Données multimédias

Il existe un certain nombre de types de données (texte, image, son et vidéo) reconnus comme des médias [Sub98]. Une des caractéristiques de ces types de données est leur grand volume : un texte ASCII de 500 pages représente un volume 1MB, 500 images noir et blanc 32MB, la même quantité d'images sous format GIF, TIFF, 1,6MB, 5 minutes de son sous format MPEG, 0,2MB, 5 minutes de vidéo haute qualité, 33GB. Ces gros volumes de données posent le problème de leur gestion, du stockage et de leur transfert en tenant compte de leurs spécificités. Cette section

caractérise ces types de données et les problèmes associés à leur représentation et à leur exploitation.

2.1.1 Texte

Représentation Le texte est un média souvent représenté comme une liste de chaînes de caractères, correspondant à un document entier ou à une partie, par exemple le titre ou le résumé. Une base de documents est organisée comme un index de chaînes de caractères (par ex. index de titres, de mots clés). Des requêtes contenant des mots clés peuvent ensuite être évaluées sur ce type de base. Une telle évaluation revient à chercher toutes les entrées de l'index qui contiennent les mots clés d'une requête.

Des travaux comme [OSEV95] proposent des représentations arborescentes de la structure des textes : titre, auteurs, chapitres, sections, etc. En même temps que cette approche permet effectivement de caractériser le contenu du texte, elle suppose des structures fixes qui peuvent devenir insuffisant lorsque d'autres types de textes doivent être considérés. Par ailleurs, des travaux comme [ISO97, QV97, GQV98] représentent la mise en page des textes. En utilisant ces modèles, des opérations de recherche sont définies sur la structure logique et sur la mise en page d'un texte.

Recherche d'information L'interrogation de bases de textes revient à analyser le contenu d'un ensemble de textes afin de vérifier si un ou plusieurs termes y sont présents. Deux aspects doivent être pris en compte pour interroger le contenu d'un document :

- Synonymie, il s'agit de prendre en compte des mots clés d'une requête mais aussi des termes avec des significations proches (par ex. serpent, reptile). Des critères plus fins sont alors considérés pour indexer les documents dans une base. Par exemple indexer les documents par rapport à des termes connexes.

- Polysémie, souvent la signification des mots est fortement dépendante du contexte. Par exemple le mot *intérêt* dans un contexte financier, fait référence aux bénéfices obtenus sur une somme d'argent épargnée. Alors que dans un article scientifique, par exemple le même mot fait référence à la pertinence d'une proposition. L'interrogation de bases de textes doit prévoir des techniques de raffinement des requêtes pour permettre de préciser le contexte de référence des mots clés d'une requête.

La plupart des travaux prennent en compte la synonymie et la polysémie à l'aide de deux mesures, la *pertinence* et la *précision*. Ces mesures sont associées aux résultats de requêtes et en général aux algorithmes d'évaluation de ces requêtes.

Considérons D, un ensemble fini de documents et A, un algorithme qui prend un sujet s en entrée et qui retourne un ensemble de documents A(s) tel que $\mathsf{A(s)} \subseteq \mathsf{D}$. Supposons aussi l'existence d'un prédicat $\mathsf{P}_r(\mathsf{s}, \mathsf{d})$ où s est un sujet et d un document. $\mathsf{P}_r(\mathsf{s}, \mathsf{d})$ est vérifié si d est pertinent par rapport à s. Le prédicat est spécifié considérant un ensemble de documents $\mathsf{D}_{Test} \subseteq \mathsf{D}$ et un ensemble de sujets S_{Test}.

La précision de l'algorithme A par rapport au prédicat P_r est l'ensemble de documents D_{Test} et le pourcentage $\mathsf{P}_s\%$ pour le sujet $\mathsf{s} \in \mathsf{S}_{Test}$ si et seulement si :

$$P_s = 100 \times \frac{1 + card(\{d \in D_{Test} | d \in A(s) \ \wedge P_r(s,d) = vrai\})}{1 + card(\{d \in D_{Test} | d \in A(s)\})}$$

Par ailleurs, la pertinence revient à répondre à la question : combien de documents récupérés sont pertinents ? Intuitivement, il faut compter tous les documents dans l'intersection par les deux régions (documents retournés par l'algorithme de recherche et les documents pertinents) dans la figure 2.1 et diviser entre le nombre de documents pertinents (région hachurée). La formule suivante interprète cette intuition :

$$R_s = 100 \times \frac{1 + card(\{d \in D_{Test} | d \in A(s) \ \wedge P_r(s,d) = vrai\})}{1 + card(\{d \in D_{Test} | P_r(s,d) = vrai\})}$$

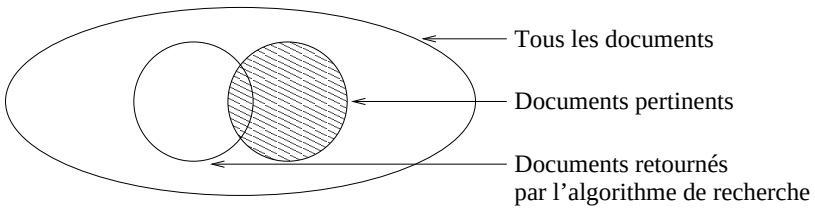


FIG. 2.1 – *Sélection de documents pertinents [Sub98]*

Des techniques d'indexation de textes ont été proposées à l'heure actuelle. La plupart des approches telles que LSI (*Latent Semantic Indexing*), *TV-trees* (*Télescoping Vector Tree*) et les fichiers signés manipulent des vecteurs des termes clés.

L'idée de base de LSI est de considérer que les documents similaires ont des fréquences similaires de mots. Cette technique associe un vecteur vec(d) des fréquences de termes dans un document d. Un document est alors représenté par un identificateur associé à un vecteur de termes (posting_list) apparaissant dans le document.

Les termes dans une liste posting_list sont ordonnés par rapport à une mesure de pertinence préétablie.

Un vecteur d'un terme est une liste représentée par un identificateur associé à une liste de documents où un terme apparaît (document_list). Deux tables de *hash* Doc_Table et Term_Table sont utilisées pour évaluer des requêtes. Ainsi, pour récupérer *tous les documents concernant un terme*, il faut retourner la document_list associée à ce terme. Pour récupérer *tous les documents concernant un ensemble de termes*, il suffit de calculer l'intersection entre document_list et posting_list.

Avec l'approche *TV-trees*, un document est représenté par un arbre de vecteurs de termes. En utilisant l'approche *TV-trees*, lorsqu'un utilisateur veut récupérer tous les documents traitant un certain sujet, il spécifie une requête Q contenant un ensemble de mots clés. Q est aussi considérée comme un document pouvant être décrit par un vecteur de termes $\mathsf{vec}(\mathsf{d}_q)$. Évaluer Q revient à chercher un *TV-tree* afin de trouver les n voisins les plus proches de $\mathsf{vec}(\mathsf{d}_q)$ par rapport à une distance prédéfinie.

Enfin, l'idée principale de l'approche par fichiers signés est d'associer à un document une signature s_d. Intuitivement, la signature représente une liste ordonnée de termes qui décrivent le contenu d'un document. La signature peut être dérivée comme le résultat d'une analyse de fréquence de termes. Les requêtes sur le contenu des documents sont évaluées sur leurs signatures.

Systèmes existants Le *DataBlade* Informix (*http://www.informix.com/datablades*) fournit des mécanismes pour la construction de bases de données textuelles. Par exemple le *Verity Text Search DataBlade Module* fournit des services pour le stockage, la fragmentation et la récupération de documents SGML. Ces mécanismes sont utilisés en coopération avec le serveur du SGBD.

Oracle (*http://www.oracle.com*) fournit le paquet *ConText* pour la récupération et la gestion de bases de textes. Intégré au SGBD, il permet d'exécuter des requêtes SQL sur les textes. Les requêtes peuvent porter sur les sujets des textes mais elles permettent aussi la création de résumés.

Le système DB2 (*http://www.software.ibm.com/data/db2/extenders/about.html*) fournit un paquet pour la gestion de textes qui permet d'exécuter des recherches sur des documents multi-langues selon des critères tels que les mots clés, synonymes et des variations de mots et de phrases.

2.1.2 Image

Le type image correspond à des dessins, de la peinture, de la photographie ou de l'impression. Des formats différents sont utilisés pour la digitalisation des images parmi lesquels GIF, PIC, etc. Des opérations de base ont été également proposés telles que le découpage, la correction chromatique et la composition de différentes sources d'images.

Représentation De manière générale, le contenu d'une image peut être modélisé comme un ensemble d'objets rélévants, décrits par leurs formes et leurs propriétés associées. Étant donné que les images peuvent représenter des volumes de données importants, des techniques de compression utilisant des transformations de Fourier et la segmentation ont été proposées. Ces techniques visent à réduire l'espace de stockage requis et le coût de manipulation de ce type de données, mais aussi à représenter leur contenu pour faciliter leur exploitation.

L'une de premières contributions concernant la description d'images faite par Grosky [Gro94] propose de schémas pour la représentation du contenu d'images. Dans des approches bases de données, le contenu des images est représenté par des schémas relationnels ou à objets, parfois ces modèles de données sont étendus pour prendre en compte les aspects spatiaux. Le contenu d'une image est donc représenté en spécifiant en plus la configuration spatiale de ses objets relevants.

Dans certaines approches, les objets sont souvent considérés comme des régions rectangulaires. Ensuite, des *R-trees* [Gut84] sont utilisés pour indexer les régions afin de permettre une récupération efficace des informations (cf. figure 2.2).

Interrogation En dehors des travaux qui proposent des extensions des langages de requêtes tels que SQL ou OQL pour l'interrogation et récupération d'images, d'autres travaux se focalisent sur la reconnaissance de formes. Des techniques telles que l'approche métrique et l'approche transformation ont été proposées pour supporter la récupération d'images basée sur la similarité du contenu [Sub98]. Dans la première approche, étant donnée une image i_1 et une distance d qui permet de comparer des images, une image i_2 est plus similaire à i_1 qu'une troisième image i_3 si la distance d entre i_1 et i_2 est plus petite que celle entre i_1 et i_3. Dans la deuxième approche, les critères de similarité sont spécifiés par un utilisateur sur un ensemble de types d'images. Un algorithme de similarité prend en compte ces critères pour comparer

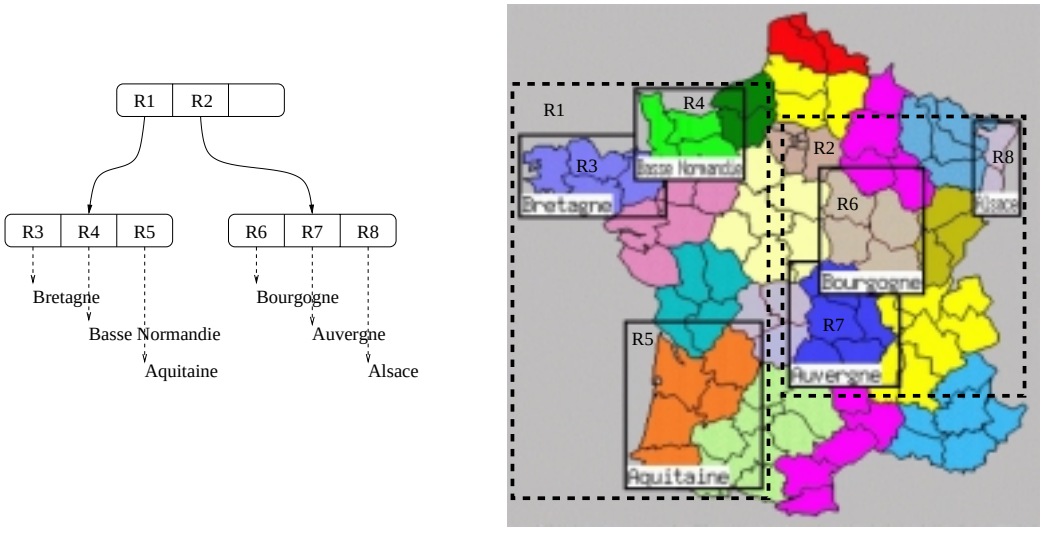


FIG. 2.2 – *Structure R-tree*

des images.

Systèmes existants Un certain nombre de bases d'images commerciales existent à l'heure actuelle et d'autres sont disponible sur le Web. Elles varient entre des gestionnaires de fichiers de type image jusqu'à des SGBD fournissant des langages étendus avec des opérateurs associées au type de données image.

Le *DataBlade* Informix (*Excalibur Image DataBlade Module*) utilise des techniques d'indexation spécialisées pour la récupération efficace d'images. Son langage de requêtes a été étendu avec des opérateurs tels que convertir, réduire (*scaling*), tourner et combiner des images. La récupération des données peut se faire par rapport à des attributs tels que la forme, la couleur, la texture et la composition. Informix fournit un module de reconnaissance du contenu d'images.

Le système DB2 a une extension pour la gestion d'images QBIC (*Query by Image Content*) qui fournit un support pour la récupération d'images en utilisant des attributs comme la couleur et la texture. Avec cette infrastructure, un utilisateur choisit une image et demande ensuite que des images similaires soient retrouvées. Des systèmes comme QBIC [FSA+95] et CHABOT [OS95] ont été spécialisés pour la récupération d'images et de vidéos. Ils utilisent des annotations décrivant des attributs tels que la texture et la couleur mais aussi des algorithmes de reconnaissance pour récupérer des images.

2.1.3 Son

Le type son fait partie des types des données dépendants du temps. On peut représenter un son au travers de flux associés à une échelle de temps progressivement constante. Certains opérations de manipulation du contenu d'un son comme couper, copier, coller peuvent être définies mais également des opérations comme jouer, enregistrer, avancer, reculer.

Représentation Le son peut être essentiellement représenté de deux manières : en utilisant des métadonnées qui décrivent le contenu, ou en reconnaissant des caractéristiques utilisant des techniques de traitement du signal. Dans la première approche, la façon la plus simple est de définir des segments de son et de leur associer des annotations décrivant par exemple l'interprète, l'instrument, le nom du morceau, etc. De manière générale, les métadonnées utilisées pour décrire le son peuvent être vues comme des objets ordonnés sur une ligne de temps qui correspond au temps du son.

Dans l'approche basée sur le traitement du signal, un son est considéré comme un signal $\phi(\mathsf{x})$ sur le temps x. Différentes caractéristiques du signal ϕ sont extraites, indexées et stockées pour une récupération efficace. La stratégie la plus commune pour décrire le contenu d'un son avec cette technique est de considérer un signal sur le temps et de diviser dans des fenêtres de temps. Le signal peut avoir beaucoup de variations dans des fenêtres différentes. Néanmoins, si les fenêtres sont suffisamment petites, le signal peut paraître assez homogène dans chaque fenêtre. Le signal est dit homogène s'il a une amplitude, une longueur d'onde et une vitesse constantes. D'autres stratégies telles que l'extraction de caractéristiques (par ex. l'intensité, le volume, le rythme, etc.) peuvent être utilisées pour indexer le contenu d'un son.

Interrogation L'interrogation des bases de son est souvent dépendante des modèles de représentation et d'indexation utilisés pour les organiser. Elle peut se faire par rapport aux métadonnées qui décrivent le contenu. Par contre, l'utilisation des techniques basées sur la reconnaissance du contenu semblent plus intéressantes pour des sons qui ne peuvent pas être décrits facilement par des métadonnées, par exemple lorsque l'on ne connaît pas la sémantique des informations (les messages vocaux enregistrés dans un répondeur).

Systèmes existants La plupart des bases de sons utilisent des métadonnées pour représenter et organiser les données. Par exemple dans *http://www.discjockey.com* une

base de sons est mise à disposition. Elle permet de jouer des sons sélectionnés en se servant de métadonnées comme titre, artiste et année.

Nous constatons que peu de bases de sons ont été proposées. Informix peut être couplé à *Muscle Fish* (*http://www.musclefish.com*) qui permet la récupération basée sur le contenu de sons stockés. Ce module fournit également un navigateur de sons contenus dans la base. DB2 fournit un outil qui permet de laisser des messages vocaux dans des répondeurs. Il permet de construire et d'interroger des résumés de sons. Les requêtes peuvent porter sur le contenu, mais aussi sur le format du son et la date de mise à jour.

2.1.4 Vidéo

Les données de type vidéo sont de grandes consommatrices de ressources systèmes telles que l'espace sur support de stockage secondaire. A titre d'exemple, une heure de vidéo compressée au format MPEG-1 nécessite 675 Giga-octets pour une qualité VHS standard.

Représentation La sémantique d'une donnée vidéo est une combinaisons des informations contenues dans chacune ses images, de la façon dont ces images sont composées et, finalement, de l'interprétation faite par l'utilisateur qui la regarde.

De nombreux travaux s'intéressent à la modélisation de données vidéos [FSA+95, HS95, WDG95, HMS96, YY97, JME99, RTV99]. De manière générale, un modèle de représentation de la vidéo doit intégrer des opérations permettant :

- *La structuration hiérarchique.* Une vidéo étant aussi un document, elle peut être organisée hiérarchiquement. Les unités élémentaires de cette hiérarchie sont les images qui sont groupées pour former les concepts de base (les plans). Les plans sont groupés pour former des entités plus complexes et abstraites (scènes). Les scènes à leur tour, sont groupées en concepts de plus haut niveau (séquences) et ainsi de suite jusqu'à arriver au dernier groupe dont le concept résume toute l'idée d'une vidéo [Loz00].

- *La description du contenu.* L'information sémantique contenue dans une vidéo n'est pas directement accessible. Il faut donc utiliser des métadonnées pour faire une description du contenu d'une vidéo. Cette description permettra son interrogation.

Il est donc possible de suivre une approche descendante (du flux vidéo aux objets qui composent la vidéo) pour définir la structure et le contenu d'une vidéo stockée comme un flux continu d'images. La structure représente l'organisation et les relations entre les différents segments de la vidéo. Le contenu représente l'information sémantique véhiculée par la vidéo. Cette modélisation doit permettre d'extraire le contenu sémantique de la vidéo lors de la phase d'interrogation. L'aspect temporel des vidéos et notamment la notion de segments vidéo qui représentent un intervalle d'images rendent complexe cette modélisation [EJH+97, Mar00].

Interrogation La réutilisation de tout type de données (la vidéo comprise) commence par la récupération de données intéressantes. Pour cela, une base de données vidéo offre d'une part, un langage avec lequel nous pouvons formuler tout type de requêtes permettant d'exprimer l'ensemble des caractéristiques de la vidéo. D'autre part, un moyen permettant de manipuler le résultat de l'exécution d'une requête afin que l'utilisateur discerne assez facilement la pertinence du résultat.

Une base de données vidéo doit permettre l'exploitation des vidéos qu'elle stocke. La composition permet la création de nouvelles vidéos à partir de la combinaison de différents segments vidéo déjà existants dans la base de données. Ces nouvelles vidéos créent de nouveaux contextes avec les mêmes besoins (structuration, description et interrogation) que les vidéos d'origine. Le processus de composition peut être comparé avec celui permettant la création de présentations multimédias où les éléments de base ou monomédias sont combinés dans le temps et dans l'espace pour la création d'une présentation.

Systèmes existants Le *DataBlade* Informix fournit un ensemble de modules avec des outils pour la gestion de la vidéo telles que des métadonnées pour la description du contenu (*Video Foundation DataBlade Module*). Les métadonnées sont stockés dans le SGBD tandis que la vidéo elle même peut être stockée dans des disques différents.

Oracle fournit un serveur vidéo qui permet la récupération et la manipulation des vidéos. DB2 fournit un module d'extension pour importer et interroger des résumés vidéos. L'interrogation des résumés se fait grâce à des annotations.

2.1.5 Discussion

La modélisation de données est une nécessité lorsque l'information doit être informatisée. Cette modélisation ne s'arrête pas à un média donné, elle concerne aussi la composition des médias dans des documents et des présentations. Dans le cas des données multimédias (simples ou composites) il faut pouvoir à la fois modéliser la sémantique et la composition de ces données et la synchronisation qui exprime les relations entre les données.

Comme nous l'avons étudié dans les sections précédentes, les données multimédias par leurs caractéristiques de taille et de complexité ainsi que par leurs propriétés temporelles et spatiales posent de nombreux problèmes de modélisation et d'indexation. Nous cherchons notamment à améliorer l'identification de contenus et à augmenter le caractère déclaratif de la création de documents ou de présentations multimédias.

La technologie à objets fournit un cadre de représentation des types de données multimédias. Les techniques d'encapsulation, d'héritage et de classes imbriquées sont un support adapté pour la modélisation et la caractérisation de ce type de données. Des librairies de classes/types et des fonctions associées ont été définies [Moc97].

À l'heure actuelle, des travaux de recherche ont proposé des librairies et des schémas génériques qui permettent la modélisation des médias et des présentations ou documents. Les modèles proposent des concepts pour décrire les caractéristiques des médias (par ex. taille, durée, couleur, etc.), la composition des médias (par ex. une image et une vidéo sont présentées de manière séquentielle).

[Mar96b] propose une librairie de classes qui modélisent des objets multimédias (par ex. images cartographiques, vidéo et son). Des classes abstraites constituent une infrastructure et peuvent être matérialisées en classes concrètes adaptées aux caractéristiques matérielles et logicielles de la plate-forme cible. Les classes caractérisent des attributs tels que la taille, la durée, la couleur ; par contre des aspects spatiaux et temporels ne sont pas modélisés.

L'infrastructure proposé par [MS96, ASS99] permet de programmer des systèmes d'informations multimédias. Cette infrastructure fournit aussi un langage de prédicats qui permet l'évaluation de requêtes concernant des bases différentes. Des structures d'indexation spécifique permettent d'organiser les informations décrivant les médias. Des aspects spatiaux et temporels sont considérés pour décrire et interroger les données. L'interaction est aussi prise en compte à travers des événements qui reflètent l'état des objets dans une présentation en cours d'exécution.

La gestion des données ne s'arrête évidemment pas à la modélisation. Les caractéristiques de ces données (gros volumes de stockage, besoins de qualité de service pour la récupération et l'exécution) nécessitent des environnements d'exécution adaptés qui puissent permettre la construction des applications utilisant ces données.

2.2 Documents et présentations multimédias

Un document est "ce qui sert à instruire, à informer" (Dictionnaire *Le nouveau Petit Robert*). En effet, un document est un conteneur d'information qui peut avoir une structure associée. Par exemple une image, un son, une vidéo, un article scientifique sont des documents. Les travaux qui traitent la multimédia s'intéressent aussi à la façon dont plusieurs médias peuvent composer des nouveaux documents.

2.2.1 Présentation de médias

De manière générale, une présentation multimédia est une combinaison de plusieurs médias y compris des présentations dans un espace à n dimensions. Or, les types de médias n'ont pas tous forcement des caractéristiques temporelles ou spatiales intrinsèquement. Par exemple comment réaliser la présentation spatiale d'un son? Les modèles de présentations proposent la notion de *présentation* comme une abstraction pour associer des caractéristiques spatiales et/ou temporelles aux médias.

Présentation temporelle La composition de médias constituant une présentation est effectuée par rapport à une ligne temporelle. Cela veut dire que l'affichage de médias (leur présentation) s'organisent par rapport à une ligne de temps. Par exemple présenter une image de la carte de France à la seconde 0 de la présentation et une image de Grenoble à la seconde 10.

Présentation spatiale La composition de médias dans une présentation est faite par rapport à des positions dans un espace à deux ou à trois dimensions. Par exemple présenter une image de la carte de France à la position (350, 300) et l'image de Grenoble à la position (790, 580).

Présentation spatio-temporelle C'est une composition de médias dans le temps et dans l'espace. Par exemple présenter une image de la carte de France à la position (350, 300) sans délai et l'image de Grenoble à la position (790, 580) à la seconde 10.

2.2.2 Modèles de présentations temporelles

Plusieurs modèles de présentations ont été proposés à l'heure actuelle [Adi96, Lay97, Ker97, BF98]. Ces modèles se regroupent selon deux approches : orientées instants et orientées intervalles.

Approche par instants Une présentation $\mathcal{P}$ est caractérisée par ses instants de début et de fin. Les instants sont des points dans une ligne de temps dont l'origine correspond au début de la présentation $\mathcal{P}$.

Les instants peuvent aussi être ordonnés les uns par rapport aux autres avec des relations telles que $>$, $<$, $=$. Par exemple la fin ($\mathcal{P}_{1_{fin}}$) de la présentation de la carte correspond au ($\mathcal{P}_{2_{debut}}$) début de la présentation de l'image de Grenoble : $\mathcal{P}_{1_{fin}} = \mathcal{P}_{2_{debut}}$.

Approche par intervalles Dans une approche par intervalles, une présentation $\mathcal{P}$ est vue comme une composition d'intervalles. Dans les présentations temporelles, un composant est représenté par un intervalle qui caractérise la durée pendant laquelle il est visible dans $\mathcal{P}$. L'ordre de présentation de chaque composant est exprimé par des positions absolues et/ou relatives. Les positions absolues sont définies par rapport à un point de repère appelé *origine* qui correspond au début de la présentation $\mathcal{P}$.

Dans certains modèles temporels de présentations [Ker97] permettant l'expression de positions relatives, les intervalles sont atomiques. Définir une présentation revient à placer des présentations les unes par rapport aux autres selon de relations comme avant, après, pendant (cf. section 2.2.4). D'autres modèles représentent chaque intervalle par son début, sa fin et sa durée. D'autres modèles comme [Adi96, Moc97] expriment la composition de présentations en combinant de positions absolues et relatives.

2.2.3 Modèles de présentations spatiales

Les modèles de présentations spatiales [Ege94, PTSE95, PT97, SS99] se regroupent selon deux approches : orientées intervalles et orientées régions. La plupart de modèles que nous avons étudié représentent des espaces à deux dimensions, c'est-à-dire qu'ils ne représentent pas des volumes.

Approche par intervalles Un objet dans un espace à deux dimensions est abstrait comme une composition d'intervalles représentant ses projections sur les axes x et y.

Ceci revient à modéliser les objets comme des MBR (*Minimum Bounding Rectangle*). Par exemple une présentation de l'image de Grenoble à la position (790, 580) est représentée par deux intervalles [0, 790] sur x, [0, 580] sur y.

La composition de présentations revient à établir des positions absolues et relatives sur les deux axes. Les positions relatives sont exprimées avec des relations d'intervalles de la même forme que dans les modèles temporels. Par exemple présenter l'image de la carte de France à la position (350, 300) et l'image de Grenoble 15 pixels au-dessus et 10 pixels à gauche du côté droit de la carte.

Approche par régions Les présentations de médias sont représentées par des régions telles que les rectangles [PTSE95, PT97], les cercles [SS99] ou des régions irrégulières [Ege94]. Chaque région à un point de repère par exemple son centre, le coin supérieur gauche d'un rectangle, etc. La position absolue d'une région dans un espace correspond à la position de son point de repère dans l'espace où elle est affichée (par ex. l'écran, une fenêtre, une page HTML). Les positions relatives des régions sont définies par des relations spatiales telles que nord, sud, ouest, à gauche, en bas, loin, etc. L'interprétation de ces relations est définie par rapport à la forme de la région de représentation du modèle (cf. section 2.2.4). Précisons, enfin que les modèles par régions ont été proposés dans les domaine de bases de données géographiques et adaptés pour la spécification de présentations multimédias. Il y a des travaux également qui les utilisent pour la reconnaissance du contenu dans les images (par ex. les cartes).

2.2.4 Relations

Relations d'intervalles Ces relations sont utilisées dans les modèles basés sur la représentation par intervalles des objets. Les relations d'intervalles proposées par Allen [All83] (cf. Figure 2.3) sont souvent adoptées pour exprimer les relations entre deux objets.

Les relations possibles entre deux objets se réduisent à toutes les combinaisons de positionnement possible de deux intervalles sur une ligne droite orientée. Il s'agit de treize relations au total consistant en sept relations de base *before*, *meet*, *overlap*, *finish*, *during*, *start*, *equal* et de leurs relations inverses, l'égalité étant elle-même son inverse. Les treize relations se répartissent en deux classes, celle des relations de séquentialité et celle des relations introduisant le parallélisme.

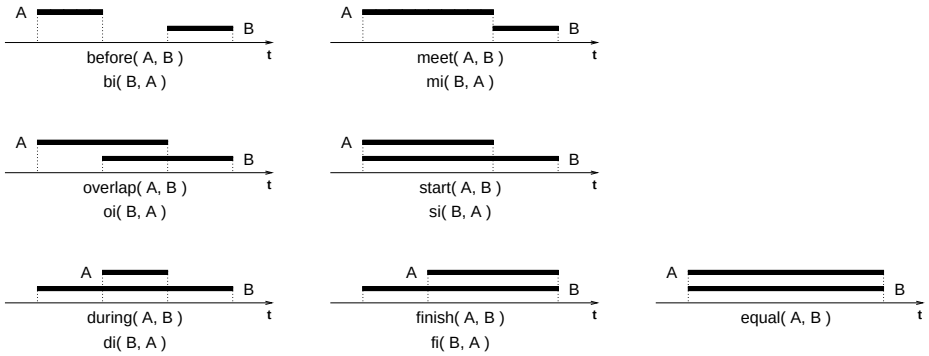


FIG. 2.3 – *Relations d'intervalles*

Relations à base d'instants Dans ces relations les unités temporelles considérées dans les relations sont les instants de début et de fin d'une présentation. Étant donnés deux instants, trois relations peuvent exister entre eux. Un instant peut en précéder un autre ($<$), lui succéder ($>$) ou lui être égal ($=$). Dans certains cas, les relations entre certains instants peuvent être imprécises, par exemple lorsque la fin d'une présentation est indéterminée.

Relations spatiales La composition spatiale vise à représenter la position relative entre les objets et leurs relations de direction. Les relations topologiques pouvant être établies entre des objets sont *disjoint*, *touch*, *overlap*, *cover*, *covered by*, *inside*, *contain*, *equal* (cf. figure 2.4). Un ensemble complet de ces relations a été proposé par [Ege94]. Étant donnés deux objets A et B huit relations sont définies :

1. A est disjoint de B (*disjoint*) ;
2. A touche B à l'extérieur (*touch*) ;
3. A chevauche B (*overlap*) ;
4. A couvre B (*cover* et son inverse *covered by*) ;
5. A contient B (*contain* et son inverse *inside*) ;
6. A et B coïncident dans l'espace (*equal*).

Les relations directionnelles entre des objets sont *north*, *south*, *east*, *west*, *northest*, *northwest*, *southeast*, *southwest*, etc. Un ensemble complet a été proposé par [PT97]

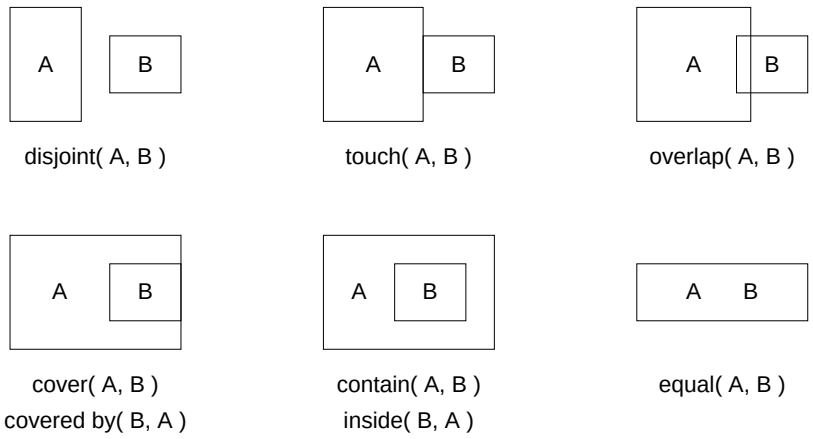


FIG. 2.4 – *Relations topologiques*

décrivant toutes les relations directionnelles possibles entre deux régions rectangulaires A et B (cf. figure 2.5). Pour cela, une région est représentée par deux intervalles, chacun correspondant aux axes x et y. Le total des relations directionnelles est obtenu en combinant tous les intervalles de A et B sur x et y. Puisque il y a 13 relations possibles entre deux intervalles sur chaque axe, il y a donc un espace de 13^2 relations directionnelles pouvant être représentées entre deux régions. L'ensemble des relations directionnelles {*north*, *south*, *east*, *west*, *northest*, *northwest*, *southeast*, *southwest*} est un sous-ensemble de cet espace de relations.

Enfin, les relations temporelles et topologiques ne sont pas suffisantes pour représenter certaines configurations. Par exemple, comment exprimer le fait que deux objets A et B sont disjoints de 8 cm. La relation *disjoint* capture leur position relative mais pas la distance entre les deux objets. Pour cela des travaux telles que [VTS98] proposent des modèles qui combinent l'expression de relations avec des valeurs qui précisent les distances relatives. Il est donc possible d'exprimer le fait qu'un objet A est affiché à la position (100, 100) et qu'un objet B est affiché 8 cm. à droite et 12 cm. au-dessous du côté haut de A.

2.2.5 Discussion

[VH95, Vaz96, VTS96, KVS97, VTS98] propose le modèle MOAP (*Multimedia Object and Application Model*). L'objectif est d'offrir un modèle de représentation de composition de médias pour supporter la construction de présentations. La représentation est basée sur les notions de composition (spatiale et temporelle) et la

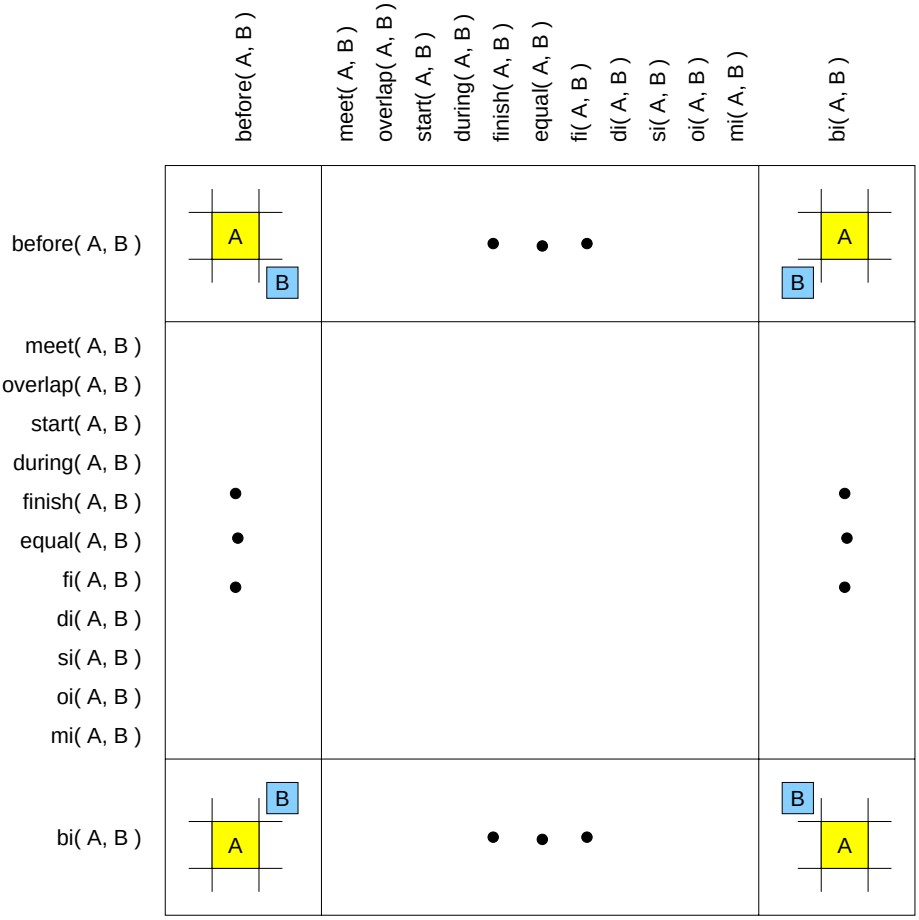


FIG. 2.5 – *Relations directionnelles*

modélisation des scénarios. La composition temporelle utilise les relations d'intervalles d'Allen, des relations spatiales (topologiques et directionnelles) sont utilisées pour la composition spatiale. D'autres aspects tels que la taille, la durée et la position des médias sont aussi modélisés.

[VKS99] propose une méthodologie pour l'édition des présentations multimédias. L'approche est basé sur la notion de présentation définie comme une chaîne d'objets média nommés acteurs. Ils fournissent aussi des outils interactifs d'aide à l'édition. Ces outils permettent d'avoir des vues différentes d'une présentation : disposition spatiale des fenêtres, disposition temporelle en indiquant les durées et les relations, exécution de l'application.

Finalement, nous constatons que les modèles à base d'intervalles permettent de mieux abstraire les composants d'une présentation et de les regrouper. La représentation à base d'intervalles fournit en très bon support pour l'exécution de présenta-

tions [Lay97]. Un telle représentation permet de considérer que l'état d'un système ne change pas de manière continue dans les temps mais qu'il évolue lors de l'apparition d'événements qui représentent le début ou la fin d'une présentation. Dans ce cas, il est possible de manipuler une structure de type graphe pouvant être exploitée comme une description des tâches à ordonnancer lors de l'exécution de la présentation.

2.3 Environnements multimédias

La mise en place d'une application multimédia nécessite l'intégration de techniques propres à trois domaines [Bre99, SD00] : le contenu, le transport et l'équipement. Le domaine du contenu de médias (textes, images, sons, vidéos) englobe des techniques d'acquisition, de traitement du signal et de création. Le domaine du transport correspond à ce qui est nécessaire à la transmission des médias. Le domaine des équipements se rapporte aux matériels et aux logiciels nécessaires à la conversion des contenus sous forme utilisable par des utilisateurs.

Un environnement multimédia [BCD93, FLMM98, Bre99, VKS99] est un ensemble de services, abstractions, librairies et logiciels intervenant dans la mise en place des modules constituant une application qui utilise des données multimédias. L'utilisation des différents services se fait par le biais d'interfaces à des API logicielles. L'ensemble des services utilisés par un tel environnement peut se diviser en modules de communication, système d'exploitation, bases de données et boites à outils (cf. figure 2.6).

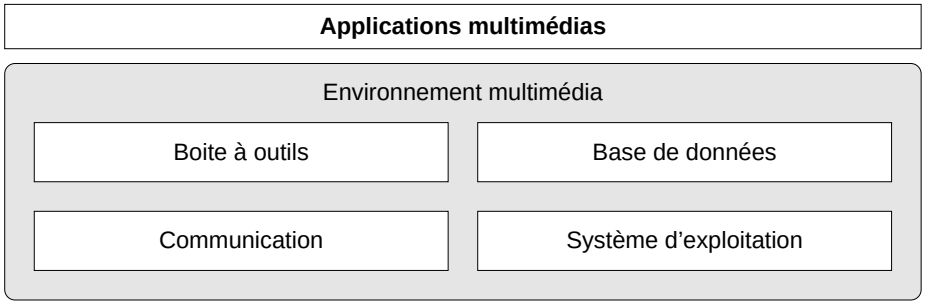


FIG. 2.6 – *Modules principaux d'un environnement multimédia*

2.3.1 Module de communication

Dans une application, différents objets peuvent participer à la composition d'une présentation. Ces objets peuvent être physiquement situés sur des plates-formes dis-

tinctes. Les objets devront être transmis à la présentation par des flux multimédias. Il faut alors envisager des couches qui permettent de synchroniser plusieurs flux.

La communication des données entre plates-formes se fait à travers des couches normalisées utilisant chacune un protocole particulier. La transmission doit pouvoir répondre à deux caractéristiques : le volume des données à transmettre et le besoin temps-réel lié au caractère temporel de ces données.

Des couches de communication ont été développées pour répondre au mieux aux caractéristiques des données multimédia. Par exemple la couche de transport utilisant le protocole RTP (*Real-Time Protocol*) définie pour les téléconférences multipoints. Ce protocole se situe au niveau application et est construit dans un environnement IP au dessus de la couche de transport UDP (*User Datagram Protocol*). De manière générale, les protocoles de communication basés sur l'échange de paquets n'offrent aucune garantie quant à l'arrivée des paquets à destination et à leur ordre.

Un des points clés de la transmission de flux de données multimédias est de pouvoir contrôler sa qualité. En ce sens, il est nécessaire de pouvoir garantir le maintien de critères de qualité de service (QoS) fixés initialement au début de la transmission. En effet, la garantie d'un niveau de QoS permet de choisir le meilleur codage des données en termes de débit, de délai et de fiabilité et ainsi de trouver un compromis 'qualité média/pertes' acceptable. Des protocoles tels que RSVP (*Ressource reServation Protocol*) et IPV6 ont été développés pour assurer des QoS en réservant des ressources et en spécifiant des traitements particuliers selon le type d'information transmise.

Le système de stockage de la vidéo ne consiste pas en un seul dispositif, mais est plutôt formé d'une hiérarchie de mémoires. Les vidéos (ou segments de vidéos) se situent soit dans la mémoire vive (mémoire RAM) avec des caractéristiques de vitesse élévée, coût élévé et capacité faible, soit dans la mémoire secondaire (disques magnétiques) avec des caractéristiques de vitesse, coût et capacité moyens, soit dans la mémoire tertiaire avec des caractéristiques de vitesse élévée, coût élévé et capacité faible. La localisation d'une vidéo dans la hiérarchie dépend de la phase de traitement où elle se trouve, ainsi que du niveau d'utilisation de chaque vidéo. Bien évidemment, des politiques déterminant le moment adéquat où la vidéo doit être relocalisée dans la hiérarchie doivent être stipulées de façon à diminuer le temps moyen d'accès aux vidéos dans la base.

La plupart des applications sont capables de composer de nombreux objets multimédias au sein d'une même scène. Les objets composant une scène doivent pouvoir être synchronisés à la fois indépendamment de leur propre base de temps mais aussi

entre eux. Les synchronisations intra ou inter objets sont d'autant plus difficiles à mettre en oeuvre que la plupart des objets composant la scène sont associés à un flux de données ayant ses propres caractéristiques temporelles. Pour faciliter la mise oeuvre de ces synchronisations, les applications s'appuyent sur les services de communication d'un couche de haut niveau au-dessus des couches de protocoles réseaux. Cette couche correspond en général à un *multiplex* synchrone qui offre la possibilité de multiplexer plusieurs flux de données en transmettant une base de temps commune et cohérente. Dans un contexte multimédia, nous constatons que l'utilité d'un telle couche est primordiale et qu'il est difficile de s'en affranchir.

Les objets d'une scène peuvent avoir leur contenu sur des plates-formes distinctes. Les différents contenus seront alors transmis via des flux de médias. Comme il n'est pas possible de multiplexer les flux provenant de plates-formes différentes, il faut envisager des couches permettant de synchroniser plusieurs flux de données multimédias. Lorsque les mêmes flux de données sont envoyés à des plates-formes différentes, il est possible d'utiliser des communications multipoints (*multicast*).

2.3.2 Système d'exploitation multimédia

Le système d'exploitation constitue la couche sur laquelle se déploient les applications multimédias. Il contrôle l'exécution des différents processus, assure leur protection mutuelle, administre la mémoire, traite les flux de données arrivant et sortant du système et gère aussi tous les événements (clavier, souris, réseau).

Les systèmes d'exploitation traditionnels permettent difficilement de gérer les médias continus. Par exemple décoder un flux son ou vidéo à partir d'un fichier peut provoquer des sautes d'images car le tampon n'est pas rempli suffisamment vite de données. Les applications multimédias ont des besoins importants en ressource processeur : il faut décoder des flux vidéo ou son, afficher les images ou traiter les événements utilisateur. De plus, la plupart de tâches à effectuer ont des contraintes temps-réel qu'il faut pouvoir respecter.

La convergence de nombreux travaux [Tok94] montre qu'il faut pouvoir intégrer principalement les deux points suivants : une gestion fine des processus temps-réel et des abstractions pour gérer les médias continus.

Des tendances actuelles cherchent à incorporer des fonctionnalités de gestion temps-réel aux couches classiques de temps partagé. Cette approche demande une conception différente du système en particulier du planificateur dont le rôle consiste

à sélectionner les processus à exécuter et les allouer au(x) processeur(s). Il faut également pouvoir suspendre un processus en cours pour démarrer un processus avec un délai de latence le plus court possible.

Enfin, plusieurs approches définissent des services spéciaux prenant en charge les médias continus (son, vidéo). Ces services intégrés dans le système d'exploitation gèrent l'allocation dynamique des tampons, la consommation des données au débit approprié ou la synchronisation des plusieurs flux. Par exemple la notion de *corde* introduit dans [TS88] est une abstraction de médias continus qui combinent les notions d'accès aux fichiers, de flux de données et de synchronisation.

2.3.3 Boîte à outils multimédias

Les boîtes à outils multimédias fournissent une panoplie de modules spécifiquement conçus pour les applications multimédias. Ces modules se présentent sous la forme de librairies directement incorporable dans l'application par l'intermédiaire des interfaces applicatives (API). Certains modules correspondent à des routines de bas niveau (proches de drivers) ; d'autres à des routines haut niveau. Ils implantent de nombreux traitements spécifiques à la chaîne multimédia classique allant du décodage au rendu.

Les boîtes à outils offrent une série de classes pour présenter et composer de façon interactive des entités multimédias (*http://www.microsoft.com*). Certaines sont dédiées à la présentation des données comme *DirectX* et *OpenGL*. *QuickTime* constitue une véritable extension du système d'exploitation (initialement du Macintosh et étendu au PC) permettant la création, la compression, l'édition et le présentation de données temporelles (i.e., vidéo). *QuickTime* correspond également à un format de représentation de données.

L'environnement *NEXT* (*http://www.nextstep.com*) est uniquement composée d'objets. Il propose une palette de librairies encapsulant des services, des interface avec les dispositifs, des classes pour gérer des images, des sons, des textes, des graphiques 2D et 3D, ainsi que des classes pour la compression et la présentation de la vidéo.

Director [DB98] est un logiciel de création d'applications multimédias. Le format *Director* permet d'encapsuler tous les types classiques de médias. Il permet de créer des séquences interactives et animées. La composition temporelle des différents médias se base sur un modèle qui combine les aspects temporels. Par exemple un canal temporel fournit des mécanismes pour contrôler la séquence des médias à travers de

contraintes (délais), la définition d'horloges, canaux d'animation qui permettent le positionnement spatial des médias.

2.3.4 Discussion

Le caractère multiforme des médias fait que chacune pose un problème de représentation et de traitement spécifique. La difficulté de spécifier des outils qui prennent en compte ces spécificités en respectant leurs besoins en termes de volume de stockage et d'exécution est évidente. Les systèmes doivent aussi permettre la combinaison des médias dans des documents, présentations et applications. Ceci accroît la difficulté de spécifier des environnements qui soient utilisables pour un large éventail d'applications (encyclopédies, publicité, etc.).

Les environnements multimédias fournissent des boîtes à outils qui permettent la construction d'applications utilisant et intégrant des données multimédias avec des données classiques. La construction d'applications doit être programmée à la main selon le type d'application. Néanmoins les aspects bas niveau, tels que les caractéristiques du système d'exploitation et de communication, semblent résolus par la plate-forme sur laquelle l'application se déploie. Ces caractéristiques doivent néanmoins être prises en compte pour implanter des politiques et des stratégies adaptées au niveau applicatif.

Enfin, chaque type de média est potentiellement représenté dans des formats différents, il possède des caractéristiques et des opérations spécifiques associées. Ces données sont stockés dans des sources spécialisées qui font partie intégrante d'un environnement multimédia. Des recherches sur les aspects structurels, temporels ou sur le contenu des données multimédias [SLR94, FSA+95, OS95, Jag96] sont faites. Il faut définir des langages de requêtes appropriés. La recherche de certaines données multimédias requiert l'utilisation de techniques d'indexation évoluées. Ces aspects sont du ressort de la technologie SGBD et ils sont traités dans la section 2.4.

2.4 SGBD multimédia

Un Système de Gestion de Bases de Données Multimédia (SGBDM) est un SGBD capable de gérer des types des données médias. La prise en compte des aspects multimédias dans un tel système a un impact important dans sa conception, ses caractéristiques et ses fonctions. La possibilité d'indexation et de recherche d'information

relative ainsi que la présentation visuelle des données sont des fonctions ajoutées à la valeur d'un SGBD.

2.4.1 Fonctionnalités

Un SGBDM est un système capable de :

- Intégrer à son modèle de données les aspects caractérisant les données multimédias, par exemple leurs attributs temporels (durée) et spatiaux (largeur, longueur). Des aspects concernant la composition temporelle et spatiale des médias doivent être également représentés dans le modèle de données.

- Modéliser la sémantique des données multimédias pour permettre des recherches sur le contenu (le modèle de données doit permettre de manipuler la structure des données en accord avec leur type de média). Le système doit fournir aussi des techniques d'indexation évoluées pour interroger les données.

- Permettre l'interrogation uniforme des données multiformes (médias, données classiques, textes) stockées dans des formats différents. Par exemple interroger des données représentées sous forme de relations ainsi que des données stockées dans des fichiers plats (par ex. une image au format GIF).

  La difficulté du traitement de requêtes accédant à des types des données hétérogènes vient du fait qu'il est difficile d'assurer la persistance des médias contenus dans un résultat de requête qui dépend fortement du format de stockage et de représentation du type de média.

- Stocker des données multimédias volumineuses requiert une grande capacité de stockage. Par exemple une vidéo durant une heure et demi nécessite un giga byte pour être stockée, même en étant fortement compressée. Les gestionnaires de stockage actuels ne sont pas adaptés aux besoins des données multimédias. Il est alors nécessaire que le SGBDM fournisse des mécanismes adaptés tels que des gestionnaires de stockage hiérarchique, des serveurs des médias continus [AH91, RV93, ORS96].

- Présenter de manière audiovisuelle les résultats de requêtes. L'utilisateur doit pouvoir spécifier la structure et la forme sous laquelle il faut présenté les résultats des requêtes qu'il pose.

La récupération et la présentation des données multimédia doit être faite en respectant leurs caractéristiques (i.e., débit de présentation). Les médias (par ex. vidéo) ont besoin d'espace mémoire important pour être présentés. Ils sont alors récupérés en morceaux à partir des supports de stockage (disque). Par conséquent, leur récupération implique la présence de mécanismes de planification permettant de préciser quand et comment récupérer ces morceaux afin d'assurer une qualité de présentation correcte.

Enfin, l'affichage des résultats doit considérer les caractéristiques matérielles où le système s'exécute (i.e., bande passante, coût de communication, trafic du réseau). Ces caractéristiques doivent être prises en compte par le SGBDM afin d'assurer une qualité de service dans la récupération et la présentation de médias.

2.4.2 Architecture d'un SGBDM

La construction d'un SGBDM doit prendre en compte les aspects concernant l'organisation du contenu des données médias ainsi que la façon dont ces données sont stockées sur disque. La figure 2.7 présente l'architecture générale d'un SGBDM.

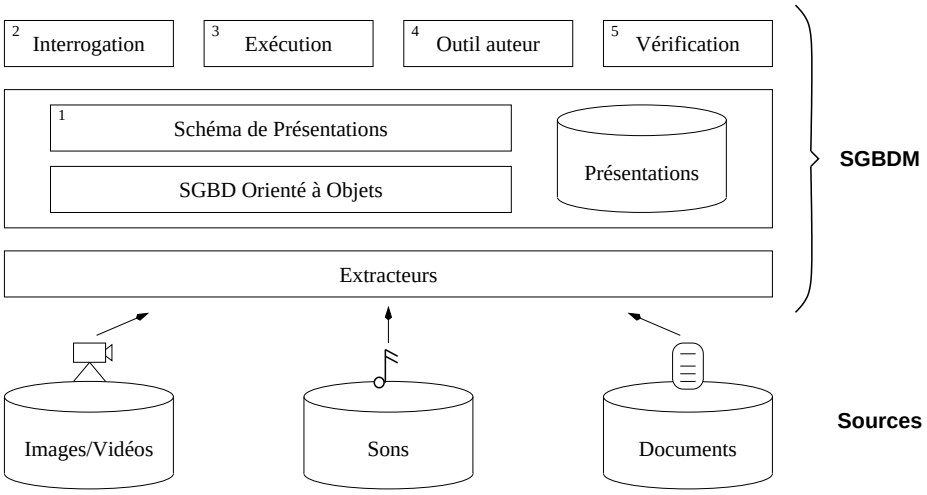


FIG. 2.7 – *Architecture générale d'un SGBDM*

1. Le gestionnaire des présentations multimédias implante un modèle permettant la représentation des médias ainsi que des relations entre ces médias pour créer,

rechercher et exécuter des requêtes comportant des données classiques et multimédias.

2. Le module d'interrogation fournit un langage permettant d'exprimer les caractéristiques des médias et les relations entre eux. En général, ces aspects sont décrits par des caractéristiques et des relations spatiales et temporelles. Le langage offert par le module d'interrogation doit permettre l'expression de tels aspects, ainsi que la spécification de la façon dont les résultats doivent être affichés à l'écran.

3. Le module d'exécution assure que chaque élément d'une présentation multimédia apparaît en temps requis et pendant une certaine période (synchronisation temporelle). Il assure également le regroupement des médias sur l'écran (placement spatiale). Il implante un modèle de synchronisation qui spécifie les politiques de construction d'un plan d'exécution des présentations ainsi que des stratégies de compensation lorsque les contraintes de récupération associées aux présentations et aux médias ne peuvent pas être respectées.

4. L'outil auteur fournit des interfaces pour la spécification de présentations multimédias. Il peut supporter des langages graphiques permettant de spécifier l'organisation spatio-temporel des médias dans une présentation. Cet outil doit coopérer avec un module de vérification qui assure la spécification cohérente de la configuration spatio-temporelle d'une présentation.

5. L'outil de vérification implante des algorithmes qui, étant donnée une spécification spatio-temporelle exprimé sous forme des relations et des médias, vérifie la cohérence temporelle et spatiale de la définition. Le principe est de calculer toutes les relations entre les médias afin de vérifier si toutes les configurations sont correctes par rapport à des règles de cohérence prédéfinies.

2.4.3 Principes de construction

Un SGBDM peut être construit selon trois principes : autonomie, uniformité, organisation hybride.

Autonomie Chaque type de média doit être organisé avec une approche adaptée à ses caractéristiques. Les techniques utilisées pour décrire (indexer) les données

changent selon qu'il s'agisse de textes, d'images, de sons ou de vidéos. Le SGBDM doit être construit de façon à fournir et à intégrer des mécanismes de gestion adaptées à chaque type de média.

L'architecture basée sur ce principe requiert la conception d'algorithmes et de structures de données pour chaque type de média. Des techniques pour calculer des jointures sur des structures de données différentes doivent être développées ou disponibles. Ceci peut être pénalisant pour la construction du SGBDM, mais améliorer les preformances de l'évaluation de requêtes.

Uniformité Un seul modèle de représentation doit être utilisé pour décrire tous les médias et pour fournir les algorithmes nécessaires permettant de les traiter. Le principe d'uniformité requiert d'avoir une structure de données commune à tous les types de médias. Définir une telle structure implique de trouver leurs aspects communs. Ce principe s'est matérialisé dans des techniques d'annotations décrites par des métadonnées. Les aspects décrivant les médias sont décrits en utilisant un métalangage. Ensuite, les métadonnées sont indexées et stockées pour supporter la manipulation des données.

L'avantage de cette technique est qu'elle est simple à implanter, cependant les annotations doivent êtres générées à la main ou automatiquement. De plus les métadonnées risquent de ne pas représenter tous les aspects décrivant un média. Lorsque les annotations sont réalisées automatiquement, des programmes de reconnaissance du contenu doivent être implantés. Toutefois ceci entraîne souvent des erreurs.

Organisation hybride Elle combine les principes d'autonomie et d'uniformité. Certains types de médias sont représentés de manière autonome et d'autres partagent leurs représentations. Au niveau du SGBDM, certains types de médias sont traités de manière particulière et d'autres sont traités suivant les mêmês techniques.

2.4.4 Systèmes existants

Aujourd'hui les SGBD commerciaux comme relationnels Sybase, UniSQL, Oracle et à objets O_2, ObjectStore supportent les BLOB (*Binary Large Object*) pour représenter les données multimédias. Un BLOB est non typé, long et avec un champ de longueur variable utilisé pour stocker des données multimédias dans la base. Le SGBD stocke la donnée multimédia comme une donnée non typé et délivre simplement des

blocs de données à l'application qui la demande. Ces systèmes fournissent également des mécanismes simples pour l'affichage des tels données.

Des travaux de recherche [Vaz96, Moc97, Li98, Loz00] se sont intéressés à l'extension des fonctionnalités des SGBD pour offrir à la fois des schémas pour la représentation des métadonnées décrivant des médias ainsi que des extensions des langages de requêtes. Par exemple les travaux de [Moc97, Loz00] utilisent les caractéristiques temporelles des médias pour les décrire et pour afficher des résultats des requêtes.

Les extensions des langages (SQL, OQL) permettent l'expression d'attributs spatiaux et temporels décrivant les données, mais aussi les formats sous lesquels les résultats des requêtes doivent être présentés. Dans [Ege94], on identifie des besoins spécifiques pour la définition d'un langage pour la spécification de requêtes spatiales géographiques. Un langage (*Spatial SQL*) qui permet de faire la récupération et la présentation des données géographiques est présenté. Un langage pour la présentation graphique (GPL) est introduit. GPL fournit des opérateurs pour la manipulation de la présentation graphique des résultats de requêtes.

Des travaux [MS96, TVS96, VTS98, ASS99] ont aussi proposé des mécanismes d'indexation des médias et/ou des métadonnées basées sur des structures multidimensionnelles.

Les auteurs de [OSEV95, LOS96b, LOSO97, Li98] ont construit le SGBD TIGUKAT, pour la gestion de documents multimédias. Ils étendent le modèle de données du SGBD pour représenter des documents par leur structure en utilisant des relations spatiales et temporelles. La structure logique des documents est exprimée en SGML et les relations spatio-temporelles en Hytime. Le modèle inclut la représentation du mouvement [LOS96a] utilisée pour la modélisation de la vidéo. Le langage MOQL [LOSO97], une extension du langages de requêtes OQL, est proposé comme langage d'interrogation de documents.

2.4.5 Discussion

La plupart de SGBDM restent dans un cadre centralisé et monolithique. Souvent, les données ne sont pas directement stockées dans la base, seule l'adresse (par ex. l'adresse URL) du fichier les contenant est gérée par le système. L'hétérogénéité et la distribution des sources de données continuent à être des problèmes ouverts dans les SGBDM mais aussi dans d'autres systèmes multimédias. Les solutions proposées restent propriétaires et très peu adaptables à des besoins applicatifs différents.

Les applications multimédias ont des besoins différents selon leur nature mais aussi selon les environnements (matériels et logiciels) dans lesquels elles s'exécutent. Lorsqu'elles utilisent des données classiques mais aussi des données multimédias leurs besoins incluent la communication, l'efficacité de récupération, etc.

L'évolution actuelle de ces applications met en relief de façon accrue la nécessité de développer des infrastructures configurables pour garantir la gestion d'un certain nombre de fonctionnalités communes à plusieurs types d'applications (i.e., communication, persistance, réplication, interrogation) avec leurs propres caractéristiques (par ex. plate-forme d'exécution) et besoins.

2.5 Les standards

Le processus de standardisation joue un rôle essentiel à la progression du multimédia. Les standards permettent de stabiliser les formats, les cadres logiciels et les interfaces applicatives (API). Un standard de présentation multimédia doit incorporer les différents médias ainsi qu'une représentation permettant de les composer.

2.5.1 Langages de programmation de présentations

La plupart des langages pour la programmation d'applications multimédias sont basés sur le langage SGML (*Standard Generalized Markup Language*) [ISO86]. Il s'agit de langages incluant des opérateurs hyperliens et de synchronisation. Des documents interactifs peuvent aussi être définis avec ces langages. Les aspects temporels sont alors largement pris en compte alors que les aspects spatiaux sont traités de manière restreinte.

HyTime (*Hypermedia/Time-based structuring language*) [ISO97] est un langage de structuration de documents (hyperdocuments) qui supporte la spécification d'hyperliens, la planification temporelle et la synchronisation. C'est un langage pour décrire la façon dont des objets sont interconnectés et la façon dont les utilisateurs y accèdent.

PREMO (*Presentation Environment for Multimedia Objects*) [ISO94] est un standard ISO qui fournit un environnement de développement d'applications multimédias. Il s'intéresse à la création de présentations interactives composées de médias. Il permet la mise en place de services multimédias sur le réseau.

XML (*Extensible Markup Language*) [W3C98a] est un métalangage de spécification des langages réguliers de marquage. Il permet de décrire les informations ca-

ractérisant une classe de documents. Les documents XML peuvent être exécutés sur un navigateur. Les présentations SMIL (*Synchronized Multimedia Integration Language*) [W3C98b] sont des documents XML modélisés comme des présentations multimédias. SMIL permet d'intégrer et de synchroniser un ensemble d'objets médias dans des présentations. Par exemple la synchronisation d'une vidéo avec un texte, un son avec une image. De manière similaire aux documents XML, les documents SMIL sont aussi exécutés sur un navigateur.

2.5.2 Environnements standards

L'évolution des standards de la famille MPEG [NL99a] (MPEG-1, MPEG-2, MPEG-4 et MPEG-7) tend clairement vers l'ajout de sémantique et de qualité. Le standard MPEG-1 (ISO/IEC 11172) très utilisé dans la production de CD propose des techniques de compression et de décompression de données avec perte basées sur la suppression des redondances. Ce standard est basé sur un taux de transfert de 1,86MB/sec pour une taille d'images de 720×576 pixels.

Le standard MPEG-2 (ISO/IEC 13818) illustre une évolution en terme de qualité en adressant les applications de télévision numérique ou la production de films en format DVD. Il propose quatre niveaux de représentation qui vont d'un taux de transfert de 4MB/sec pour le niveau le plus bas à 60MB/sec pour le plus élevé. La taille des images quant à elle varie de 352×288 pixels à 1920×1152 pixels. En dehors de ces caractéristiques, MPEG-2 propose des améliorations telles que la définition de trois niveaux de chrominance contre un seul pour MPEG-1.

Le standard MPEG-4 (ISO/IEC 14496) représente une évolution considérable par rapport à ces deux prédécesseurs en ne s'intéressant plus à la compression des données mais à la création ascendante d'une vidéo en partant des objets qui la composent. La vidéo est considérée comme un ensemble d'objets qui sont composés dans l'espace et dans le temps en utilisant un langage de réalité virtuelle tel que VRML pour décrire les animations. L'avantage essentiel est que la quantité de données pour reproduire une vidéo est moins élevée que celle du flux vidéo dans son intégralité. Cette caractéristique est très appréciée pour les applications de type Web. MPEG-4 offre également un ensemble de paramètres liés à la qualité de service.

Enfin, le standard en cours de définition MPEG-7 [NL99b] est basé sur le standard MPEG-4 et s'intéresse à la description du contenu des différentes scènes. Cette description concerne à la fois les éléments visuels (couleur, texture, position) et les

informations de niveau d'abstraction supérieur telles que le nom d'une personne présente sur la vidéo et son activité. Ce bref historique montre que la vidéo n'est plus considérée comme un type de données figé sans structure ni sémantique mais au contraire comme un type de données complexe à intégrer avec les autres données plus conventionnelles.

2.5.3 Discussion

La présence des standards (formats de compression, langages de spécification, modèles de synchronisation, plates-formes d'exécution) permet de stabiliser et de caractériser les besoins que doit combler la technologie multimédia. Par contre, l'émergence de standards semble loin de se stabiliser, ce qui a pour conséquence que les applications et les outils de construction doivent prendre en compte des extensions et des nouveaux critères des standards.

Les standards proposent des langages de spécification qui permettent la définition de documents : structure, médias contenus, disposition spatiale et temporelle, caractéristiques du "décor". Des environnements comme la famille des MPEG tentent d'offrir des outils pour la compression, la représentation, l'exécution, l'inter-opération des sources de données, etc. qui supportent la construction d'applications.

L'approche MPEG rejoint l'idée des librairies de classes (services spécialisés) pour la construction d'applications. Cela signifie que des interfaces applicatives (API) standard sont mises en place pour la gestion de données multimédias. Avoir des standards, ne signifie pas que des applications et des outils *ad hoc* ne doivent pas être intégrés dans des systèmes plus complexes. L'inter-opérabilité entre le monde standard et le non standard reste à être explorée.

2.6 Conclusion

Nous avons présenté les types de données multimédias et la problématique liée à leur gestion. Nous avons souligné que la spécificité de chaque type de donnée impose une étude particulière et des techniques de gestion spécifiques.

Ces caractéristiques donnent lieu à la construction de systèmes avec des mécanismes adaptés pour chaque média, ce qui implique des architectures lourdes à gérer et à maintenir. Les systèmes varient entre des librairies de classes utilisables pour la programmation d'applications manipulant des médias, aux systèmes de support tels

que les SGBD au dessus desquels se construisent des applications.

La plupart des propositions (non issues du domaine bases de données) ne s'intéressent pas à l'indépendance physique et logique dans la construction des bases de données. Les algorithmes d'interrogation sont donc fortement associés à la façon dont la base de médias est construite. Par ailleurs, la plupart des standards fournissent des langages de programmation de présentations multimédias. Aucun mécanisme de support à la construction des présentations n'est mis à disposition, en dehors des compilateurs (interpréteurs) et des exécuteurs *ad hoc* de ces langages.

De notre point de vue, les documents multimédias spécifiés en utilisant des langages standards sont aussi des données multimédias qui devraient pouvoir être exploitées : interrogées, combinées, réutilisées complètement ou partiellement. Nous sommes convaincus que des composants permettant l'intégration transparente des données, des applications et des outils de différentes natures doivent encore être proposés.

Concernant l'exploitation des médias et des documents, produire des outils d'aide à la recherche sur les données multimédias est un problème crucial pour les systèmes. Cet aspect concerne directement le domaine de bases des données et a été largement abordé par différentes approches. Les SGBD ne permettent pas d'interroger avec exactitude les médias.

Les produits commerciaux témoignent de la nécessité d'avoir des modules spécialisés fournissant des fonctionnalités précises associées à des médias. Il nous semble que le premier pas est de séparer la gestion des médias et des présentations avec un mécanisme spécialisé fournissant un composant adapté pour les gérer. Un tel système (service) doit fournir les moyens permettant de représenter les médias et leurs caractéristiques, des langages pour spécifier la composition des médias et les mécanismes adaptés pour les gérer et pour les exécuter sur des plates-formes cibles. Dans le chapitre suivant, nous abordons le problème de la représentation de médias et des présentations à travers un modèle spatio-temporel.

Chapitre 3

Modèle spatio-temporel

Ce chapitre introduit notre modèle qui permet de caractériser les propriétés spatiales et/ou temporelles des objets et la façon dont ils sont ordonnés lorsqu'ils sont présentés. Il fournit des concepts spécifiques pour caractériser la position et la taille des objets ainsi que les relations spatio-temporelles entre eux.

La section 3.1 introduit les concepts de base utilisés dans le modèle. La section 3.2 décrit les types de présentations supportés par le modèle. Ensuite, la section 3.3 caractérise les types de relations utilisées pour décrire la configuration spatiale et/ou temporelle d'un espace. La section 3.4 présente la composition de présentations et montre comment le modèle permet de décrire des relations spatiales et temporelles entre des médias, ainsi que le contenu des médias. La section 3.5 compare notre modèle avec des approches existantes. Enfin la section 3.6 conclut ce chapitre.

3.1 Concepts de base

3.1.1 Espace de présentation

Nous considérons un espace comme une composition de n dimensions où chacune est considérée comme un continuum. De manière générale, une dimension est un intervalle de réels dont les bornes inférieure et supérieure sont respectivement $-\infty$ et ∞. Dans tout espace, nous distinguons un point particulier $(0_1, \ldots, 0_n)$ appelé origine. Selon le nombre de dimensions, nous distinguons les types d'espace suivants :

- Un espace temporel unidimensionnel est représenté par une ligne de temps continue t avec zéro comme borne inférieure. Quelque soit la représentation d'un

instant dans le temps, nous pouvons le transformer en un élément du domaine des réels.

- Un espace 2D est une composition de deux dimensions (x, y) bornées par $[-\infty, \infty]$. Quelque soit la représentation d'un point 2D, nous pouvons le désigner par une paire (a, b) où a et b sont des éléments du domaine des réels.
- Un espace 3D est une composition de trois dimensions :
  1. (x, y, z) où x, y et z sont bornées par $[-\infty, \infty]$,
  2. (x, y, t) où x et y sont bornées par $[-\infty, \infty]$ et t par $[0, \infty]$.

  De manière analogue aux types d'espace précédents, quelque soit la représentation d'un point 3D, nous pouvons le désigner par un triplet (a, b, c) où a, b et c sont des éléments du domaine des réels.
- Un espace 4D est une composition de quatre dimensions (x, y, z, t) où x, y et z sont bornées par $[-\infty, \infty]$ et t par $[0, \infty]$. Quelque soit la représentation d'un point 4D, nous pouvons le désigner par un quadruplet (a, b, c, d) où a, b, c et d sont des éléments du domaine des réels.

Nous définissons une notion de granularité comme une partition de l'ensemble des points d'une dimension. Chaque partition ainsi obtenue est un ensemble convexe appelé grain. Par exemple le partitionnement en semaines, mois et années de la dimension temporelle, et celui en millimètres, centimètres et mètres des dimensions spatiales correspondent à des granularités. Du fait qu'une granularité partitionne les points d'une dimension en des ensembles convexes, l'ordre défini sur les points induit un ordre linéaire et total sur les grains composant une granularité [Dum00].

3.1.2 Ombre spatio-temporelle

La figure 3.1 illustre le concept d'ombre [Adi96]. Elle est constituée de deux intervalles consécutifs δ et d dénommées décalage et taille. Ces intervalles sont caractérisés par leurs points de début et de fin. Le concept d'ombre est appliqué pour caractériser la présentation d'un objet sur chacune des dimensions d'un espace. Selon la dimension sur laquelle elle est utilisée, l'ombre peut avoir des interprétations différentes.

Une ombre temporelle (OT) décrit l'intervalle de temps δ, défini entre le moment où un objet (i.e., texte, image, son, vidéo, etc.) est prêt à être présenté mais il n'est

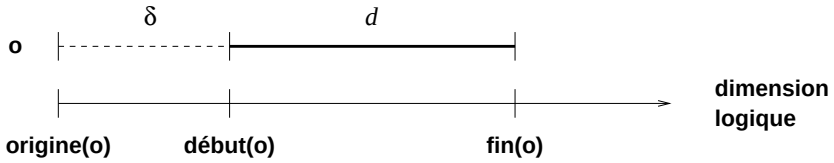


FIG. 3.1 – *Ombre*

pas encore visible (perceptible) et l'intervalle de temps d pendant lequel il est visible (la durée de sa présentation).

Dans l'espace 2D, un objet est associé à une ombre spatiale (OS) composée de deux ombres (une pour chaque dimension). Elle décrit les attributs d'un objet sa position (δ_x, δ_y) dans un espace et sa taille (d_x, d_y). Notez que le concept d'ombre spatiale est analogue au concept de MBR (*Minimum Bounding Rectangle*) utilisé pour avoir une représentation des objets [PTSE95, PT97].

Enfin dans un espace à trois dimensions, une Ombre Spatio-Temporelle (OST) décrit la position (δ_x, δ_y, δ_t) et la taille (d_x, d_y, d_t – largeur, hauteur et durée) d'un objet. Il s'agit d'un sextuplet (δ_x, δ_y, δ_t, d_x, d_y, d_t) qui combine une ombre spatiale (OS) et une ombre temporelle (OT), où (δ_x, δ_y, d_x, d_y) définit OS et (δ_t, d_t), OT.

La figure 3.2 montre un exemple d'une OST. En première approximation, OS est donnée en pixels et OT en secondes. L'origine de OS est supposée être le coin supérieur gauche d'un écran.

OST	
OS	OT
$\delta_x = 60,\ \delta_y = 35$ $d_x = 30,\ d_y = 25$	$\delta_t = 40$ $d_t = 28$

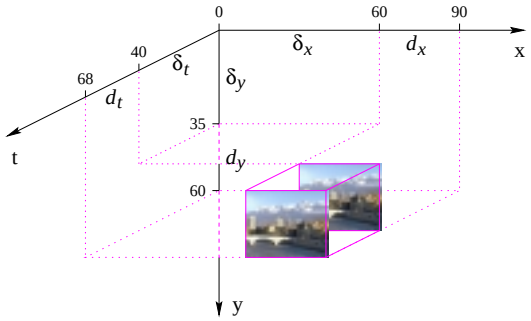


FIG. 3.2 – *Ombre spatio-temporelle*

Dans la figure 3.2, une image de Grenoble est présentée à la position (60, 35) à partir de la seconde 40. L'image a une longueur de 30 pixels et une hauteur de 25 pixels, et elle reste exposée pendant 28 secondes. Donc le OST de la présentation de Grenoble est (60, 35, 30, 25, 40, 28) où OS = (60, 35, 30, 25) et OT = (40, 28).

L'idée principale du concept d'ombre est que différentes présentations peuvent être associées à un objet. Cela permet de considérer les aspects spatiaux et temporels simultanément ou séparément selon les besoins des applications.

3.1.3 Types d'objets

Type Objet Il représente tous les types du modèle spatio-temporel. Un type est défini par les règles suivantes :

- un type atomique (integer, real, string, boolean, Text, Image, Audio, Video, ...) est un type ;
- si T est un type alors {T} est un type ensemble ;
- si $A_1 \ldots A_n$ sont des noms d'attributs distincts et $T_1 \ldots T_n$ leurs types respectifs, alors $(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n)$ ou $(A_1, A_2, \ldots, A_n)$ est un type n-uplet ;
- une ombre est un type ; si δ et d dénotent des réels alors $\mathcal{S}$ dénote un type n-uplet S de la forme (δ, d) ;
- une présentation est un type ; si P dénote la présentation $\mathcal{P}$, alors P dénote un type n-uplet de la forme $(\mathcal{S} : S, \mathcal{O} : \text{Objet})$ où S est le type d'une ombre associée à un objet $\mathcal{O}$ de type Objet.

Les types atomiques sont munis d'opérateurs arithmétiques, temporels, logiques, ensemblistes, de comparaison, des opérateurs sur les chaînes de caractères ainsi que des opérateurs relatifs aux différents types des médias (texte, image, son, vidéo).

Le type présentation P regroupe un ensemble de types abstraits de présentations : Temporelle, Spatiale 2D, Spatiale 3D, Spatiale 2D et Temporelle, Spatiale 3D et Temporelle. Il est possible d'établir des relations d'ordre entre ces types :

- Temporelle $<$ (Spatiale 2D et Temporelle) $<$ (Spatiale 3D et Temporelle),
- Spatiale 2D $<$ Spatiale 3D,
- Spatiale 2D $<$ (Spatiale 2D et Temporelle) $<$ (Spatiale 3D et Temporelle),
- Spatiale 3D $<$ (Spatiale 3D et Temporelle).

Une présentation de type P décrit la façon dont un ensemble d'objets doivent être combinés dans un espace de présentation (i.e., une fenêtre, l'écran). Elle caractérise un objet par ses projections sur les dimensions d'un espace. Les projections sont modélisées comme des intervalles., Un objet dans un espace à n dimensions est donc représenté comme une combinaison de n intervalles (i.e., dans 1D les objets sont des intervalles (lignes), dans 2D des rectangles, dans nD des hyper-rectangles). Par exemple dans la dimension spatiale, un objet est représenté comme un rectangle dont les côtés correspondent à des projections sur les axes x et y.

Le type présentation peut disposer d'opérateurs associés qui permettent de décrire par exemple la composition spatio-temporelle ainsi que des opérations telles que la sélection et l'agrégation.

Règles de composition de présentations Les règles suivantes définissent la composition de présentations dans notre modèle. Notons que pour chacun de ces types, il existe un type d'espace associé (cf. tableau 3.1).

- Si $\mathcal{O}$ est de type Objet alors (OT, $\mathcal{O}$), (OS, $\mathcal{O}$), (OST, $\mathcal{O}$) sont respectivement des présentations de type Temporelle, Spatiale 2D et (Spatiale 2D et Temporelle).
- Si A et B sont deux présentations de type T_1 et T_2, et $\sigma\tau$ une relation alors ($\mathcal{S}$, A $\sigma\tau$ B) est une présentation composite de type :
  - T_2 si $\mathsf{T}_1 \leq \mathsf{T}_2$;
  - T_1 si $\mathsf{T}_1 > \mathsf{T}_2$;
  - $\mathsf{T}_1 + \mathsf{T}_2$ si $\mathsf{T}_1 \not\leq \mathsf{T}_2$ et $\mathsf{T}_1 \not> \mathsf{T}_2$.

    Par exemple la composition d'une présentation Spatiale 2D d'une image avec une présentation temporelle d'un son résulte en une présentation composite spatio-temporelle (Spatiale 2D et Temporelle) qui combine les deux présentations.

  $\mathcal{S}$ est une ombre du même type que la présentation résultante.

3.2 Présentations spatio-temporelles

Une présentation $\mathcal{P}$ spécifie (1) l'ensemble d'objets qu'elle contient (2) la taille et la position absolue de chaque objet par rapport à un référentiel ; et (3) les positions

Type atomique	Type composite	Type d'espace
Temporelle	Temporelle	1D
Spatiale 2D	Spatiale 2D	2D
Spatiale 3D	Spatiale 3D	3D
Spatiale 2D et Temporelle	Spatiale 2D et Temporelle	3D
Spatiale 3D et Temporelle	Spatiale 3D et Temporelle	4D

TAB. 3.1 – *Types des présentations*

relatives de chaque objet par rapport aux autres. Dans notre modèle, une présentation est en général contenue et rendue visible dans un espace à trois dimensions incluant le temps.

Les présentations peuvent être formées (récursivement) par d'autres présentations. Nous distinguons les présentations atomiques des présentations composites où plusieurs objets sont associés en utilisant des relations spatiales et temporelles (cf. section 3.3).

3.2.1 Présentation atomique

Une présentation atomique associe un objet (i.e., texte, image, son, vidéo, etc.) à une ombre spatio-temporelle. Une telle présentation est définie par un couple (OST, $\mathcal{O}$) où l'ombre OST exprime les caractéristiques spatiales et/ou temporelles (i.e., position, taille, délai et durée) d'un objet source $\mathcal{O}$ à présenter. En reprenant l'exemple de la figure 1.1, le titre peut être défini comme une présentation atomique de la manière suivante : ((210, 80, 580, 90, 0, 315), **title**). Cette expression spécifie que le titre doit être présenté à la coordonnée (210, 80) avec une taille de 580×90 à partir de la seconde 0 jusqu'à la seconde 315.

Une présentation est une manière de percevoir un objet donné. Ainsi un même objet (par exemple une image) peut être présenté de différentes manières en jouant sur les paramètres temporels et/ou spatiaux. Notons que OST spécifie aussi les caractéristiques spatiales et temporelles de l'espace où la présentation est matérialisée. La figure 3.3 (a) montre la présentation spatiale d'un objet dans un espace de même taille que l'objet. Dans la figure 3.3 (b), la position de l'objet source est modifiée. L'objet est présenté à la position (60, 35). Par conséquent, la taille de l'espace de présentation est modifiée aussi. Enfin, dans la figure 3.3 (c), la taille de l'objet lui même est modifiée. L'objet est présenté avec une réduction de 50%.

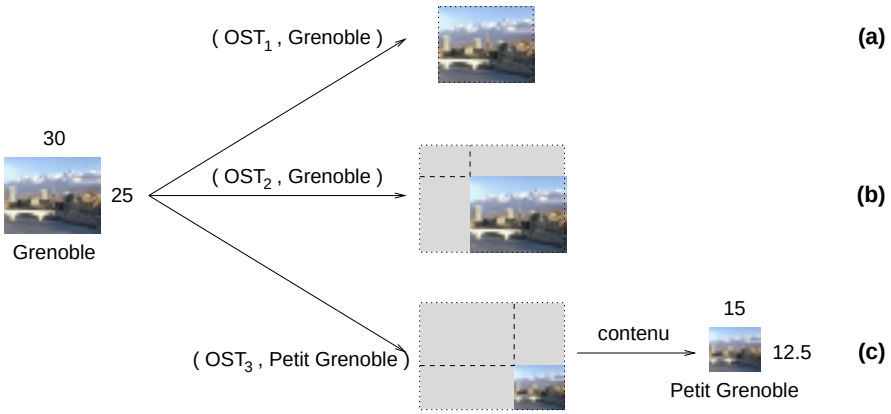


FIG. 3.3 – *Présentations atomiques*

3.2.2 Présentation composite

Une présentation composite est définie par un n-uplet de la forme (OST, A $\sigma\tau$ B) où deux présentations A et B (atomiques ou composites) sont combinées au moyen d'une relation spatio-temporelle $\sigma\tau$. Les décalages δ_x, δ_y, δ_t de OST sont définis par rapport à l'origine de l'espace de présentation ; d_x, d_y, d_t sont respectivement déterminées à partir des ombres spatio-temporelles de A et B.

Sur chaque dimension, $\sigma\tau$ spécifie quelle présentation (i.e., A ou B) est prise comme référentiel et quelle autre, comme cible. Dans cette composition, les présentations référence et cible représentent respectivement la position absolue de la présentation et la position relative entre A et B. Cette représentation est faite en considérant que les relations spatiales et temporelles sont exprimées en termes d'intervalles (cf. section 3.3).

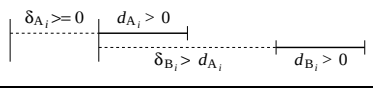
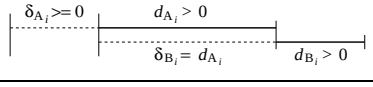
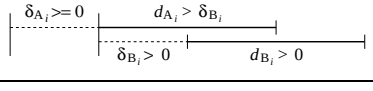
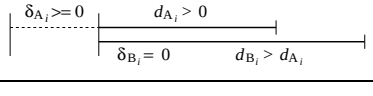
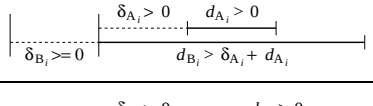
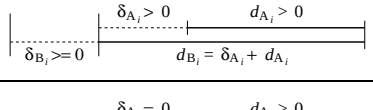
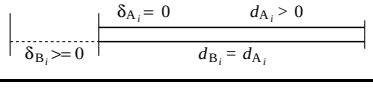
3.3 Relations spatio-temporelles

Dans une présentation composite, la position de présentation de chaque objet (i.e., présentation atomique) est caractérisée (1) par une position absolue par rapport l'origine et (2) par des positions relatives par rapport aux autres présentations d'objets. Les positions relatives peuvent être exprimées qualitativement en termes de relations spatiales et temporelles complétés, pour certaines, par des informations quantitatives (par exemple des délais et des décalages relatifs).

3.3.1 Relations d'intervalles

Puisque la présentation d'un objet est modélisée par des intervalles, nous utilisons les relations proposées par [All83] pour représenter la composition d'un objet dans l'espace et dans le temps. Cela est possible parce que les relations spatiales (topologiques et directionnelles, cf. section 3.3.2) peuvent être exprimées en utilisant ces relations.

Pour représenter une présentation composite, une relation spatio-temporelle combine des relations d'intervalles (une pour chaque dimension). Considérons un espace à trois dimensions, où $i \in \{$ `x`, `y`, `t` $\}$ est une dimension et A_i, B_i sont respectivement les intervalles de A et B sur la dimension i. La relation spatio-temporelle A $\sigma\tau$ B est définie par trois prédicats r_x(A, B), r_y(A, B), r_t(A, B), où $r \in \{$ `none`, `seq`, `par`, `before`, `meet`, `overlap`, `start`, `during`, `finish`, `equal`, `bi`, `mi`, `oi`, `si`, `di`, `fi` $\}$. Chaque prédicat r_i: $\mathcal{P} \times \mathcal{P}$ -> `boolean` est vérifié si les ombres sur i de A et B sont arrangées en accord avec les contraintes du prédicat r_i défini dans le tableau 3.2.

Générique		Spécifique	Inverse	Sémantique des ombres
`none`$_i$(A, B)	`seq`$_i$(A, B)	`before`$_i$(A, B)	`bi`$_i$(B, A)	
		`meet`$_i$(A, B)	`mi`$_i$(B, A)	
	`par`$_i$(A, B)	`overlap`$_i$(A, B)	`oi`$_i$(B, A)	
		`start`$_i$(A, B)	`si`$_i$(B, A)	
		`during`$_i$(A, B)	`di`$_i$(B, A)	
		`finish`$_i$(A, B)	`fi`$_i$(B, A)	
		`equal`$_i$(A, B)	`equal`$_i$(B, A)	

TAB. 3.2 – *Prédicats*

Nous introduisons ensuite les prédicats qui décrivent les relations d'intervalles en termes d'ombre et nous montrons de quelle façon ces prédicats sont utilisés pour

spécifier des présentations spatio-temporelles :

- `none`$_i$(A, B) dénote que les ombres de A et B sont arrangées indépendamment sur la dimension i. Par exemple la présentation du titre et de l'explication définie dans l'exemple de la figure 1.1 est défini comme suit (OST, (OST$_A$, **title**) $\sigma\tau$ (OST$_B$, **explanation**)), où $\sigma\tau$ est définie par `none`$_x$(A, B), `none`$_y$(A, B) et `equal`$_i$(A, B).

- `seq`$_i$(A, B) décrit un arrangement séquentiel. Cette relation généralise les relations `before` et `meet`.

- `par`$_i$(A, B) décrit un arrangement parallèle, où les intervalles A$_i$ et B$_i$ se superposent. Cette relation généralise les relations `overlap`, `start`, `during`, `finish` et `equal`.

3.3.2 Relations topologiques et directionnelles

Les tableaux 3.3 et 3.4[1] présentent les relations topologiques et directionnelles que nous adoptons. Elles sont définies en termes de relations d'intervalles en considérant un espace 2D. Les relations topologiques (i.e., `Disjoint` (`DJ`), `Touch` (`TC`), `Overlap` (`OV`), `Cover` (`CV`), `Covered by` (`CB`), `Inside` (`IN`), `Contain` (`CT`), `Equal` (`EQ`)) caractérisent le fait que deux régions spatiales se superposent, se touchent ou sont disjointes. Pour ces relations, nous avons utilisé la définition donnée par [EF91, Li98]. En particulier, nous avons défini toutes les relations et leurs inverses.

Les relations directionnelles (i.e., `North` (`NT`), `South` (`ST`), `West` (`WT`), `East` (`ET`), `Northwest` (`NW`), `Southeast` (`SE`), `Northeast` (`NE`), `Southwest` (`SW`)) concernent l'ordre des objets dans l'espace. Elles sont utilisées comme des critères de sélection en divisant l'espace, par exemple Grenoble est placé dans la région sud-est de la France. Les relations métriques évaluent les distances spatiales entre objets, par exemple Lyon est proche de Grenoble, Paris est loin de Grenoble, etc.

La raison principale pour laquelle nous avons choisi les relations topologique et directionnelles est que les relations spatiales entre intervalles peuvent être exprimées en les utilisant. Nous excluons les relations métriques parce que leur sémantique est dépendante de l'application et sujet à des interprétations diffuses. En plus, une par-

1. Nous utilisons une notation courte {} pour distribuer la disjonction sur les relations d'intervalles. Par exemple A$_x$ {`before`,`bi`} B$_x$ est équivalent à `before`$_x$(A, B) $\vee$ `bi`$_x$(A, B).

Relation	Inverse	Définition
DJ(A, B)	DJ(B, A)	A_x {before,bi} $B_x \vee A_y$ {before,bi} B_y
TC(A, B)	TC(B, A)	A_x {meet,mi} $B_x \wedge$ A_y {meet,mi,overlap,oi,start,si,during,di,finish,fi,equal} $B_y \vee$ A_x {meet,mi,overlap,oi,start,si,during,di,finish,fi,equal} $B_x \wedge$ A_y {meet,mi} B_y
OV(A, B)	OV(B, A)	A_x {overlap,oi} $B_x \wedge$ A_y {overlap,oi,start,si,during,di,finish,fi,equal} $B_y \vee$ A_x {equal} $B_x \wedge A_y$ {overlap,oi} $B_y \vee$ A_x {start,during,finish} $B_x \wedge A_y$ {overlap,oi,si,di,fi} $B_y \vee$ A_x {si,di,fi} $B_x \wedge A_y$ {overlap,oi,start,during,finish} B_y
CV(A, B)	CB(B, A)	A_x {di} $B_x \wedge A_y$ {si,fi,equal} $B_y \vee$ A_x {si,fi} $B_x \wedge A_y$ {si,di,fi,equal} $B_y \vee$ A_x {equal} $B_x \wedge A_y$ {si,di,fi} B_y
IN(A, B)	CT(B, A)	A_x {during} $B_x \wedge A_y$ {during} B_y
EQ(A, B)	EQ(B, A)	A_x {equal} $B_x \wedge A_y$ {equal} B_y

TAB. 3.3 – *Relations topologiques*

ticularité intéressante des relations topologiques et directionnelles est qu'en les combinant dans une description spatiale elles permettent d'exprimer des distances en termes qualitatifs. Par exemple un titre au dessus d'une carte peut être décrit avec North et Touch.

Relation	Inverse	Définition
NT(A, B)	ST(B, A)	A_x {during,di,equal} $B_x \wedge$ A_y {before,meet,overlap,start,fi} B_y
WT(A, B)	ET(B, A)	A_x {before,meet,overlap,start,fi} $B_x \wedge$ A_y {during,di,equal} B_y
NW(A, B)	SE(B, A)	A_x {before,meet,overlap,start,fi} $B_x \wedge$ A_y {before,meet,overlap,start,fi} B_y
NE(A, B)	SW(B, A)	A_x {bi,mi,oi,si,finish} $B_x \wedge$ A_y {before,meet,overlap,start,fi} B_y

TAB. 3.4 – *Relations directionnelles*

En nous basant sur [Her94, TPSS98], nous donnons une définition des relations directionnelles, présentées dans le tableau 3.4. Ceci nous permet de combiner ces relations avec les relations topologiques et nous permet d'exprimer n'importe quelle relation dans notre espace de relations d'intervalles (169 relations, cf. annexe B).

Pour plusieurs relations, leur sémantique n'est pas suffisante pour interpréter la connaissance quantitative. Par exemple la relation composite DJ(A, B) avec NT(A, B) caractérise l'ombre absolue (a) et relative (r) en y (cf. première ligne du tableau 3.5). Pour caractériser (c) l'ombre en x, nous utilisons d_{A_x} et d_{B_x} : si DJ(A, B) avec ST(B, A)

avec $d_{A_x} > d_{B_x}$, alors δ_{A_x} est une position absolue et δ_{B_x} est une position relative.

Relation directionnelle	DJ(A, B)	TC(A, B)	OV(A, B)	CV(B, A) CB(A, B)	OS_A		OS_B	
					δ_{A_x}	δ_{A_y}	δ_{B_x}	δ_{B_y}
NT(A, B) ST(B, A)					c	a	c	r
WT(A, B) ET(B, A)					a	c	r	c
NW(A, B) SE(B, A)					a	a	r	r
NE(A, B) SW(B, A)					r	a	a	r

TAB. 3.5 – *Ombre et relations spatiales*

Comme dans [Her94, KR98], nous faisons la distinction entre les relations topologiques qui peuvent ou non être naturellement combinées avec les relations directionnelles. Nous avons décrit le premier groupe dans le tableau 3.5 qui établit la correspondance entre les relations d'intervalles et l'information spatiale représentée par l'ombre OS. Chaque OS composante maintient une connaissance quantitative, soit absolue ou relative.

Le deuxième groupe est composé par `Inside`, `Contain`, `Equal`. Les relations directionnelles ont des propriétés particulières lorsqu'elles sont utilisées avec les relations topologiques du deuxième groupe. Les relations directionnelles ne fournissent pas d'informations supplémentaires sur la configuration spatiale (`Equal`) ou elles ne spécifient pas une direction en n'utilisant que des informations qualitatives (`Inside` et `Contain`). L'utilisation de l'information quantitative de OS_A et OS_B permet de déduire une relation directionnelle.

3.3.3 Positionnement des présentations

Pour faciliter la description de positionnement, nous avons défini un ensemble de relations d'alignement qui permettent d'exprimer par exemple le fait que deux objets sont centrés, alignés à gauche ou à droite. De manière similaire à toutes les relations de notre modèle, elles ont été définies en termes d'intervalles et d'ombres.

Pour définir ces relations, nous distinguons trois points sur les intervalles qui caractérisent les présentations de ces objets à savoir le début, le milieu et la fin. De cette manière, nous avons défini cinq configurations pour combiner les intervalles de

A et B sur une dimension i où A est pris comme référence :

- **beginning** : les débuts des deux intervalles coïncident. Selon la relation de taille entre A et B, il y a trois compositions possibles : $\texttt{si}_i$(A, B), $\texttt{start}_i$(A, B), ou $\texttt{equal}_i$(A, B) (cf. figure 3.4).

- **centered** : les milieux des deux intervalles coïncident, i.e., la composition de A et de B vérifie l'un des prédicats suivants $\texttt{di}_i$(A, B) avec $\delta_{B_i} = (d_{A_i}$ - $d_{B_i})$ / 2, $\texttt{during}_i$(A, B) avec $\delta_{A_i} = (d_{B_i}$ - $d_{A_i})$ / 2 ou $\texttt{equal}_i$(A, B) (cf. figure 3.5).

- **end** : les fins des deux intervalles coïncident, i.e., A et B vérifient $\texttt{fi}_i$(A, B) ou $\texttt{finish}_i$(A, B) ou $\texttt{equal}_i$(A, B) (cf. figure 3.6).

- **middle** : soit le milieu de d_{B_i} coïncide avec le début ou la fin de d_{A_i} (i.e., A et B vérifient $\texttt{oi}_i$(A, B) avec $\delta_{A_i} = d_{B_i}/2$ ou $\texttt{overlap}_i$(A, B) avec $\delta_{B_i} = d_{A_i} - d_{B_i}/2$), soit le milieu de d_{A_i} coïncide avec le début ou la fin de d_{B_i} (i.e., A et B vérifient $\texttt{oi}_i$(A, B) avec $\delta_{A_i} = d_{B_i} - d_{A_i}/2$ ou $\texttt{overlap}_i$(A, B) avec $\delta_{B_i} = d_{A_i}/2$) (cf. figure 3.7).

- **meet** : la fin de d_{B_i} coïncide avec le début de d_{A_i} (i.e., A et B vérifient $\texttt{mi}_i$(A, B)) ou que la fin de d_{A_i} coïncide avec le début de d_{B_i} (i.e., A et B vérifient $\texttt{meet}_i$(A, B)) (cf. figure 3.8).

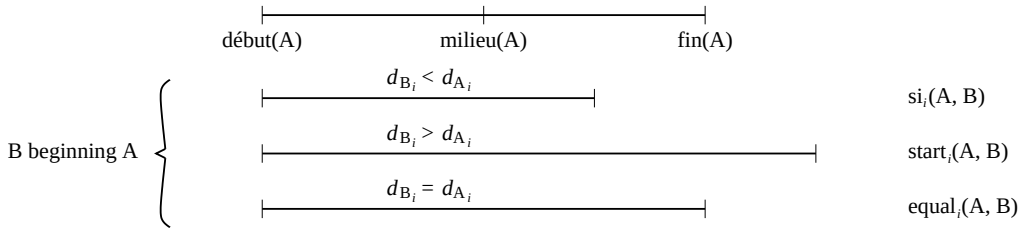


FIG. 3.4 – *Configuration* **beginning**

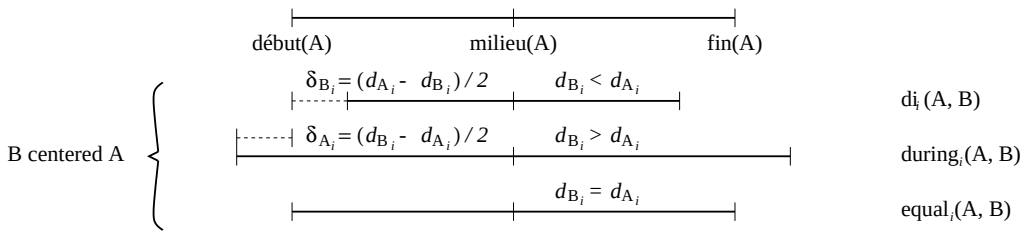


FIG. 3.5 – *Configuration* **centered**

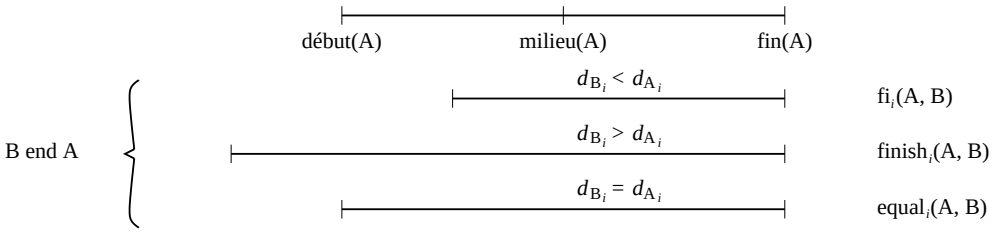


FIG. 3.6 – *Configuration* **end**

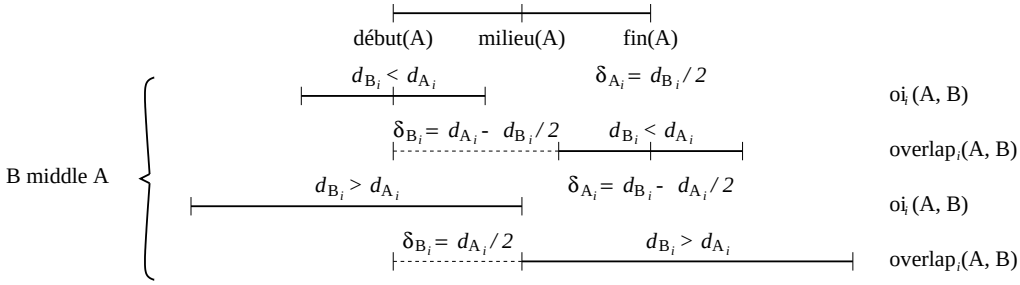


FIG. 3.7 – *Configuration* **middle**

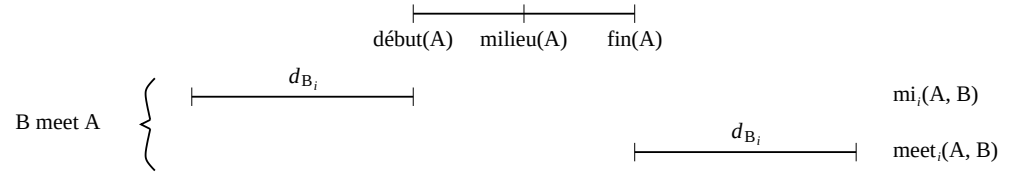


FIG. 3.8 – *Configuration* **meet**

3.4 Composition de présentations

Une présentation composite combine deux présentations A et B (atomiques ou composites) au moyen d'une relation spatio-temporelle $\sigma\tau$. Cette approche permet de procéder soit (1) par construction récursive de présentations ; ou (2) par description du contenu d'un objet réalisée à partir des présentations combinées par des relations $\sigma\tau$.

Dans notre modèle, la notion d'objet est polymorphe et subjective. Elle est fortement dépendante du cadre d'application considéré. Ainsi, nous pouvons soit partir de plusieurs objets à priori indépendants et les combiner dans une présentation, soit découper un objet existant en éléments que nous pouvons ainsi réutiliser dans d'autres présentations. Par exemple une carte de France représentée par une image au format GIF peut être considérée soit comme un tout, soit comme une combinaison d'objets élémentaires constitués par les régions ou les départements qui composent la France.

3.4.1 Description inter-média

La composition inter-média est une façon récursive de définir des présentations avec une expression de la forme : (OST, (OST$_A$, $\mathcal{O}_A$) $\sigma\tau$ (OST$_B$, $\mathcal{O}_B$)). Les OST$_A$, OST$_B$ décrivent respectivement les positions absolues et relatives des présentations de $\mathcal{O}_A$ et de $\mathcal{O}_B$, et transitivement les caractéristiques spatio-temporelles de l'espace où la présentation sera rendue visible (cf. figure 3.9).

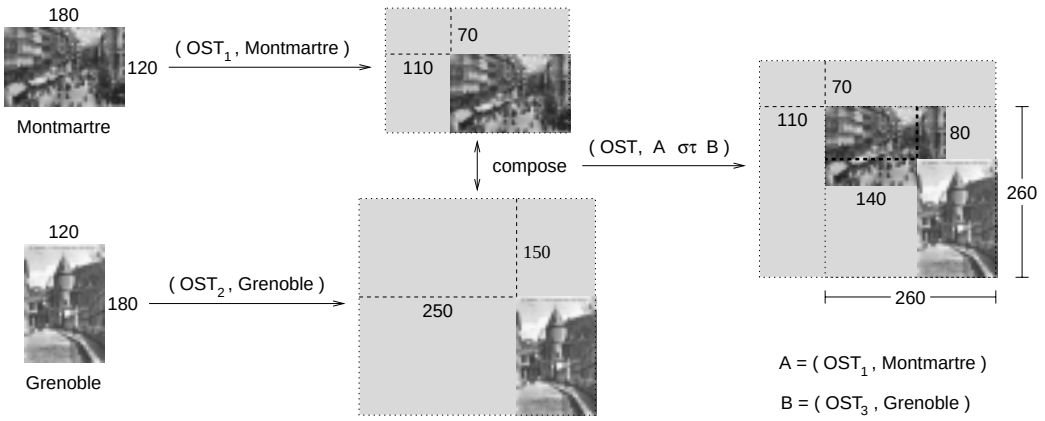


FIG. 3.9 – *Description inter-média*

Par exemple considérons la présentation de la figure 3.9 où l'image de Montmartre est affichée dans la position (110, 70) pendant 15 secondes. Ensuite, 5 secondes plus tard, une image de Grenoble est affichée à la position (250, 150) chevauchant l'image

de Montmartre du côté droit pendant 10 secondes. Cette présentation est définie par l'expression $\mathcal{P}_c$ = (OST, A $\sigma\tau$ B) où $\sigma\tau$ = ($\texttt{overlap}_x$, $\texttt{overlap}_y$, $\texttt{fi}_t$).

Le OST ($\delta_x = 110$, $\delta_y = 70$, $\delta_t = 0$, $d_x = 260$, $d_y = 260$, $d_t = 15$) spécifie la taille de la région qui englobe les deux images, correspondant au MBB (*Minium Bounding Box*[2]) et à la position de $\mathcal{P}_c$ dans un espace de présentation (cf. figure 3.9). La position de $\mathcal{P}_c$ est spécifiée par δ_x, δ_y, δ_t. Sa taille est définie par les valeurs de d_x, d_y, d_t. Nous verrons dans la suite la façon dont ces valeurs sont calculées en utilisant l'information contenue dans les OST des présentations de Montmartre et Grenoble (i.e., A et B) et par la relation spatio-temporelle $\sigma\tau$.

La présentation de Montmartre est représentée par l'expression suivante :

$$\begin{aligned} \text{A} = ((&\delta_{A_x} = 110,\ \delta_{A_y} = 70,\ \delta_{A_t} = 0,\\ &d_{A_x} = 180,\ d_{A_y} = 120,\ d_{A_t} = 15),\ \text{Montmartre}). \end{aligned}$$

Dans cet exemple, $\sigma\tau$ dénote dans les trois dimensions la position relative de B par rapport à A et permet d'interpréter le OST de B :

$$\begin{aligned} \text{B} = ((&\delta_{B_x} = 140,\ \delta_{B_y} = 80,\ \delta_{B_t} = 5,\\ &d_{B_x} = 120,\ d_{B_y} = 180,\ d_{B_t} = 10),\ \text{Grenoble}). \end{aligned}$$

Notons que δ_{B_x}, δ_{B_y}, δ_{B_t} sont calculés en prenant la présentation de Montmartre comme référence. En considérant la définition des prédicats de $\sigma\tau$ (cf. section 3.3), $\delta_{B_x} = 250 - \delta_{A_x}$; $\delta_{B_y} = 150 - \delta_{A_y}$; $\delta_{B_t} = d_{A_t} - d_{B_t}$.

En effet, la présentation résultante représente la situation où l'image de Montmartre est présentée à la position (110, 70) pendant 15 secondes et, 5 secondes après, l'image de Grenoble est montrée pendant 10 secondes superposant la présentation de Montmartre sur son coté droit. Notons que $\sigma\tau$ vérifient les définitions de la relation `Overlap` (`OV`) (cf. tableau 3.3) et de la relation `Southeast` (`SE`) (cf. tableau 3.4). Ces deux relations sont définies en utilisant la relation d'intervalle `overlap` sur `x` et `y`.

3.4.2 Description du contenu d'objets

Notre modèle peut être utilisé pour décrire le contenu des objets multimédias en utilisant des régions rectangulaires avec des images, des segments avec le son et des régions rectangulaires et des segments avec la vidéo. Prenons un exemple en partant

2. Dans un espace à trois dimensions, le concept de MBR est étendu à MBB [PTS97].

de l'objet FrenchMap de type image (335×340 pixels) montré dans la figure 3.10. À partir de cet objet, il est possible de décrire deux sous-objets caractérisés par les rectangles englobants R1 (Bretagne) défini par les points (8, 85) et (101, 140) et R2 (Aquitaine) défini par (74, 206) et (161, 316). Plusieurs possibilités sont offertes :

- créer une présentation où n'apparaissent que R1 et R2 en jouant sur la relation $\sigma\tau$ qui les relie (i.e., la région de Bretagne est disjointe et au nord-ouest de celle d'Aquitaine). Notons dans cet exemple que la présentation descriptive permet de récupérer soit la présentation composite, i.e., la position relative entre les deux régions, soit les présentations en absolue de chaque région (cf. figure 3.10).

- obtenir les objets Bretagne et Aquitaine correspondant à R1 et à R2 (i.e., les contenus des présentations) et leur appliquer des opérations spécifiques en leur associant des présentations indépendantes (cf., figure 3.10 à droite). Par exemple présenter en tons de gris toutes les régions qui sont au sud-est de Bretagne, en parallèle et pendant 17 secondes.

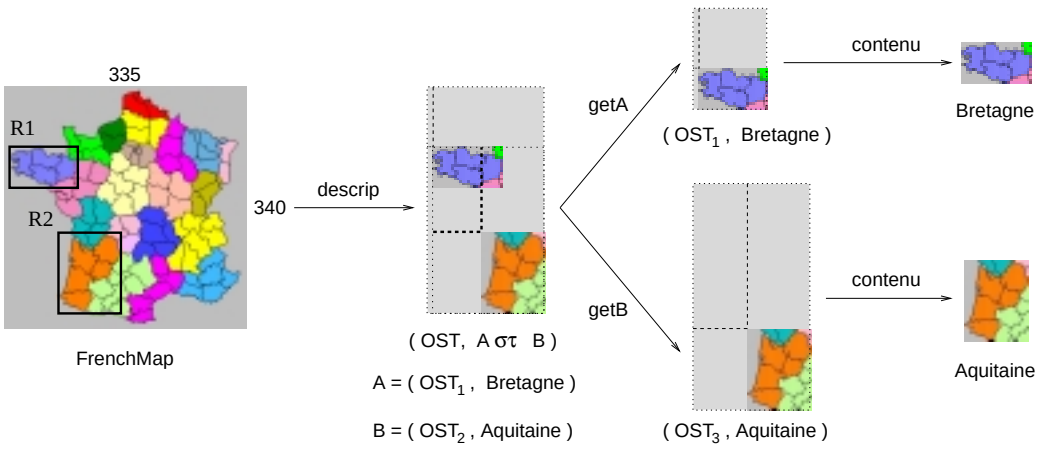


FIG. 3.10 – *Description intra-média*

Notons que notre modèle permet de représenter la composition et la description d'objets. Dans le premier cas, les présentations sont combinées en considérant les spécifications spatiales et temporelles données par un utilisateur ou une application. Dans le deuxième cas, la taille et la position des présentations sont déterminées par les caractéristiques des objets à décrire.

3.5 Comparaison avec d'autres modèles

Dans cette section, nous présentons les principaux modèles qui abordent les aspects spatiaux et temporels des objets. Ces modèles adoptent des approches différents selon leur contexte d'utilisation. Nous comparons les modèles par rapport à la façon utilisée pour décrire les objets (attributs spatiaux) et aux relations spatiales et/ou temporelles définies.

3.5.1 Modèles spatiaux

Les modèles spatiaux s'intéressent à la description d'un espace à deux dimensions en représentant les objets contenus dans cet espace et leur disposition au travers de relations directionnelles et/ou topologiques. Les modèles servent ensuite de support pour la définition des extensions spatiales de langages d'interrogation tels que SQL et OQL [Ege94].

[Ege94] décrit les aspects spatiaux des objets en utilisant un domaine générique *spatial* qui regroupe les domaines *spatial_0*, *spatial_1*, *spatial_2*, *spatial_3* utilisés pour représenter des objets dans des espaces à zéro, une, deux et trois dimensions. De manière générale, un objet est représenté par des surfaces (rectangles ou autres) décrites par des facettes externes et internes. La configuration de l'espace est décrite en spécifiant des relations topologiques (*disjoint*, *meet*, *overlap*, *inside/contains*, *covers/coveredBy* et *equal*) et directionnelles (*left/right*, *north/south*, *over/under*) entre ces surfaces.

Au lieu de représenter n'importe quelle surface comme [Ege94], les modèles proposés par [PTSE95, PT97, SS99] représentent les objets par des surfaces spécifiques telles que le MBR (*Minimum Bounding Rectangle*) [PTSE95, PT97] et le MBC (*Minimum Bounding Circle*) [SS99].

Dans le modèle de [PTS97], un objet est représenté par des MBR, MMB (*Minimum Bounding Box*) ou des hyper-rectangles qui décrivent son extension dans chacune des dimensions d'un espace à n dimensions. Cette approche revient à utiliser des intervalles pour représenter des aspects spatiaux des objets comme dans notre modèle. De manière similaire à notre modèle, les relations directionnelles et topologiques sont définies en termes d'intervalles. À la différence de notre approche, les relations topologiques et directionnelles ne sont pas combinées pour décrire des configurations spatiales.

Dans le modèle proposé par [SS99], un objet dans un espace à deux dimensions est représenté par un MBC. La configuration de l'espace est décrite par des relations topologiques et directionnelles définies pour des surfaces circulaires. L'intérêt de cette approche vient du fait que cette représentation semble améliorer le coût de calcul de similarité entre deux objets.

3.5.2 Modèles temporels

Les modèles temporels visent à représenter la façon dont un ensemble de présentations d'objets se synchronisent entre elles. Les présentations s'organisent par rapport à des instants appartenant à une ligne de temps, continue ou discrète.

[MS96] caractérise les présentations multimédias à travers les notions *media-instance* et *media-event*. Un *media-event* représente l'état global des médias dans une *media présentation* à un instant donné. Par exemple à un instant t une image de Grenoble apparaît dans l'écran en même temps qu'un son explicatif s'entend. La présentation est donc composée des deux *media events* l'image de Grenoble et le son qui se produisent à l'instant t. À la différence de notre modèle, [MS96] ne représente pas la synchronisation des médias dans le temps par l'intermédiaire de relations.

Des modèles tels que [Lay97, Ker97, Loz00] représentent la synchronisation des médias dans le temps par l'utilisation de relations d'intervalles. De manière similaire à notre modèle, la présentation d'un objet est représentée par un intervalle qui décrit le temps pendant lequel il est perceptible. [Loz00] utilise la notion d'ombre temporelle (intervalles) pour décrire la présentation d'un objet et définit une sémantique des relations d'Allen par rapport à ce concept.

Par contre, Madeus [Lay97] utilise quatre types d'attributs pour caractériser la présentation d'un objet : les attributs de structuration qui spécifient le format de codification, le contenu et l'identificateur d'un objet ; les attributs de présentations qui précisent des caractéristiques tels que la police pour les textes et le volume pour le son ; les attributs temporels qui permettent de caractériser la durée de présentations ; finalement, les liens hypermédias qui permettent de définir des liens de navigation ou des liens d'inclusion des autres présentations. Madeus utilise les relations d'Allen et des relations causales (*min*, *max*, *master*) pour la composition temporelle. Les relations causales spécifient la terminaison de la composition en forçant la terminaison de l'une de deux présentations qu'elles relient.

Menkalinan [DK95, Ker97] représente la présentation temporelle d'un objet

comme un seul intervalle. Des présentations composites sont définies en utilisant deux types de relations : temporelles et non temporelles. Les premières sont définies par des opérateurs de synchronisation, ils permettent d'encapsuler des intervalles à synchroniser dans un intervalle englobant. Les deuxièmes permettent la fusion de présentations et d'accéder aux attributs des présentations.

3.5.3 Modèles spatio-temporels

[GBE+00] propose un modèle spatio-temporel où les abstractions fondamentales sont *moving point* et *moving object*. Ces deux concepts permettent de représenter respectivement la position d'un objet dans le temps et la taille et la position d'un objet dans l'espace. Les deux aspects sont représentés par des types de données abstraits.

Les types spatiaux de base sont *point* représentant un point dans l'espace euclidien, *points* est un ensemble fini de points, *ligne* est un ensemble fini de courbes continues sur un plan et *region* est un ensemble fini de parties disjointes appelées facettes. Le type temporel de base est l'*instant* qui représente un point dans le temps linéaire et continu. Le point est donc isomorphe aux réels. Ensuite, la position d'un objet dans le temps est décrit par des *moving types* tels que *mpoint*, *mpoints*, *mligne*, *mregion*. Par exemple *mregion* représente la position d'une région dans le temps. Les objets sont donc représentés dans le temps et l'espace par des ensembles de points.

[Li98] utilise la notion d'intervalle pour représenter les aspects spatiaux et temporels entre les présentations des objets. Dans l'espace, il représente les objets par leurs projections sur `x` et `y` en utilisant le MBR et un centroïde qui est le centre du rectangle. Il considère douze relations directionnelles classées en trois catégories : quatre relations strictes (*north*, *south*, *west*, *eath*), quatre relations mixtes (*northwest*, *southeast*, *northeath*, *southwest*) et quatre relations de position (*above*, *below*, *left*, *right*). Ces relations sont définies en termes des relations d'Allen et ne considèrent que les dispositions où les MBR se touchent ou sont disjointes. Il considère six relations topologiques (*equal*, *inside*, *cover*, *overlap*, *touch*, *disjoint*). Pour les aspects temporels, les relations d'Allen sont utilisées pour spécifier l'ordre entre les intervalles temporels des présentations.

Le modèle défini par [VH95, Vaz96, VTS96] utilise la notion d'intervalle pour représenter les aspects spatiaux et temporels entre les présentations des objets. Pour les aspects temporels, il adopte les relations d'Allen en les caractérisant avec des valeurs pour spécifier la distance relative entre le début et la fin des intervalles de

durée. Dans l'espace, il adopte les relations topologiques et directionnelles entre des rectangles qui caractérisent la position et la taille des présentations.

3.6 Conclusion

Nous avons présenté notre modèle spatio-temporel basé sur la notion de présentation. Un objet est associé à une présentation qui décrit les attributs spatiaux et temporels avec lesquels il est rendu perceptible. Le modèle permet de représenter la présentation d'objets dans des espaces à une, deux, trois et quatre dimensions. La façon dont un ensemble de présentations sont organisées dans le temps et l'espace est décrite par des relations spatiales et/ou temporelles.

Les modèles temporels sont pour la plupart spécifiés selon deux approches : orientées instants ou intervalles. Les modèles orientés instants considèrent le temps comme un continuum linéaire de points correspondant à des instants. Toute présentation temporelle peut dans ce cas être représentée par un couple de deux points correspondant à ses instants de début et de fin. Les modèles orientés intervalles sont basés sur un modèle de temps relatif, c'est à dire qu'ils ordonnent le temps par des relations temporelles. De ce fait, l'intervalle temporel est considéré comme un élément atomique.

Notre modèle est orienté intervalles, cette approche permet en particulier d'avoir une représentation homogène des aspects spatiaux et temporels, mais aussi d'avoir une définition de relations spatiales et temporelles en termes d'intervalles. Enfin, le modèle représente des informations qualitatives et quantitatives, ce qui permet d'avoir une représentation plus fine de la façon dont un ensemble d'objets s'organise dans l'espace-temps.

Dans les modèles à intervalles, un objet est représenté par des projections dans le temps modélisées par des intervalles. Ceci permet de construire des présentations en utilisant des relations d'intervalles.

Par ailleurs, nous constatons que la diversité des modèles vient du fait qu'ils sont définis avec des objectifs différents. Des modèles comme [PS94, PD97, PKA99] privilégient la récupération d'objets multimédias en prenant en compte des critères de similarité et de reconnaissance du contenu par des relations. La représentation des objets adoptée par ces modèles est souvent bien adaptée pour définir des stratégies d'indexation supportant la récupération des données.

En général, les modèles spatiaux sont utilisés dans des applications géographiques. Lorsque les aspects temporels sont pris en compte, la plupart des modèles supportent

la représentation du mouvement des objets dans un espace à deux dimensions. Ces modèles se présentent comme des extensions des modèles existants tels que le modèle relationnel. D'autres profitent des caractéristiques du modèle à objets pour représenter les caractéristiques spatiales et temporelles des objets.

Notre modèle vise à fournir une représentation pivot qui doit supporter l'intégration de sources et des applications hétérogènes. Cette intégration est assurée le mécanisme de médiation que nous proposons dans ce travail.

Chapitre 4

Langage OQLiST

Dans ce chapitre, nous présentons le langage OQLiST, une extension du langage OQL (*Object Query Language*) proposé par ODMG [CBB+00], pour la spécification et la manipulation spatio-temporelle de présentations d'objets. La section 4.1 introduit les concepts de base. La section 4.2 présente les constructeurs de présentations de OQLiST. La section 4.3 montre les opérateurs appliqués aux présentations. La section 4.4 fait une classification des types de requêtes et décrit la construction de présentations comme résultats de requêtes. Enfin, la section 4.5 conclut ce chapitre.

Dans tout ce chapitre, les opérateurs et les fonctions réalisables par le langage sont illustrées au moyen de requêtes qui ont été exécutées par notre prototype décrit au chapitre 6.

4.1 Concepts de base

OQLiST aborde le problème de l'ordonnancement de présentations dans un espace de présentation à trois dimensions `x`, `y`, `t` où chaque dimension est considérée comme un continuum avec une origine définie par la position (0, 0, 0). Pour l'espace, cette origine est définie comme le coin supérieur gauche d'un écran, d'une fenêtre ou d'une région sur un navigateur Web. Pour le temps, l'origine est fixée au début de l'exécution de la présentation.

4.1.1 Présentation de OQL

OQL est un langage fonctionnel qui regroupe des opérateurs élémentaires tels que des opérateurs de création, de modification ou de consultation d'objets mais aussi des

opérateurs plus évolués qui effectuent des itérations sur collections tels que le bloc select-from-where, l'opérateur de groupement ou les quantificateurs [Clu98]. Dans les paragraphes suivants, nous présentons brièvement la syntaxe OQL en classant ces principaux types de requêtes.

Constructeurs

Les constructeurs sont des opérateurs de requêtes qui créent tous les types de données de l'ODMG (entiers, réels, chaînes de caractères, structures, collection, etc.). Par exemple :

```
"file://localhost/FrenchMap.gif";        /* création d'une chaîne de caractères */
9.5;                                     /* création d'un réel */
set( 7, 1, 2 );                          /* création d'un ensemble d'entiers */
list( "Grenoble.gif", "Montmartre.gif"); /* création d'une liste */

define FrenchMap as                      /* création d'un objet nommé */
        Image( URL = "file://localhost/FrenchMap.gif" )
```

Les quatre premières requêtes construisent des littéraux (objets identifiés par leur contenu). La dernière requête construit un objet de type Image et lui associe l'adresse URL "file://localhost/FrenchMap.gif".

Accès aux données

OQL fournit également des primitives qui accèdent aux données en utilisant les noms des objets, des attributs et des méthodes :

```
FrenchMap;                               /* accès par le nom */
FrenchMap.crop( 8, 85, 93, 55 )          /* accès à une méthode */
```

Parcours de collections

Enfin, OQL fournit des primitives de haut niveau pour manipuler les collections d'objets (listes, ensembles). Ainsi, OQL définit les opérateurs ensemblistes (union, intersection, test d'appartenance), des opérateurs de quantification (universelle, existentielle), un bloc select-from-where, un opérateur de tri (order by), un opérateur de

groupement (group by) ... Par exemple retrouver la liste des images avec une largeur inférieur à 10 pixels :

```
select e from e in Images
where e.getHeight < 10 order by *
```

4.1.2 Objets

OQLiST est un langage de présentations, le type présentation est donc le seul type d'objet qu'il manipule. Tout autre type est associé à un objet présentation (par défaut) pour pouvoir être manipulé. La figure 4.1 illustre les types d'objets définis par les classes Text, Image, Audio et Video qui sont généralisées par la classe Document. Ces objets sont caractérisés par une adresse URL, un nom et une liste de mots clés. Nous détaillons ensuite les présentation par défaut associées à ces types.

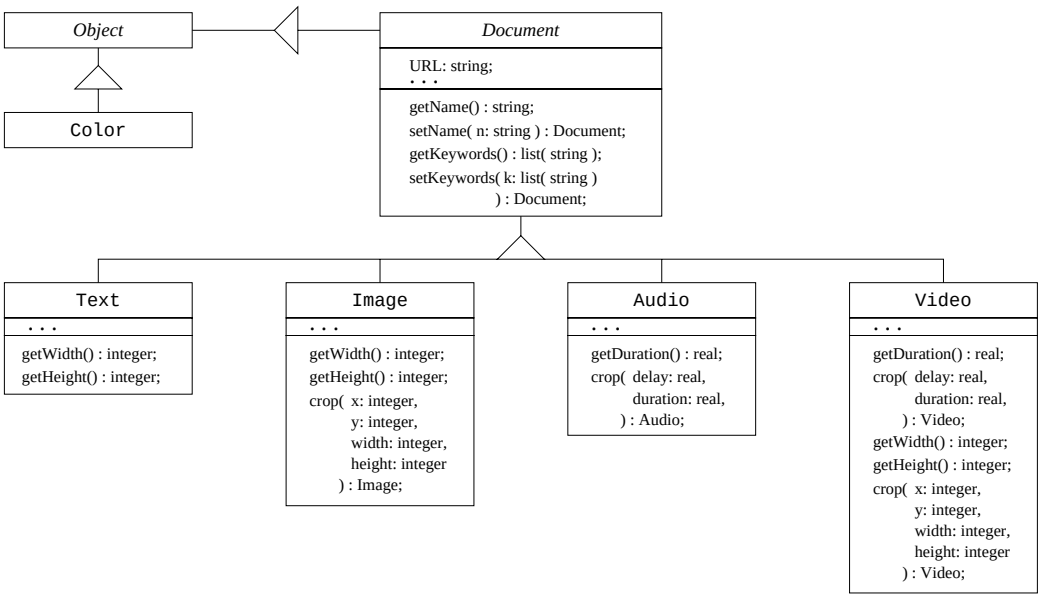


FIG. 4.1 – *Objets*

Textes et images

Dans un espace 2D, les textes et les images sont présentés par défaut à l'origine avec une taille définie par leurs méthodes getWidth() et getHeight(). Pour un objet de type Text sa taille est calculée en fonction de la police utilisée. La taille d'un objet de type Image correspond à celle définie dans son fichier source de type GIF, PIC, etc. Les

types d'objets qui ne possèdent pas des caractéristiques temporelles inhérentes tels que les textes et les images sont décrits par un délai nul et une durée indéterminée. Par exemple Image(URL = "file://localhost/FrenchMap.gif") construit un objet de type Image dont sa présentation par défaut présente la carte de France FrenchMap à l'origine de l'espace sans délai et avec une durée indéterminée.

Sons et vidéos

Les sons et les vidéos sont de données avec des attributs spatiaux et temporels intrinsèques en général définis par leurs fichiers source. Ces attributs sont utilisés pour leur associer une présentation par défaut. Cette présentation positionne un objet de type Video et Audio à l'origine sans délai dans le temps ni décalage dans l'espace. Dans les cas d'un objet de type Audio, sa position et sa taille sont indéterminées dans sa présentation par défaut.

Collections

Une collection peut être de type liste (list) ou ensemble (set). La présentation par défaut d'une liste décrit un ordre sur les présentations de ses objets (i.e., l'ordre de définition). Elle compose ces présentations par rapport au même référentiel. L'ordre permet de donner une profondeur à chaque élément simulée en superposant les présentations dans l'espace (cf. figure 4.2). Dans le temps, les présentations maintiennent leurs caractéristiques temporelles (délai et durée).

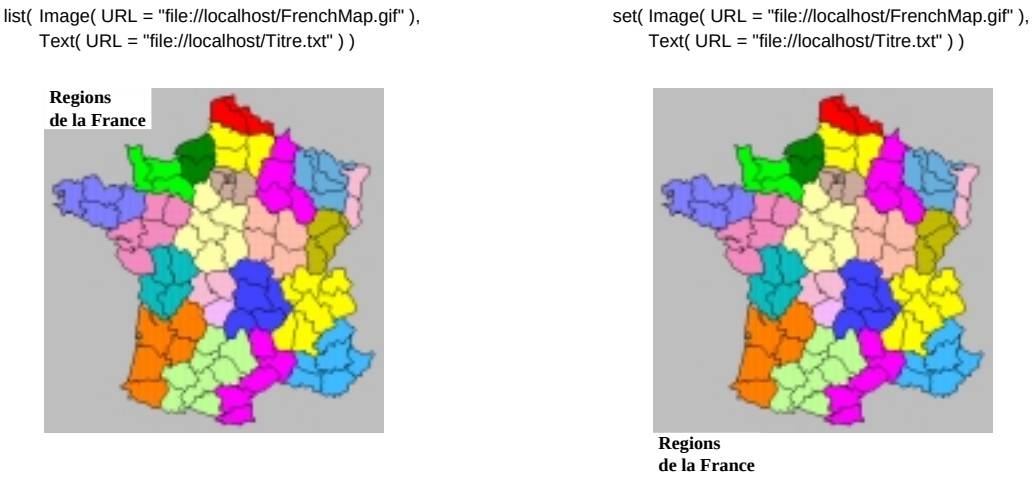


FIG. 4.2 – *Présentation par défaut de deux collections*

Un ensemble est présenté comme une liste d'éléments dans une colonne de présentations. Dans ce cas, il n'y a pas de notion de profondeur puisque les présentations sont affichées les unes au-dessous des autres (cf. figure 4.2). En plus, elles sont présentées toutes en même temps. Toute présentation appartenant à l'ensemble maintient sa durée mais pas son délai.

Couleurs

La classe Color caractérise des objets décrivant des couleurs à travers une spécification RGB. Tout objet de type couleur est associé à une présentation par défaut qui décrit une région de taille 10×10 pixels de couleur blanche (white), noire (black), grise (lightGray , gray, darkGray), rouge (red), verte (green), bleue (blue), etc.

Méthodes

Les objets médias (texte, image, son, vidéo) ont des méthodes. Nous nous intéressons particulièrement à la méthode crop car elle nous permettra de montrer son utilisation dans la description du contenu des médias (cf. section 4.4.3). Cette méthode permet de découper soit une région dans une image, soit un segment dans un son, soit une région et un segment dans une vidéo. Cette méthode est utilisée pour décrire et récupérer des morceaux du contenu des médias. Par exemple nous pouvons présenter la région définie par les points (60, 40) et (100, 80) d'une vidéo à partir de la seconde 10 et jusqu'à la seconde 19.5 comme suit (cf. figure 4.3) :

```
set( Video( URL = "file://localhost/inwater.mov" ).crop( 10.0, 9.5 ),
     Video( URL = "file://localhost/inwater.mov" ).crop( 10.0, 9.5 )
                                                   .crop( 60, 40, 40, 40 ) )
```

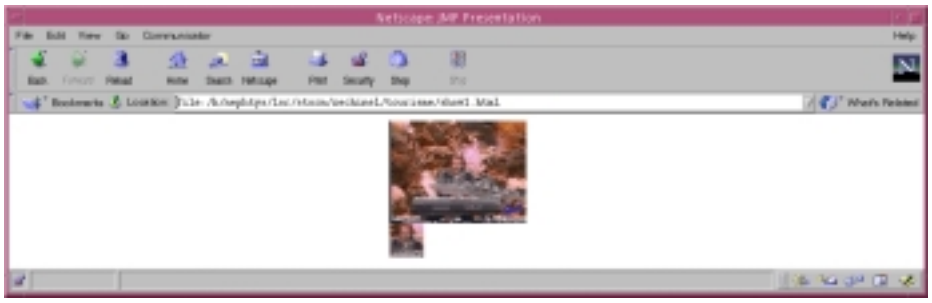


FIG. 4.3 – *Vidéo découpée*

4.1.3 Présentations

Les présentations sont représentées par des structures arborescentes où les nœuds intermédiaires sont des présentations composites et les feuilles sont des présentations atomiques. Dans cette structure, une présentation composite est un arbre binaire qui combine deux présentations (atomiques ou composites) nommées A et B. Une présentation atomique est caractérisée par la méthode `content` qui retourne l'objet qu'elle présente.

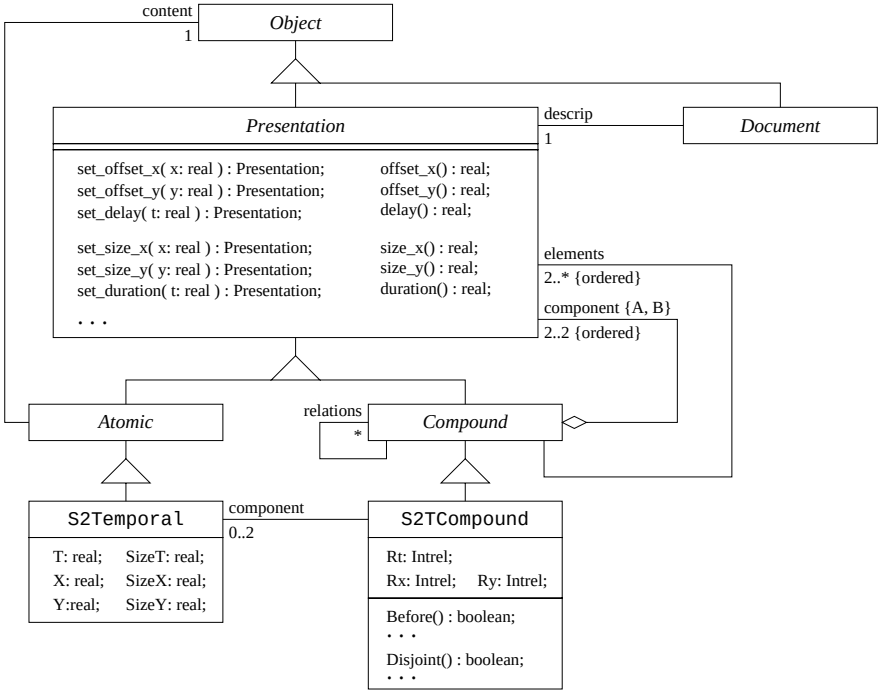


FIG. 4.4 – *Schéma de présentations*

Attributs spatio-temporels

Les attributs spatio-temporels des présentations prennent leur valeur dans le domaine des réels. Nous représentons la valeur infinie par la valeur `Free` et nous interprétons un attribut affecté par cette valeur comme indéterminé. Par exemple un délai `Free` spécifie une présentation qui attend indéfiniment avant d'être présentée. Une présentation avec une durée indéterminée (`Free`) est perceptible indéfiniment. De manière analogue, une position indéterminée est placée à l'infini, alors qu'une taille `Free` a une longueur infinie.

Les méthodes offset_x, offset_y, size_x, size_y, delay et duration permettent d'interroger les attributs spatiaux et temporels d'une présentation. Les deux premières donnent en sortie sa position dans l'espace ; les méthodes size_x et size_y renvoient sa longueur et sa largeur. Les deux dernières donnent son délai et sa durée.

Composants d'une présentation

Les composants d'une présentation sont récupérés avec la méthode elements. Cette méthode prend en entrée une présentation et donne en sortie la liste de présentations composantes avec leurs positions absolues. Par exemple $\mathcal{P}_{abcd}$.elements renvoit la liste de présentations [$\mathcal{P}_a$, $\mathcal{P}_b$, $\mathcal{P}_c$, $\mathcal{P}_d$] (cf. figure 4.5). L'ordre des éléments de la liste résultat correspond à l'ordre de spécification des présentations composantes. Cet ordre est utilisé pour donner à chaque présentation une profondeur dans un espace 2D.

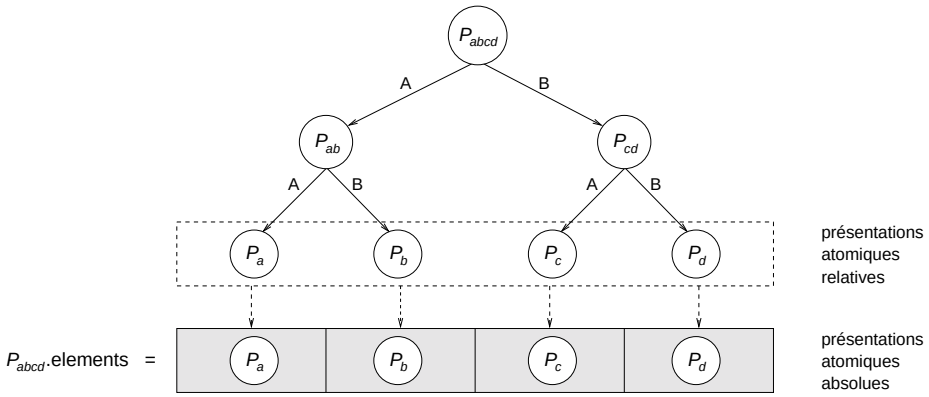


FIG. 4.5 – *Éléments des présentations*

OQLiST étend les opérateurs d'OQL first et last. Ils prennent comme argument une présentation $\mathcal{P}_{abcd}$ et ils retournent le premier et le dernier élément de $\mathcal{P}_{abcd}$ (par ex. first($\mathcal{P}_{abcd}$) = $\mathcal{P}_a$ et last($\mathcal{P}_{abcd}$) = $\mathcal{P}_d$). Dans le cas d'une présentation atomique, les opérateurs first et last retournent la même présentation (cf. figure 4.6).

Les éléments d'une présentation composite ont une profondeur associée qui correspond à leur position dans l'arbre de présentation. Une présentation composite dispose la méthode get. Soit i un entier avec $0 <= i <$ count($\mathcal{P}_{abcd}$.elements), $\mathcal{P}_{abcd}$.get(i) permet de récupérer l'élément à la $(i+1)$ième position.

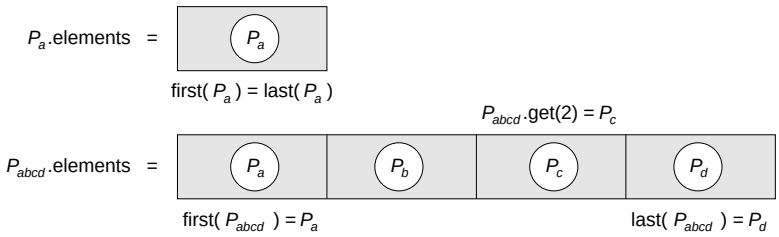


FIG. 4.6 – *Sélection des éléments*

Relations dans une présentation

Une présentation composite relie deux présentations la composant. Par exemple $\mathcal{P}_{abcd}$ représente la relation entre $\mathcal{P}_{ab}$ et $\mathcal{P}_{cd}$. Toute présentation composite implante des prédicats (méthodes booléennes) qui permettent de vérifier si ses présentations composantes maintiennent ou pas une relation spatiale et/ou temporelle particulière. Les tableaux 4.1 et 4.2 présente les prédicats temporels et spatiaux.

r.op est un prédicat où op est une relation temporelle (Before, Meet, etc.), topologique (Disjoint, Touch, etc.) ou directionnelle (North, South etc.). Soit $\mathcal{P}_{abcd}$ = (OST, A $\sigma\tau$ B) une présentation composite, le prédicat r.op($\mathcal{P}_{abcd}$) est vérifié si $\sigma\tau$ vérifient la relation op.

Prédicat temporel	Inverse
r.Before	r.Bi
r.Meet	r.Mi
r.Overlap	r.Oi
r.Start	r.Si
r.During	r.Di
r.Finish	r.Fi
r.Equal	

TAB. 4.1 – *Prédicats temporels*

Espace de relations dans une présentation Il définit toutes les relations établies entre les éléments feuilles d'une présentation composite. Une présentation avec n éléments feuilles a au total $n * (n-1)$ relations. Par exemple dans la présentation composite Spatial 2D présentée dans la figure 4.7, il y a 6 relations : North(Montmartre, Grenoble), West(Montmartre, Chambery) et Southeast(Grenoble, Chambery) et leurs inverses.

La méthode $\mathcal{P}$.relations calcule toutes les relations binaires spatio-temporelles entre

Prédicat topologique	Prédicat directionnel
r.Disjoint	r.North
r.Touch	r.South
r.Spatial_Overlap	r.West
r.Cover	r.East
r.Covered_by	r.Northwest
r.Inside	r.Northeast
r.Contain	r.Southwest
r.Spatial_Equal	r.Southeast

TAB. 4.2 – *Prédicats spatiaux*

FIG. 4.7 – *Relations dans une présentation composite*

les éléments d'une présentation composite (sans considérer les inverses). Le résultat est une collection de présentations composites dont les éléments sont des présentations atomiques. Par exemple, pour les présentations $\mathcal{P}_a$, $\mathcal{P}_{ab}$ et $\mathcal{P}_{abcd}$ montrées à la figure 4.5, nous avons :

- $\mathcal{P}_a$.relations = { },
- $\mathcal{P}_{ab}$.relations = { $\mathcal{P}_{ab}$ },
- $\mathcal{P}_{abcd}$.relations = { $\mathcal{P}_{ab}$, $\mathcal{P}_{cd}$, $\mathcal{P}_{ac}$, $\mathcal{P}_{ad}$, $\mathcal{P}_{bc}$, $\mathcal{P}_{bd}$ }.

Enfin, la méthode inverse calcule récursivement les relations inverses dans une

présentation composite. Elle prend une présentation $\mathcal{P}_{ab...n}$ et elle retourne une présentation $\mathcal{P}_{n...ba}$.

4.2 Constructeurs

4.2.1 Présentations atomiques et composites

La construction de présentations de type atomique et composite en OQLiST se fait respectivement au travers des opérateurs atomic et compound. Soient $\mathcal{O}_1$, $\mathcal{O}_2$ et $\mathcal{O}$ des expressions qui dénotent des objets :

- atomic $\mathcal{O}$ construit une présentation atomique de $\mathcal{O}$ en lui associant une présentation par défaut. Si $\mathcal{O}$ est déjà une présentation atomique, cette opération donne en résultat la même présentation. Par exemple l'expression suivante définit une présentation de l'image de la carte de France FrenchMap :

  ```
  atomic Image( URL = "file://localhost/FrenchMap.gif" )
  ```

  Dans cette présentation l'image FrenchMap est présentée par défaut à l'origine, sans changement de taille, sans délai et avec une durée indéterminée.

- $\mathcal{O}_1$ compound $\mathcal{O}_2$ combine les présentations par défaut des objets $\mathcal{O}_1$ et $\mathcal{O}_2$ dans une présentation composite. Elle caractérise la relation spatio-temporelle entre les présentations de $\mathcal{O}_1$ et $\mathcal{O}_2$. Par exemple nous pouvons combiner une image du drapeau français avec le son de la Marseillaise comme suit :

  ```
  Image( URL = "file://localhost/FrenchFlag.gif" )
  compound Audio( URL = "file://localhost/Marseillaise.au" )
  ```

  Dans cette expression, le constructeur compound combine les présentations par défaut de l'image FrenchFlag et du son Marseillaise.

4.2.2 Décor des présentations

Nous présentons ensuite quatre constructeurs qui associent des caractéristiques statiques (par ex. un cadre, un arrière plan) et dynamiques (un hyperlien) aux présentations.

Soient $\mathcal{O}_1$, $\mathcal{O}_2$ et $\mathcal{O}$ des expressions qui dénotent des objets, W et H des expressions de type réel et URL une expression de type chaîne de caractères :

– $\mathcal{O}$ border W, H spécifie une présentation où $\mathcal{O}$ est entourée d'un cadre. La taille de $\mathcal{O}$ dans la nouvelle présentation est calculée par rapport à la taille de sa présentation par défaut et de W et H. Soit X et soit Y la largeur et la longueur de la présentation par défaut de $\mathcal{O}$, la largeur et longueur de $\mathcal{O}$ border W, H est $X - (2 * W) \times Y - (2 * H)$. $\mathcal{O}$ est affiché avec une bordure de W sur la largeur et H sur la longueur (cf. figure 4.8).

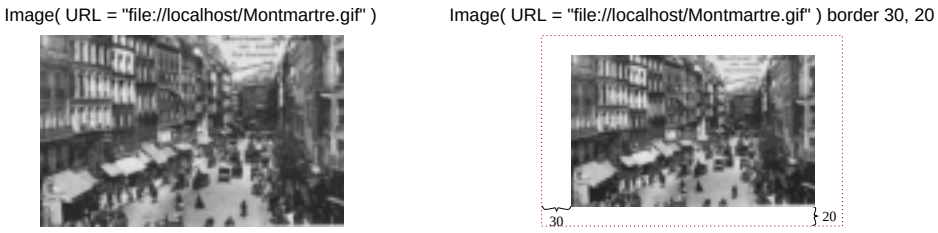


FIG. 4.8 – *Opérateur* border

Une propriété des présentations ainsi construites est que les opérations de redimensionnement n'affectent pas le cadre (cf. annexe B). Nous verrons dans la suite comment profiter de cette propriété pour définir des tableaux de présentations (cf. section 4.2.4).

– $\mathcal{O}_1$ background $\mathcal{O}_2$ caractérise une présentation de $\mathcal{O}_1$ avec comme arrière plan la présentation de $\mathcal{O}_2$. La présentation de $\mathcal{O}_2$ a les mêmes caractéristiques spatio-temporelles que celle de $\mathcal{O}_1$.

Les opérateurs background et border peuvent être combinés pour associer un arrière plan au contour de la présentation d'un objet. (cf. figure 4.9).

– $\mathcal{O}$ button URL associe la présentation de $\mathcal{O}$ avec un bouton lié à l'adresse URL. En cliquant il affiche la présentation du contenu de cette adresse.

– $\mathcal{O}_1$ link $\mathcal{O}_2$ construit une présentation de $\mathcal{O}_1$ qui, en cliquant dessus, permet de visualiser la présentation de $\mathcal{O}_2$.

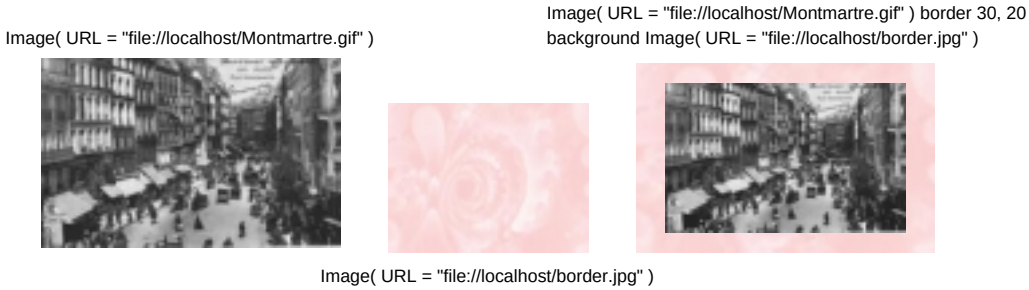


FIG. 4.9 – *Les opérateurs* border *et* background

4.2.3 Positionnement et taille des présentations

Les constructeurs ci-dessous construisent de nouvelles présentations en modifiant les caractéristiques spatio-temporelles des présentations existantes ou de celles qui sont définies par défaut dans le contexte de présentation. Ils permettent de caractériser la position (par rapport à un référentiel absolu), la taille, le délai et la durée des présentations. Soient X, Y, D, W, H des expressions de type réel, $\mathcal{O}$ une expression de type objet et $\mathcal{P}$ une expression de type présentation :

- $\mathcal{O}$ at X, Y construit une présentation de $\mathcal{O}$ positionnée à la coordonnée (X, Y). Par exemple $\mathcal{O}$ at 100, 100 construit une présentation qui présente $\mathcal{O}$ à la position (100, 100).

- $\mathcal{O}$ at D construit une présentation avec un délai égal à D, i.e., D est la position de la présentation de $\mathcal{O}$ sur l'axe t. Par exemple en supposant que le temps est mesuré en secondes, l'expression $\mathcal{O}$ at 10 présentera $\mathcal{O}$ à la seconde 10.

- $\mathcal{O}$ with W, H construit une présentation de $\mathcal{O}$ de longueur W et largeur H.

- $\mathcal{O}$ fit W, H construit une présentation en respectant la relation de proportion entre la longueur W et la largeur H de la présentation par défaut de $\mathcal{O}$ (cf. la définition de fit dans l'annexe A).

- $\mathcal{O}$ during D construit une présentation de $\mathcal{O}$ avec une durée de D secondes. En le combinant avec d'autres opérateurs, nous pouvons présenter, par exemple l'image de la carte de France FrenchMap avec une taille de 20×30 pixels, à la position (100, 100), avec un délai de 5 secondes et une durée de 10 secondes :

```
Image( URL = "file://localhost/FrenchMap.gif" ) at 100, 100 with 20, 30
                                                at 5 during 10
```

– $\mathcal{O}$ using $\mathcal{P}$ prend les caractéristiques spatio-temporelles de $\mathcal{P}$ (la position, la taille, la durée et le délai) pour construire une présentation de $\mathcal{O}$.

4.2.4 Présentations prédéfinies pour des collections d'objets

OQLiST fournit des opérateurs qui spécifient des organisations spatio-temporelles prédéfinies pour la présentation de collections d'objets. Par exemple *présenter en parallèle les cellules d'un tableau*. Dans cette spécification, *tableau* et *en parallèle* décrivent respectivement l'organisation spatiale et temporelle de la présentation.

Tableaux d'objets Soient $\mathcal{C}$ une expression qui dénote une collection d'objets et N de type entier :

– $\mathcal{C}$ table construit un tableau où les nombres de lignes et de colonnes sont calculés par rapport à la cardinalité de $\mathcal{C}$. Les cellules du tableau ont toutes la même taille et les présentations ont la même taille que la cellule.

– $\mathcal{C}$ table fit construit un tableau où le nombre de lignes et de colonnes sont calculés par rapport à la cardinalité de $\mathcal{C}$. Les cellules ont toutes la même taille et les présentations sont adaptées de manière proportionelle à cette taille.

– $\mathcal{C}$ table N construit un tableau à N colonnes dans lesquelles s'organisent les présentations des objets de la collection $\mathcal{C}$. Toutes les cellules du tableau ont la même taille.

– $\mathcal{C}$ table fit N construit un tableau à N colonnes. Les cellules du tableau ont toutes la même taille et l'opérateur fit adapte proportionnellement les présentations des objets à celle-ci.

A titre d'exemple, supposons l'existence de trois présentations de la France, à savoir Disposition, FrenchCitiesViews et Destinations dans une base de données. Ces présentations illustrent différentes facettes de la France : disposition géographique, vues historiques et destinations touristiques. Chacune est une présentation composite avec des caractéristiques spatiales et temporelles différentes. Par exemple FrenchCitiesViews synchronise la présentation d'une image de la carte de France avec une présentation séquentielle d'images des villes françaises. L'expression OQLiST suivante définit un tableau à trois colonnes pour ces présentations :

list(atomic Disposition, atomic FrenchCitiesViews, atomic Destinations) table fit 3
with 1050, 284 background white

Le tableau et de taille 1050×284 pixels (cf. with) avec un arrière plan blanc (cf. background). L'opérateur atomic permet de rendre atomique les présentations composites en faisant abstraction des caractéristiques spatio-temporelles de leurs composants. Notons dans la figure 4.10 qu'il y a une séparation entre chaque cellule du tableau. Ceci est dû à l'adaptation proportionnelle (cf. fit) des présentations.

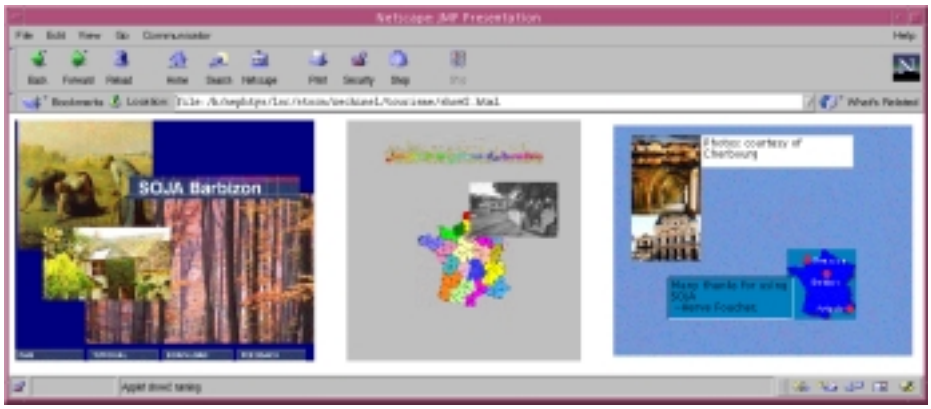


FIG. 4.10 – *Opérateur* table fit

Organisation d'une collection d'objets Soient $\mathcal{C}$ une expression qui dénote une collection d'objets, D une expression de type réel et A $\in$ { x, y, t } un axe d'un espace à trois dimensions :
$\mathcal{C}$ arrange all D on A construit une liste L de présentations des objets de $\mathcal{C}$ avec D sur l'axe A entre les présentations. D spécifie la position relative de chaque élément à partir de la position de l'élément précédent. L'opérateur arrange all peut être combiné avec cinq opérateurs qui permettent de déterminer la position des présentations de L entre elles. Soient L = ($\mathcal{C}$ arrange all D on A) une liste de présentations, $\mathcal{P}_i$ et $\mathcal{Q}_j$ des éléments de L tel que $\mathcal{Q}_j$ précède $\mathcal{P}_i$ avec $j = i - 1$ et $0 <= i <$ card(L). Soit D :

- beginning, les éléments de L sont organisés avec un décalage nul (cf. tableau 4.3).
- centered, les éléments de L sont centrés les uns par rapport aux autres (cf. tableau 4.3).
- end, les points de fin de toutes les présentations de L sont les mêmes (cf. tableau 4.3).

– middle, l'un des deux cas suivants est possible : soit $d_{\mathcal{P}_{iA}} <= d_{\mathcal{Q}_{jA}}$ le point de milieu de $\mathcal{P}_i$ correspond au point de fin de $\mathcal{Q}_j$, soit $d_{\mathcal{P}_{iA}} > d_{\mathcal{Q}_{jA}}$ le point de début de $\mathcal{P}_i$ correspond à celui de milieu de $\mathcal{Q}_j$ (cf. tableau 4.3).

– meet, chaque élément $\mathcal{P}_i$ est positionné à la fin de $\mathcal{Q}_j$ (cf. tableau 4.3).

Opérateur	Définition
beginning	$\delta_{\mathcal{P}_{iA}} = 0$
centered	$\delta_{\mathcal{P}_{iA}} + d_{\mathcal{P}_{iA}}/2 = \delta_{\mathcal{Q}_{jA}} + d_{\mathcal{Q}_{jA}}/2$
end	$\delta_{\mathcal{P}_{iA}} + d_{\mathcal{P}_{iA}} = \delta_{\mathcal{Q}_{jA}} + d_{\mathcal{Q}_{jA}}$
middle	$d_{\mathcal{P}_{iA}} <= d_{\mathcal{Q}_{jA}} \rightarrow \delta_{\mathcal{P}_{iA}} + d_{\mathcal{P}_{iA}}/2 = \delta_{\mathcal{Q}_{jA}} + d_{\mathcal{Q}_{jA}} \vee$ $d_{\mathcal{P}_{iA}} > d_{\mathcal{Q}_{jA}} \rightarrow \delta_{\mathcal{P}_{iA}} = \delta_{\mathcal{Q}_{jA}} + d_{\mathcal{Q}_{jA}}/2$
meet	$\delta_{\mathcal{P}_{iA}} = \delta_{\mathcal{Q}_{jA}} + d_{\mathcal{Q}_{jA}}$

TAB. 4.3 – *Organisation de collections*

En reprenant l'exemple des présentations de la France, nous pouvons définir un tableau à trois colonnes de taille 1050×568 pixels avec un arrière plan blanc contenant les présentations Disposition, FrenchCitiesViews et Destinations (cf. figure 4.11). L'expression OQLiST suivante spécifie cette présentation :

```
list( atomic Disposition, atomic FrenchCitiesViews, atomic Destinations ) table fit 3
arrange all meet on x
arrange all middle on y with 1050, 568 background white
```

Les présentations dans ce tableau sont organisées les unes après les autres sur x grâce à l'opérateur meet et en escalier sur y grâce à l'opérateur middle.

Organisation des présentations OQLiST spécialise les constructeurs des collections présentés ci-dessus pour organiser les éléments d'une présentation composite. Par exemple l'expression suivante spécifie une présentation composite de Disposition, FrenchCitiesViews et Destinations (toutes rendues atomiques). Dans ce cas, table fit manipule trois éléments :

```
atomic Disposition compound atomic FrenchCitiesViews
                    compound atomic Destinations table fit 3
arrange all meet on x
arrange all middle on y with 1050, 568 background white
```

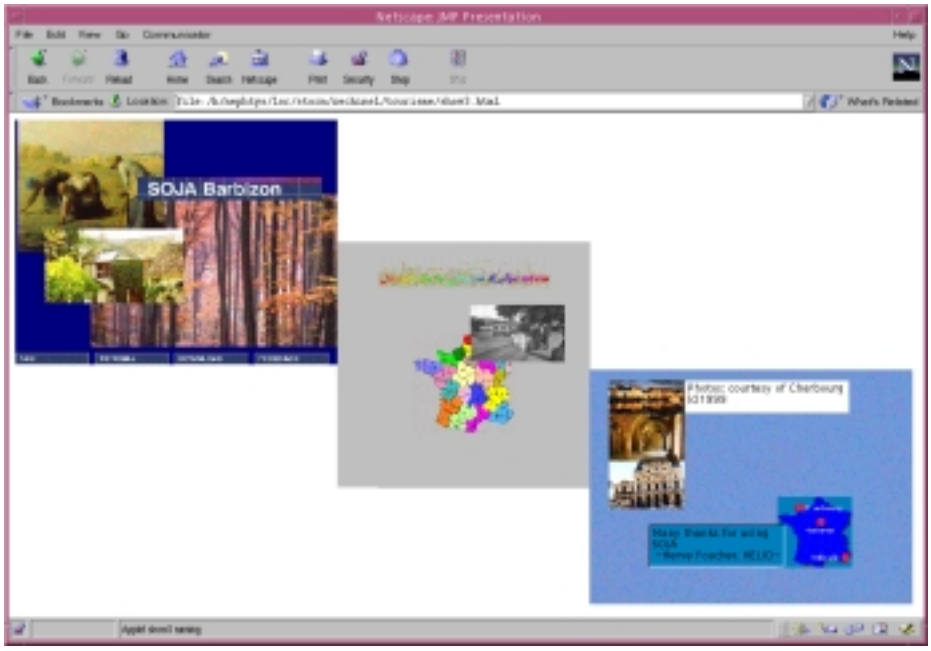


FIG. 4.11 – *Opérateur* arrange all

Afin de mettre en évidence l'intérêt de ces opérateurs pour des présentations, prenons $\mathcal{PC}$ = (atomic Disposition compound atomic FrenchCitiesViews compound atomic Destinations background white) et la définition de table fit sur les collections. Pour pouvoir réorganiser les éléments de $\mathcal{PC}$ dans un tableau, il faut d'abord obtenir la liste d'éléments qui composent $\mathcal{PC}$ et puis, appliquer l'opérateur table fit, i.e., $\mathcal{PC}$.elements table fit 3. Lors de la conversion de $\mathcal{PC}$ en collection, notre définition perd les attributs de $\mathcal{PC}$ comme l'arrière plan ainsi que les relations entre les éléments de $\mathcal{PC}$. Cela n'est pas le cas avec table fit sur les présentations.

Si nous choisissons de garder seulement les opérateurs sur les présentations, ce sont les expressions select-from-where qui se compliquent. Il faut composer les éléments de la collection résultante du select-from-where avant d'appliquer l'opérateur table fit. Enfin, la solution d'avoir les deux types d'opérateurs nous donne plus d'expressivité en permettant de les combiner. Par exemple :

```
list( atomic Disposition, atomic FrenchCitiesViews, atomic Destinations ) table fit 3
background white arrange all meet on x
                arrange all middle on y with 1050, 568
```

combine table fit qui prend une liste de présentations, background qui fait de la liste résultante une présentation composite et, finalement, arrange all et with qui organisent

et adaptent respectivement la présentation résultante.

4.3 Ordonnancement spatial et/ou temporel

OQLiST fournit des opérateurs pour définir des relations binaires entre deux composants d'une présentation composite. Ces deux composants sont nommés référence et cible. Lorsqu'il s'agit d'une présentation formée par deux éléments A et B, A est la référence utilisée pour positionner la cible B. Pour une présentation composite, nous avons choisi de considérer que son premier élément est la référence et la présentation composite des éléments restants la cible. Par exemple, pour la présentation composite $\mathcal{P}_{abcd}$, $\mathcal{P}_a$ est la référence et $\mathcal{P}_{bcd}$ la cible (cf. figure 4.12). Nous présentons dans la suite les types de configurations spatiales et temporelles pouvant être décrités en OQLiST.

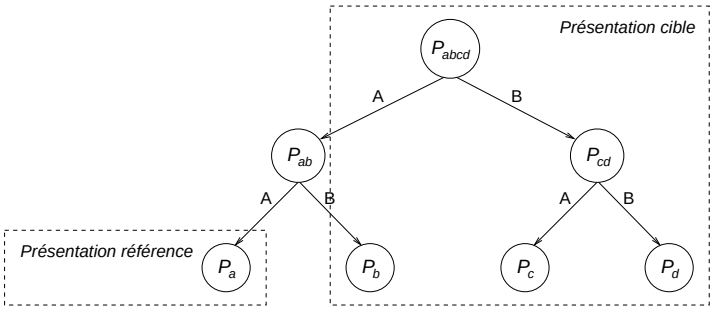


FIG. 4.12 – *Présentation référence et présentation cible*

4.3.1 Opérateurs spatiaux

Les opérateurs spatiaux combinent les relations topologiques et directionnelles (cf. section 3.3.3). Ils permettent de décrire l'ordonnancement spatial entre les éléments d'une présentation composite $\mathcal{PC}$ en spécifiant n'importe quelle configuration parmi les 169 configurations (cf. annexe B). Selon la forme et la sémantique de paramètres des opérateurs spatiaux, nous distinguons trois groupes d'opérateurs :

1. Combinaison d'une relation topologique avec une relation directionnelle drel $\in$ { north, south, east, west }. Par exemple présenter une image de Montmartre au sud et disjointe d'une image de Grenoble.

   Des précisions doivent être spécifiées pour préciser cette configuration. Par exemple de combien de pixels ces images doivent être disjointes. Doivent elles

être ajustées à droite, à gauche au centre? Ces informations sont spécifiées en combinant des opérateurs relatifs tels que beginning, centered, end, middle, meet avec des valeurs.

Le tableau 4.4 donne des expressions de configurations utilisant notre premier groupe d'opérateurs. Soient D, E des expressions de type réel avec D > 0 et E >= 0. Soit Q une expression de la forme : beginning + D, centered ± E, end − D et P de la forme : beginning − D, end + D, middle ± E et meet − D.

Expression
$\mathcal{PC}$ disjoint D, Q, drel
$\mathcal{PC}$ touch, Q, drel
$\mathcal{PC}$ overlap P, Q, drel
$\mathcal{PC}$ cover, Q, drel
$\mathcal{PC}$ covered by, Q, drel

TAB. 4.4 – *Premier groupe d'opérateurs spatiaux*

– $\mathcal{PC}$ disjoint D, Q, drel décrit une configuration où la présentation référence de $\mathcal{PC}$ est D pixels disjointe au nord, sud, est, ou ouest de la cible. D spécifie la distance entre les frontières les plus proches entre les deux éléments. L'expression Q spécifie que la cible peut être alignée E pixels par rapport au centre centered ± E, D pixels à gauche (beginning + D) ou à droite (end − D) de la référence.

  Par exemple présenter une image de Grenoble disjointe (50 pixels), au centre et en bas d'une image de Montmartre (cf. figure 4.13[1]) :

  ```
  Image( URL = "file://localhost/Montmartre.gif" ) compound
  Image( URL = "file://localhost/Grenoble.gif" )
  disjoint 50, centered, south
  ```

  La séparation entre le côté bas de Montmartre et le côté haut de Grenoble est de 50 pixels. Cette séparation plus la taille en y de Montmartre détermine le décalage en y de la présentation de Grenoble. Le décalage en x pour la présentation de Grenoble est calculé par l'opérateur centered (cf. section 3.3.3).

– $\mathcal{PC}$ touch, Q, drel décrit une configuration où la cible touche la référence au nord, sud, est ou ouest. L'expression Q spécifie l'alignement de la cible par

1. Nous montrons en gris son espace de présentation.

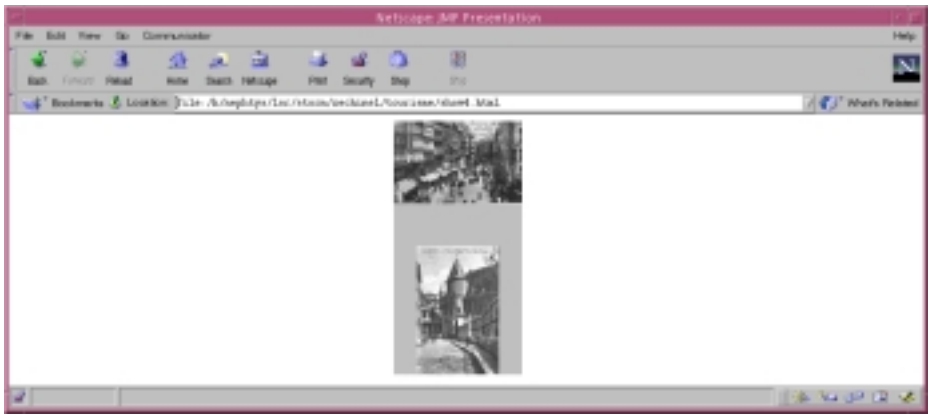


FIG. 4.13 – *Présentations disjointes de deux images*

rapport à la référence. Par exemple présenter une image de Grenoble qui touche une image de Montmartre au sud. L'image de Grenoble doit être alignée à gauche 10 pixels vers l'extérieur de Montmartre.

```
Image( URL = "file://localhost/Montmartre.gif" ) compound
Image( URL = "file://localhost/Grenoble.gif" )
touch, beginning - 10, south
```

– $\mathcal{PC}$ overlap P, Q, drel décrit une configuration où la présentation référence de $\mathcal{PC}$ chevauche la cible au nord, sud, est, ou ouest. L'alignement de la cible par rapport à la référence est spécifiée par les expressions P et Q. Par exemple présenter une image de Grenoble qui chevauche une image de Montmartre au sud. Grenoble est alignée 20 pixels en x et 20 pixels en y par rapport au centre de Montmartre.

```
Image( URL = "file://localhost/Montmartre.gif" ) compound
Image( URL = "file://localhost/Grenoble.gif" )
overlap, centered + 20, centered + 20, south
```

– $\mathcal{PC}$ cover, Q, drel décrit une configuration où la cible couvre la référence au nord, sud, est ou ouest et où Q précise l'alignement de la cible par rapport à la référence. Par exemple présenter l'image de Grenoble au sud de Montmartre. Grenoble doit couvrir Montmartre et elle doit être alignée à droite 5 pixels à l'intérieur de l'image de Montmartre.

```
Image( URL = "file://localhost/Montmartre.gif" ) compound
Image( URL = "file://localhost/Grenoble.gif" )
cover, end − 5, south
```

– $\mathcal{PC}$ covered by, Q, drel décrit une configuration où la référence est couverte par la cible au nord, sud, est ou ouest où Q précise l'alignement de la cible par rapport à la référence. Par exemple présenter l'image de Montmartre au sud de Grenoble. Montmartre doit couvrir Grenoble et elle doit être alignée à droite 5 pixels à l'intérieur de l'image de Grenoble.

Image(URL = "file://localhost/Montmartre.gif") compound
Image(URL = "file://localhost/Grenoble.gif")
coveredby, end − 5, south

2. Le deuxième groupe combine une relation topologique avec une relation directionnelle drel ∈ { northwest, southeast, northeast, southwest }. Le tableau 4.5 donne des expressions de configurations exprimées avec des décalages relatifs sur x et y respectivement spécifiés par les expressions P et Q.

Expression
$\mathcal{PC}$ disjoint, P, Q, drel
$\mathcal{PC}$ touch, P, Q, drel
$\mathcal{PC}$ overlap, P, Q, drel
$\mathcal{PC}$ cover, drel
$\mathcal{PC}$ covered by, drel

TAB. 4.5 – *Deuxième groupe d'opérateurs spatiaux*

– $\mathcal{PC}$ disjoint P, Q, drel décrit une configuration où la référence est disjointe de la cible au nord-est, sud-est, nord-ouest ou sud-ouest. La distance entre les deux est exprimée avec une position relative par rapport à un des côtés bas ou haut et à un des côtés latéraux de la référence. Deux cas de figure sont possibles :

(a) si la cible est décalée D pixels à droite de la référence, alors elle peut seulement être alignée E pixels en haut beginning − E, en bas meet ± E ou au milieu middle ± E de la référence. Notons que cet alignement doit vérifier que la cible ne touche jamais le milieu d'un des côtés de la référence, sinon la configuration décrirait une configuration du premier groupe. Si P = meet + D ⇒ Q ∈ { beginning − E, end + E, middle ± E, meet ± E }.

(b) si la cible est décalée D pixels en bas de la référence, alors elle peut seulement être alignée E pixels à gauche beginning − E, à droite meet ±

E ou au milieu middle ± E de la référence. Si Q = meet + D ⇒ P ∈ { beginning − E, end + E, middle ± E, meet ± E }.

- $\mathcal{PC}$ touch P, Q, drel décrit une configuration où la cible touche la référence au nord-est, sud-est, nord-ouest ou sud-ouest. La distance entre les deux est exprimée avec une position relative par rapport à un des côtés bas ou haut et à un des côtés latéraux de la référence. Deux cas de figure sont possibles :
  (a) la cible est décalée tout au long des côtés bas de la référence avec un alignement à droite, à gauche ou au centré. Si P = meet ⇒ Q ∈ { beginning − E, end + E, middle ± E, meet − E }.
  (b) la cible est décalée tout au long d'un des côtés latéraux de la référence avec un alignement à droite, à gauche ou au centré. Si Q = meet ⇒ P ∈ { beginning − E, end + E, middle ± E, meet − E }.
- $\mathcal{PC}$ overlap P, Q, drel décrit une configuration où la cible chevauche la référence au nord-est, sud-est, nord-ouest ou sud-ouest. La région chevauchée est décrite par deux positions relatives P et Q exprimées respectivement par rapport à un des côtés bas ou haut et à un des côtés latéraux de la référence. La cible peut donc être alignée à droite, à gauche ou au centre par rapport aux côtés latéraux et au ou bas de la référence. P et Q ∈ { beginning − D, end + D, middle ± D, meet − D }.
  Par exemple présenter une image de Grenoble au sud-est d'une image de Montmartre. Grenoble doit chevaucher Montmartre. Elle doit être alignée à 40 pixels du côté droit et à 40 pixels du côté bas de Montmartre (cf. figure 4.14[2]). Cette configuration est définit par l'expression OQLiST suivante :

```
Image( URL = "file://localhost/Montmartre.gif" ) at 110, 70 compound
Image( URL = "file://localhost/Grenoble.gif" )
overlap, meet − 40, meet − 40, southeast
```

Les relations cover et covered by combinées avec une relation directionnelle drel définissent une position précise. Chacune d'elles spécifie quatre configurations possibles qui positionnent la présentation cible dans l'un des quatre coins de la présentation référence. Le coin sélectionné correspond à la relation drel choisie.

2. Nous montrons en gris son espace de présentation.

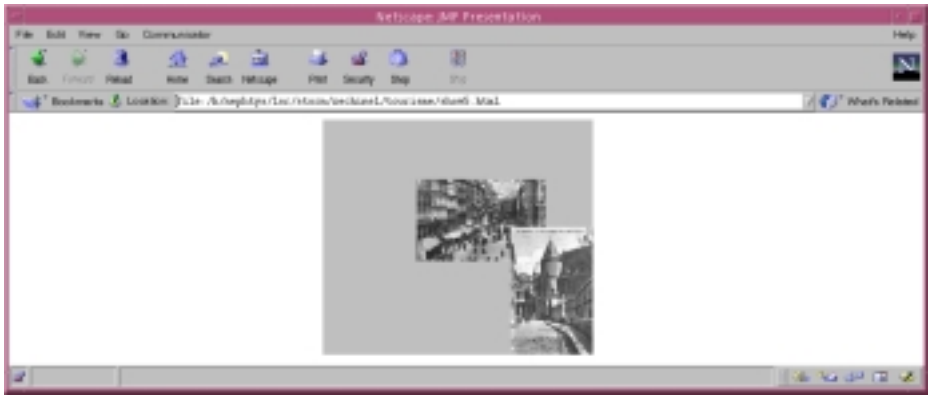


FIG. 4.14 – *Présentations superposées de deux images*

3. Le troisième groupe d'opérateurs permet de définir des configurations qui ne peuvent pas être décrites en utilisant une relation directionnelle (cf. tableau 4.6).

Expression
$\mathcal{PC}$ cover, P, Q
$\mathcal{PC}$ covered by, P, Q
$\mathcal{PC}$ inside, P, Q
$\mathcal{PC}$ contain, P, Q
$\mathcal{PC}$ spatial equal

TAB. 4.6 – *Troisième groupe d'opérateurs spatiaux*

- $\mathcal{PC}$ cover, P, Q la référence chevauche la cible. La région chevauchée est décrite par deux positions relatives exprimées respectivement par rapport à l'un des côtés latéraux et l'un des côtés bas ou haut de la référence. La cible est donc alignée E pixels à gauche ou en haut beginning + E, au centre centered ± E ou à droite ou en bas de la référence end − E. Précisons que cette configuration dénote des zones de la référence dans lesquelles la direction est ambiguë.
- $\mathcal{PC}$ covered by, P, Q cette configuration est l'inverse de cover.
- $\mathcal{PC}$ inside, P, Q la cible est contenu dans la référence. La position de la cible est précisée par deux positions relatives exprimées respectivement par rapport à l'un des côtés latéraux et l'un des côtés bas ou haut de la référence.
- $\mathcal{PC}$ contain, P, Q cette configuration est l'inverse de inside.

– $\mathcal{PC}$ spatial equal la cible est la référence ont la même taille et la référence se superpose à la cible.

L'exemple suivant illustre une configuration du troisième groupe. Considérons les présentations des deux vidéos. SolarSystem montre la création du système solaire et Sun fait une présentation du soleil.

```
define SolarSystem as Video( URL = "file://localhost/SolarSystem.mov" );
define Sun as Video( URL = "file://localhost/Sun.mov" );
```

Nous spécifions ensuite une présentation de ces deux vidéos où SolarSystem sert de cadre à la présentation de la vidéo Sun. Pour cela, nous augmentons la largeur de SolarSystem de 40%, puis nous les composons avec une relation covered by. Les deux présentations sont centrées l'une par rapport à l'autre en x et en y. Finalement, nous spécifions que la taille de la présentation est de 250×250 et qu'elle est entourée par un cadre noir de 10×10 pixels (cf. figure 4.15) :

```
atomic( SolarSystem with Sun.getWidth,
                         Sun.getHeight + 0.4 * Sun.getHeight
        compound Sun
        covered by, centered, centered with 250, 250 )
border 10, 10 background black
```

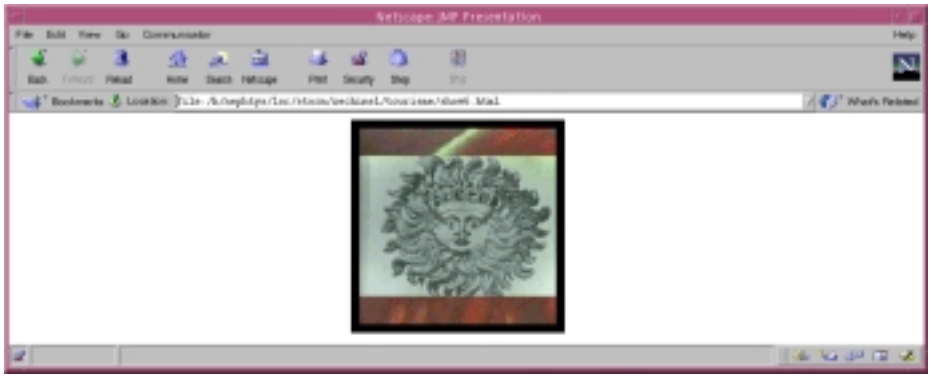


FIG. 4.15 – *Présentations superposées de deux vidéos*

4.3.2 Opérateurs temporels

Les tableaux 4.7 et 4.8 présentent les relations pouvant être utilisées pour exprimer la synchronisation de présentations composites en OQLiST. Par exemple synchroniser la présentation d'une image du drapeau français avec celle d'un son qui joue la Marseillaise. Selon le type de synchronisation il peut être nécessaire ou pas d'avoir une précision.

Le tableau 4.7 présente les relations qui ont besoin de précisions absolues pour spécifier une synchronisation. Soient D un réel et R une des relations before, overlap, during, bi, oi, di. $\mathcal{PC}$ synchronize R, D caractérise une configuration où la référence et la cible de la présentation composite $\mathcal{PC}$ sont synchronisées D unités de temps avec une relation R.

Relation	Inverse
before, D	bi, D
overlap, D	oi, D
during, D	di, D

TAB. 4.7 – *Relations temporelles caractérisées*

Par exemple considérons une présentation de l'image du drapeau français (FrenchFlag) et une présentation de la Marseillaise. Nous pouvons définir une configuration temporelle où les deux présentations sont synchronisées avec une relation during pendant 2 unités de temps :

```
Image( URL = "file://localhost/FrenchFlag.gif" ) compound
Audio( URL = "file://localhost/Marseillaise.au" ) synchronize during, 2
```

Notons que les deux présentations ont des durées différentes, l'image a même une durée indéterminée. La valeur D permet de borner la période de temps pendant laquelle la cible et jouée.

Le tableau 4.8 présente des relations qui spécifient une position de manière précise. Soit R une des relations meet, start, finish, equal, mi, si, fi. L'expression $\mathcal{PC}$ synchronize D décrit une configuration où la présentation de la référence :

- commence (start, si) ou finit (finish, fi) en même temps que la cible ;
- suit (meet) ou précède (mi) celle de la cible ;
- commence et fini en même temps que la cible equal.

Relation	Inverse
meet	mi
start	si
finish	fi
equal	

TAB. 4.8 – *Relations temporelles non caractérisées*

Par exemple présenter l'image de Montmartre à la position (110, 70) pendant 15 secondes. Une image de Grenoble chevauche Montmartre au sud-est alignée 40 pixels à droite et 40 pixels en bas. Grenoble est présentée pendant 10 secondes et sa présentation finit en même temps que celle de Montmartre. Nous utilisons la relation fi parce que la durée de Montmartre est plus grande de celle de Grenoble.

Image(URL = "file://localhost/Montmartre.gif") at 110, 70 during 15 compound
Image(URL = "file://localhost/Grenoble.gif") during 10
overlap, meet − 40, meet − 40, southeast synchronize fi

4.3.3 Ordre et profondeur des présentations

OQLiST fournit trois opérateurs qui permettent de manipuler l'ordre des éléments dans une présentation composite : inverse, move et swap. Soient $\mathcal{PC}$ une expression qui dénote une présentation composite et S, D deux expressions de type entier :

– $\mathcal{PC}$ inverse construit une nouvelle présentation où l'ordre des éléments de $\mathcal{PC}$ est inversé. Par exemple étant donnée une présentation $\mathcal{P}_{MG}$ où une image de Grenoble superpose une image de Montmartre, $\mathcal{P}_{MG}$ inverse donne une présentation $\mathcal{P}_{GM}$ où Montmartre superpose Grenoble (cf. figure 4.16).

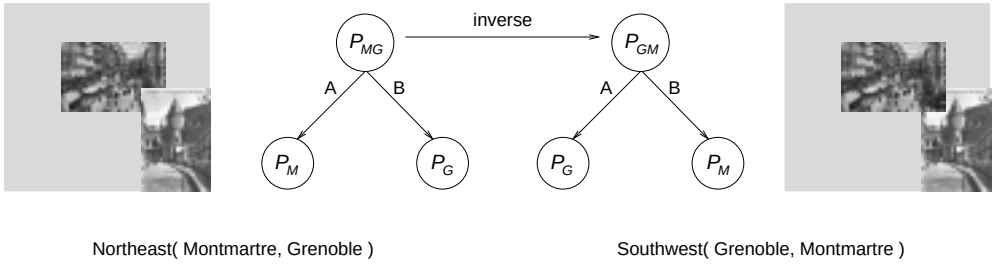


FIG. 4.16 – *Présentation inverse*

– $\mathcal{PC}$ move S to D construit une présentation où l'élément de $\mathcal{PC}$ à la position S est placé à la position D.

– $\mathcal{PC}$ swap S, D construit une présentation où deux éléments situés respectivement à la position S et D échangent leur position.

4.4 Interrogation et construction

4.4.1 Taxonomie des requêtes spatio-temporelles

La figure 4.17 présente une taxonomie de types de requêtes pouvant être exprimées en OQLiST. Nous identifions 15 types de requêtes : neuf résultent d'utiliser des positions absolues et/ou relatives pour exprimer une requête (cf. figure 4.17 à gauche) ; six résultent de combiner des positions absolues et (absolues)relatives –spatiales et temporelles– (cf. figure 4.17 à droite).

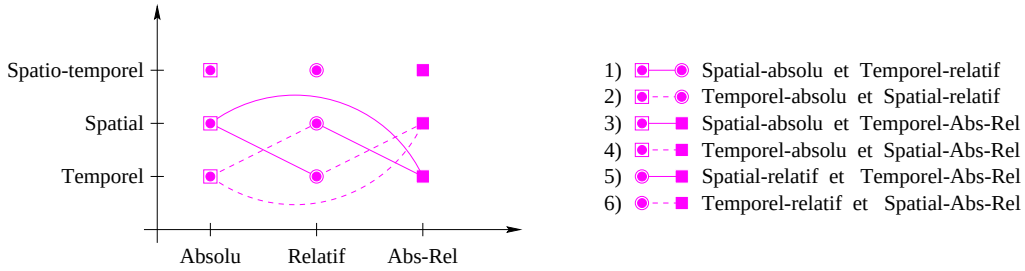


FIG. 4.17 – *Types de requêtes*

Requête absolue Elle est exprimée en termes de positions absolues spécifiées par rapport à un référentiel. Considérons la présentation `FrenchCitiesViews` où des images de villes françaises sont affichées à gauche de l'écran, i.e., les positions des images sur l'axe x < 300. En particulier, une image de la ville de Grenoble est présentée dans la partie basse de l'écran, i.e., avec y > 700. Nous pouvons par exemple définir une présentation constituée seulement par des objets qui sont présentés dans la partie basse de l'écran pendant les 200 premiers secondes de la présentation `FrenchCitiesViews` :

```
select e from e in FrenchCitiesViews.elements
where e.offset_y >= 700 and not e.delay >= 200
```

Requête relative Elle est exprimée en termes de positions relatives (relations spatiales et/ou temporelles) éventuellement combinées avec des informations quantita-

tives. La position relative d'une présentation A dans une présentation est exprimée par rapport à une présentation B. Par exemple quels sont les éléments de la présentation `FrenchCitiesViews` qui se chevauchent et démarrent 30 seconds avant la présentation de l'image de Grenoble. Dans ce cas, la présentation de l'image de Grenoble est prise comme référence. Voici la requête OQLiST correspondante :

```
select  r.A.content from r in ( FrenchCitiesViews.relations union
                                select r.inverse from r in FrenchCitiesViews.relations )
where  r.Spatial_Overlap and r.Before and
       r.B.delay >= 30 and ((Document) r.B.content).getName = "Grenoble"
```

Requête absolue-relative Elle est exprimée en utilisant des positions absolues et relatives. Considérons une configuration où des images de villes françaises sont présentées du côté gauche de l'écran et où une image de Grenoble est présentée dans une autre région de l'écran. Nous pouvons définir une présentation contenant les images qui apparaissent du côté gauche de l'écran au sud-ouest de l'image de Grenoble. Cette description utilise des informations absolues (i.e., du côté gauche de l'écran correspondant à une position $x < 300$) et relatives (i.e., au sud-ouest de Grenoble).

Des restrictions portant sur des attributs absolus et relatifs peuvent être ajoutées. La requête suivante montre comment combiner ces restrictions avec des restrictions spatiales et temporelles :

```
select  r.A from r in ( FrenchCitiesViews.relations union
                        select r.inverse from r in FrenchCitiesViews.relations )
where  r.Northwest and r.A.offset_x < 300 and
       r.Before and r.A.delay > 70 and
       ((Document) r.B.content).getName = "Grenoble"
```

Elle récupère les éléments de la présentation FrenchCitiesViews dont la présentation démarre dans les premiers 70 secondes mais avant celle de Grenoble. Les éléments récupérés sont présentés du côté gauche de l'écran et sud-ouest de la présentation de Grenoble.

Requête hybride Elle est exprimée en combinant des positions absolues et (absolues)relatives avec des relations spatiales et temporelles. Par exemple retrouver les éléments de la présentation FrenchCitiesViews qui s'affichent en bas de l'écran et 30

secondes après la présentation de Grenoble :

```
select  e.content
from    e in FrenchCitiesViews.elements,
        r in ( FrenchCitiesViews.relations union select r.inverse
                                                from  r in FrenchCitiesViews.relations )
where   e.offset_y >= 700 and e.content = r.A.content and
        r.Before and r.A.delay >= 30 and
        ((Document) r.B.content).getName = "Grenoble"
```

Dans cet exemple, nous combinons des informations spatiales absolues (en bas de l'écran) avec des informations temporelles relatives (les objets présentés 30 secondes après la présentation de Grenoble).

4.4.2 Présentations comme résultats de requêtes

En OQLiST, une présentation est construite en spécifiant les objets qui la composent (i.e., son contenu) et la configuration spatio-temporelle de ses objets. Les objets peuvent être spécifiés en extension et/ou en intention. La spécification en intention des objets composants est exprimée par des requêtes qui récupèrent partiellement ou totalement des présentations existantes. En supposant que les objets de la présentation `FrenchCitiesViews` ont un attribut indiquant des mots qui décrivent leur contenu, nous pouvons récupérer et créer une présentation de l'ensemble d'objets dont les objets font référence à la région Rhône - Alpes :

```
select e.content from e in FrenchCitiesViews.elements
where "Rhône - Alpes" in ((Document) e.content).getKeywords
```

La spécification en extension est faite en donnant par exemple l'adresse URL de chaque objet qui sera contenu dans la présentation.

La configuration spatio-temporelle d'une présentation peut être spécifiée de trois façons : spécification explicite des attributs spatiaux et temporels, utilisation de formats prédéfinis, réutilisation d'attributs des présentations existantes.

Spécification explicite Une configuration spatiale et temporelle est associée explicitement au résultat d'une requête. Dans l'exemple, précédant, nous pouvons associer une configuration aux images des villes qui apparaissent au sud de la carte. Par

exemple présenter chaque image pendant 5 secondes à la position (100, 100). Ceci est exprimé avec OQLiST de la façon suivante :

```
select  r.A at 100, 100 during 5
from    r in ( FrenchCitiesViews.relations union
                 select r.inverse from r in FrenchCitiesViews.relations )
where  r.South and ((Document) r.B.content).getName = "FrenchMap" order by *
arrange all meet on t
```

Utilisation de opérateurs sur les collections d'objets (table) L'expression suivante décrit la présentation d'un tableau d'images à trois colonnes avec un arrière plan rouge. Le tableau a une taille de 550×440 pixels ce qui implique que la taille des présentations des images du contenu devra être adaptée. Toutes les lignes du tableau sont présentées en même temps. Les images du tableau sont récupérées de la présentation `FrenchCitiesViews` :

```
select e from e in FrenchCitiesViews.elements
table 3 with 550, 440 background red
arrange all beginning on t
```

Réutilisation de configurations Des relations spatiales et temporelles définies entre des objets récupérées en intention peuvent être réutilisées pour spécifier la configuration d'une présentation. Il est possible de spécifier des sous-présentations prises dans un intervalle de temps dans une présentation. Par exemple si la présentation `FrenchCitiesViews` dure 1 minute, une nouvelle présentation peut être obtenue en définissant une sous-présentation composée des éléments de `FrenchCitiesViews` visibles entre les secondes 10 et 20. Dans ce cas, la présentation garde la configuration des objets dans la présentation initiale.

4.4.3 Description du contenu des objets

OQLiST définit deux opérateurs pour manipuler les descriptions d'objets : tilde (~) et descrip. Soient $\mathcal{P}$ une expression OQLiST définissant une présentation et $\mathcal{O}$ un objet :

- $\mathcal{O}$ ~$\mathcal{P}$ crée une présentation qui associe $\mathcal{P}$ à l'objet $\mathcal{O}$. L'expression $\mathcal{P}$ peut décrire le contenu de $\mathcal{O}$ en spécifiant des régions (dans l'espace) et des segments

(dans le temps) sur cet objet. Par exemple l'expression OQLiST ci-dessous décrit le contenu d'une image de la carte de France. Elle coupe deux régions de l'image nommées "Bretagne' et "Aquitaine" pour en composer une présentation et l'associer à la présentation FrenchMap.

```
define FrenchMap as Image( URL = "file://localhost/FrenchMap.gif" );
FrenchMap ~
        FrenchMap.crop( 8, 85, 93, 55 ).setName( "Bretagne" ) at 8, 85
        compound
        FrenchMap.crop( 74, 206, 87, 110 ).setName( "Aquitaine" ) at 80, 55
```

Précisons que cette opération lève une exception si $\mathcal{O}$ est un objet de type présentation. Des requêtes peuvent éventuellement être posées sur la présentation FrenchMap et sur sa présentation associée (sa description).

- descrip($\mathcal{O}$) permet de récupérer la description associée à l'objet $\mathcal{O}$. Par exemple descrip(FrenchMap) récupère la présentation qui correspond à la description de FrenchMap :

```
FrenchMap.crop( 8, 85, 93, 55 ).setName( "Bretagne" ) at 8, 85 compound
FrenchMap.crop( 74, 206, 87, 110 ).setName( "Aquitaine" ) at 80, 55
```

4.5 Conclusion

Le langage OQLiST est une extension OQL qui fournit un ensemble d'opérateurs pour manipuler (i.e., sélectionner, modifier et spécifier) les attributs et les relations spatio-temporelles des présentations. Une requête est exprimée en termes de positions absolues et relatives spatiales, temporelles et spatio-temporelles. Les opérateurs fournis par OQLiST permettent de :

- construire des présentations (atomiques et composées) d'objets.
- interroger les attributs spatio-temporels des présentations. Par exemple donnez moi toutes les villes qui sont au sud de Paris ?
- spécifier les caractéristiques spatio-temporelles des présentations intentionnelles résultant de requêtes. Elles peuvent être construites en modifiant (1) les caractéristiques des présentations composantes (positions et tailles) et/ou (2) les re-

lations spatio-temporelles des objets contenus. Par exemple présenter en même temps un ensemble d'images dans un tableau.

Notre langage hérite du langage OQL la possibilité de manipulation d'objets complexes ce qui permet de définir et de naviguer dans des présentations composées récursivement par d'autres présentations. Nous avons également profité de la notion d'objet complexe pour définir des présentations d'objets qui contiennent en plus des descriptions de ses objets. Cette stratégie permet d'interroger et de construire des présentations à partir des objets et des morceaux d'objets.

Concernant l'expression de relations spatiales et temporelles définies par OQLiST, notre approche a consisté à être les plus exhaustifs possible en permettant d'exprimer un espace de relations d'intervalles "maximum" dans un espace 1D, 2D et 3D : 13 sur le temps, 13^2 sur l'espace et 13^3 sur l'espace-temps. De plus, nous avons intégré la notion d'ordre dans la présentation d'objets, ce qui permet d'avoir une notion de profondeur. La profondeur permet donc d'exprimer le fait qu'un objet est devant ou derrière un autre.

Finalement, nous pensons que la taxonomie de requêtes devrait pouvoir aider à spécifier des techniques d'optimisation de requêtes OQLiST. Ce travail est d'autant plus intéressant considérant la taille importante de données manipulées.

Chapitre 5

Infrastructure pour la spécification de gestionnaires de présentations

Dans ce chapitre, nous présentons JAGUAR, une infrastructure pour la spécification de gestionnaires de présentations. Un gestionnaire assure la définition, le stockage, l'interrogation et la mise en forme de présentations multimédias spatiale et/ou temporelles. Les gestionnaires spécifiés servent de médiateurs entre des sources de données hétérogènes réparties et des applications multimédias. Un gestionnaire constitue donc une couche d'intégration entre des données représentées par des modèles différents et des applications utilisant des synchronisations et des formattages divers.

La section 5.1 présente notre approche pour gérer de présentations au travers de gestionnaires. Les sections 5.3 et 5.4 décrivent les fonctions principales d'un gestionnaire : la transformation et mise en forme des spécifications et la construction de présentations. La section 5.5 introduit l'architecture de JAGUAR et décrit également la mise en place et la génération d'un gestionnaire. Enfin, la section 5.6 conclut ce chapitre.

5.1 Intégration d'objets dans des présentations

Considérons une application qui permet de spécifier des présentations de sites touristiques que l'on peut visiter en France selon des thèmes particuliers : sites historiques, stations de ski, promenades écologiques, sites balnéaires, promenades artistiques, etc. Des applications Web pouvant tourner sur des plates-formes logicielles et matérielles différentes (par ex. PDA, PC, stations de travail, navigateurs Web) sont

mises à disposition pour visualiser les informations des sites touristiques. Les utilisateurs de ces applications récupèrent des présentations selon la région et le type de visites qu'ils veulent réaliser. Par exemple *les stations d'hiver à la région Rhône Alpes et les activités que l'on peut réaliser.*

Pour pouvoir mettre en œuvre une application, il faut qu'elle soit implantée sur une plate-forme (librairie de classes) permettant de spécifier le traitement des spécifications dans un langage particulier (par ex. un langage graphique) et l'exécution de la présentation des résultats (cf. figure 5.1). Selon le type d'environnement matériel et logiciel pour lequel l'application est destinée, le nombre maximal d'images à afficher et les types d'opérations sur les présentations (stocker, exécuter, adapter) sont fixés.

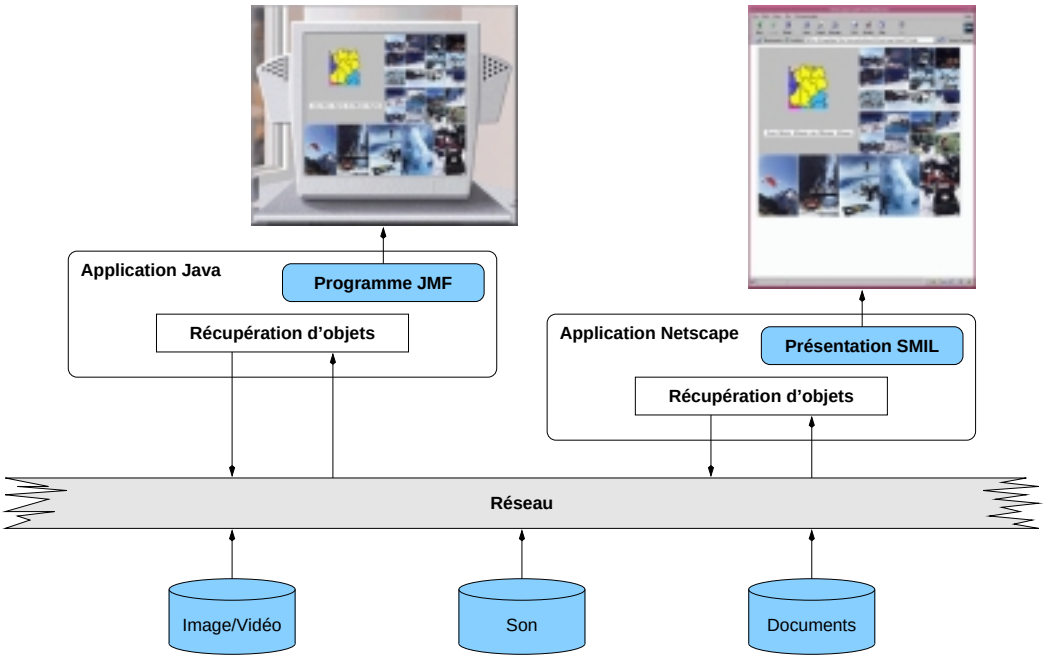


FIG. 5.1 – *Applications multimédias*

Les applications doivent (1) récupérer les objets directement à partir de leurs sources ou interagir avec un niveau de médiation qui les récupère (par ex. un moteur de recherche) ; (2) interpréter correctement les spécifications de présentations (par ex. organisation spatio-temporelle et transformation d'objets). Les sources sont prédéfinies par le constructeur de l'application. Il faut ensuite intégrer les données (décrits dans des formats différents) récupérées dans une représentation commune et construire une présentation selon le modèle de synchronisation de la plate-forme d'exécution choisie par l'application cible. Les objets doivent être récupérés en res-

pectant la synchronisation temporelle (i.e., QoS). Autrement, les applications doivent implanter des stratégies pour compenser les problèmes de délai (par ex. caching).

5.1.1 Gestionnaire de présentations

Un gestionnaire de présentations est une infrastructure logicielle intermédiaire entre les applications et les données (cf. figure 5.2). Chaque application interagit avec un gestionnaire de présentations à travers un adaptateur qui lui permet de spécifier des présentations de manière transparente (i.e., sans considérer le format des données et les sources à accéder). Par exemple *présenter sous forme de tableau des images des stations d'hiver et les activités que l'on peut réaliser dans la région Rhône Alpes.*

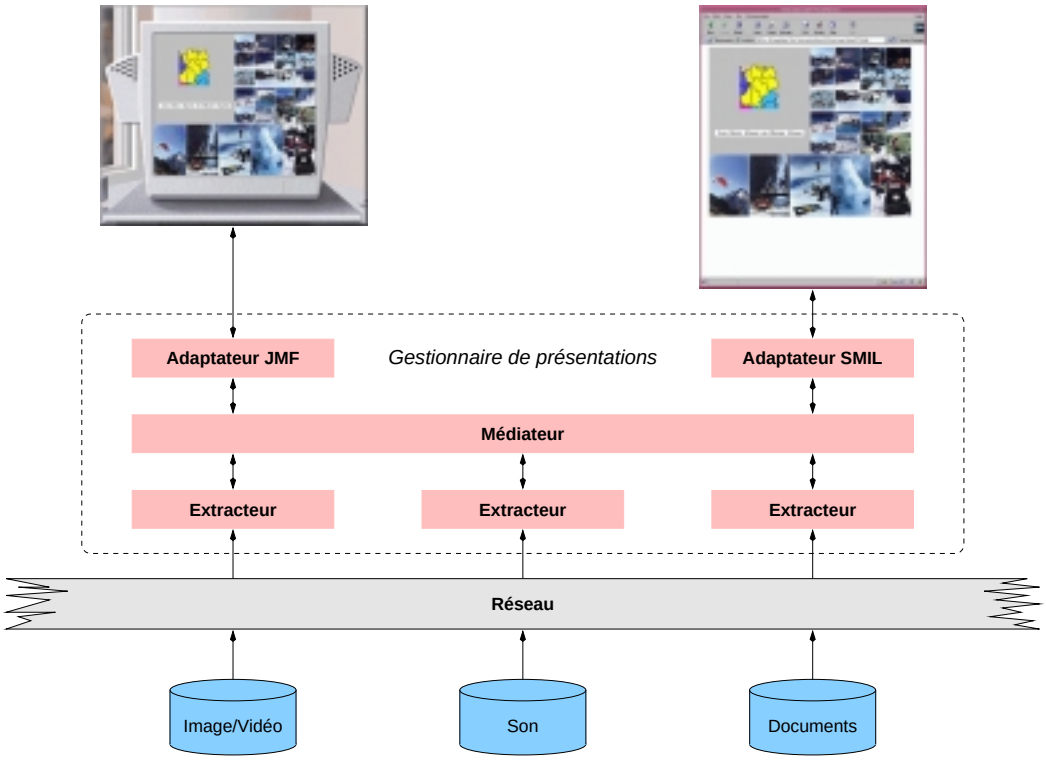


FIG. 5.2 – *Intégration d'applications et de sources*

Un gestionnaire fournit des interfaces d'interaction avec les applications et avec les sources d'information (localisation, types de format). Il est basé sur un modèle spatio-temporel pivot qui permet l'intégration et l'interrogation des données multimédias (images GIF, documents XML, objets bases de données) récupérées à partir des sources distribuées (cf. chapitre 3).

Les présentations construites sont mises en forme sous des formats différents selon les applications cibles. Les formats dépendent des modèles de synchronisation des plates-formes d'exécution utilisées par les applications. Le gestionnaire sert éventuellement comme support pour l'exécution des présentations. Il prend en compte aussi les délais de récupération des données et il couple les stratégies propres aux applications multimédias avec des techniques de préchargement et gestion de cache.

Dans notre exemple touristique, le gestionnaire reçoit les spécifications de présentations sur les sites à visiter à travers des adaptateurs qui interagissent avec chaque application (cf. figure 5.2). Le médiateur récupère les données requises auprès des sources à travers des extracteurs spécialisés. Il intègre les résultats obtenus par les extracteurs dans une présentation qui est ensuite envoyée aux adaptateurs. Chaque adaptateur met en forme la présentation résultante selon le format requis par l'application (par ex. les formats JMF et SMIL).

5.1.2 Fonctionnement

La figure 5.3 présente l'architecture générale d'un gestionnaire de présentations qui réalise trois fonctions principales : transformation, construction et génération (mise en forme).

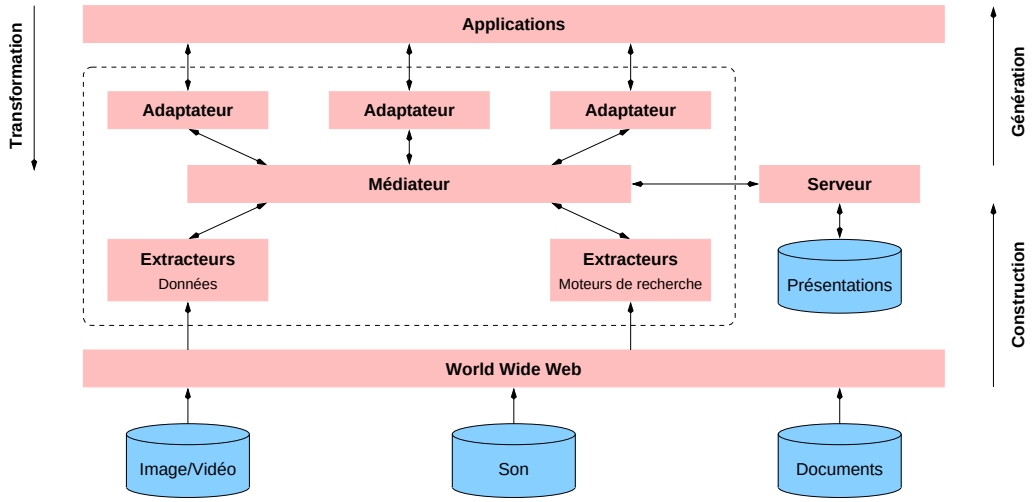


FIG. 5.3 – *Architecture générale d'un gestionnaire de présentations*

1. **Transformation**

   Cette opération est exécutée entre les applications et le gestionnaire de présentations par l'intermédiaire des adaptateurs. Un adaptateur est spécialisé pour transformer la spécification de l'application en une expression OQLiST que le médiateur comprend et doit évaluer. Cette transformation est réalisée par des règles de transformation spécifiées par le programmeur de l'application et exécutées par l'adaptateur.

2. **Construction**

   Le médiateur évalue l'expression de la spécification transformée. Dans cette phase, il interagit avec un système de persistance (i.e., un SGBD, le serveur de présentations) pour déterminer si la présentation ou une partie d'elle a déjà été créée et stockée. Lorsque c'est le cas, seules les données manquantes sont récupérées à partir des sources. Le médiateur reçoit les résultats du serveur de présentations et des extracteurs et il les combine dans une présentation en accord avec la spécification initiale (par ex. sous forme de tableau). Chaque extracteur renvoie les images au médiateur qui transforme les données vers le modèle de données pivot en utilisant leurs caractéristiques spatiales (leurs tailles).

3. **Génération**

   Le médiateur donne la présentation résultante à un adaptateur qui la met en forme sous un format compréhensible pour l'application cliente (par ex. programme SMIL). Cette mise en forme est guidée par des règles de génération gestionnaire-application spécifiées pour chaque plate-forme cible et implantées par un adaptateur.

5.2 Données d'un gestionnaire

Un gestionnaire de présentations gère (définit, construit, stock, met en forme) les présentations des types de données utilisés par un ensemble d'applications. Il est configuré pour gérer la présentation de ces types. Pour cela, il implante un modèle de données spatio-temporel qu'il couple avec un schéma de données. Il implante également des règles d'interaction avec les applications et les extracteurs des sources qu'il utilise pour récupérer des données.

5.2.1 Modèle spatio-temporel et schéma de données d'une application

Un gestionnaire de présentations implante notre modèle spatio-temporel (cf. chapitre 3). Le modèle du gestionnaire fournit quatre classes principales : Object, Presentation, Atomic et Compound. Tout objet a une présentation associée. Le modèle caractérise les présentations des objets à travers la classe générique Presentation qui peut être atomique Atomic ou Compound. Les présentations peuvent être spatiales, temporelles et spatio-temporelles.

Un schéma de données décrit les types de base et les opérations associées à chaque type. Ce schéma est utilisé par les applications et le gestionnaire. Object est une classe générique qui est spécialisée pour décrire les types utilisés par un ensemble d'applications.

Dans notre exemple, nous pouvons considérer que les applications touristiques n'affichent que des données du type texte et image. La classe Object est donc spécialisée en deux classes : Text et Image (cf. figure 5.4). Nous pouvons aussi spécifier que les présentations de la classe Image fournissent une méthode découper (crop) permettant de sélectionner une sous-région. Nous montrons dans la figure 5.4 que des méthodes sont associées à chaque type de donnée. Elles implantent le calcul de caractéristiques spatiales (largeur, longueur) et/ou temporelles (durée).

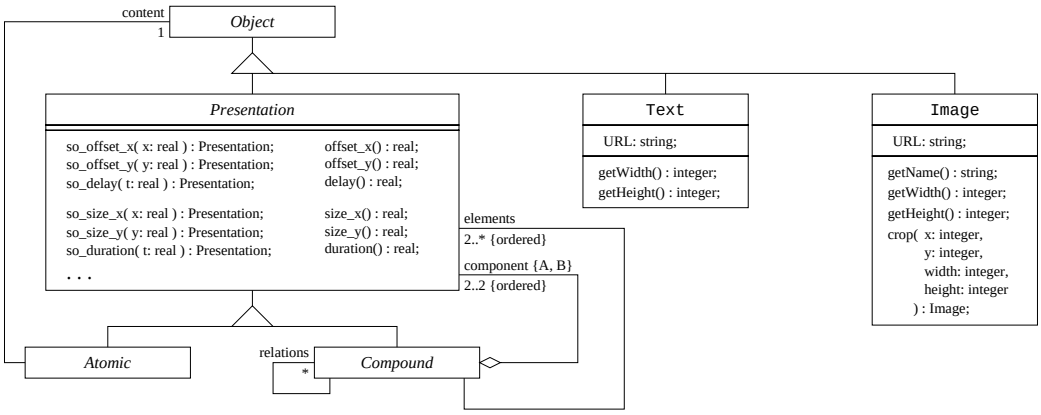


FIG. 5.4 – *Schéma de données d'une application*

5.2.2 Contexte de présentation

Interaction avec les sources L'interaction d'un gestionnaire de présentations et des extracteurs est décrite par des règles de correspondance. Une règle établissent la correspondance entre un type de données et un objet du modèle du gestionnaire. Les règles expriment aussi la localisation de la source et le format d'interrogation à utiliser pour récupérer les données d'un type spécifique. Par exemple :

- lorsque la source est un moteur de recherche, les requêtes sont exprimées par des conjonctions et disjonctions de mots clés.

- lorsque la source est une base de données, il faut exprimer les requêtes en utilisant sont langage d'interrogation.

Présentations par défaut Un contexte de présentation décrit comment présenter chaque type du schéma de données des applications en lui associant un type de présentation du modèle spatio-temporel du gestionnaire. Les types d'un schéma sont alors associés à des présentations par défaut (association entre un Object et une Presentation).

Dans notre exemple d'applications touristiques, nous pouvons associer les types du schéma (Image et Text) à des présentations de type Spatiale 2D et Temporelle. Un tel type de présentation décrit les attributs spatiaux et temporels ainsi que des opérations pouvant être exécutées pour présenter un objet. Par exemple le type Image est associé à une présentation spatio-temporelle qui affiche une image dans la position (0, 0) avec une durée indéterminée (puisque les images n'ont pas de caractéristiques temporelles inhérentes). La taille de la présentation correspond à celle de l'image brute (cf. figure 5.5).

FIG. 5.5 – *Présentation par défaut d'une image*

5.2.3 Interaction avec les applications

L'interaction d'un gestionnaire de présentations avec des applications est décrite par un ensemble de règles. Pour chaque application cliente, un gestionnaire implante :

1. Règles de transformation entre le langage de spécification de présentations utilisé par l'application et le langage du médiateur (OQLiST). Ce sont des règles décrivant la correspondance entre les types de base, leurs opérations associées et les relations spatiales et temporelles entre les deux langages.

2. Règles de génération entre le modèle spatio-temporel utilisé pour représenter les présentations résultantes et le format utilisé par l'application pour exécuter les présentations. Ces règles décrivent la correspondance entre une présentation construite dans le modèle de données du gestionnaire et celui pour synchroniser l'application cible. Il s'agit de spécifier l'interaction avec un planificateur pour que le gestionnaire génère des exécutables compréhensibles par des exécuteurs de présentations.

5.2.4 Utilisation d'un gestionnaire

Le gestionnaire de présentations est implanté dans une architecture client-serveur où plusieurs applications clientes communiquent avec le serveur, le gestionnaire de présentations (cf. figure 5.6).

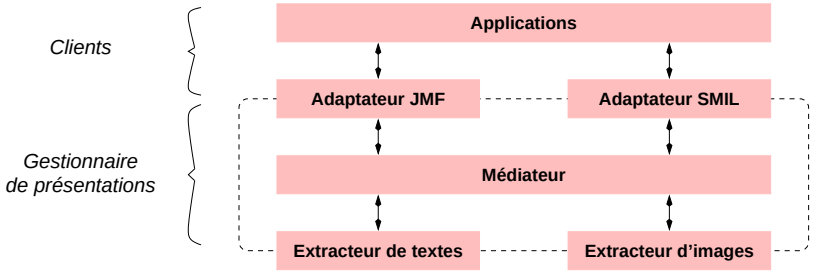


FIG. 5.6 – *Utilisation d'un gestionnaire*

Initialisation Pour initialiser un système utilisant un gestionnaire, le serveur est lancé, il prend contacte avec son serveur de présentations et avec les extracteurs des sources avec lesquelles il interagit. Ensuite, il reste en attente de requêtes provenant des applications.

Toute application ouvre une communication avec le serveur d'un gestionnaire en envoyant son contexte de présentation. L'application dialogue avec le gestionnaire à travers un adaptateur pour évaluer des spécifications de présentations et obtenir des présentations exécutables.

Changement de contexte Les applications communiquent aussi avec le médiateur pour changer leur contexte de présentation. Ceci implique l'utilisation des nouveaux constructeurs d'objets.

Changement d'adaptateur Cette facilité permet à une même application de faire exécuter des présentations en utilisant des plates-formes différentes. Par exemple si une application souhaite n'utiliser que des images et des sons exécutés par SOJA, elle peut utiliser une plate-forme SMIL. Par contre, si certaines de ses présentations contiennent des données vidéo, elle peut alors passer à un contexte *Java Média Framework*.

5.3 Transformation et génération de présentations

La transformation et la génération sont les processus par lesquels une présentation est traitée et mise en forme en accord avec une spécification provenant d'une application. Ces processus sont réalisés grâce à un adaptateur. Un adaptateur se présente comme une librairie de classes qui implante au moins un contexte de présentation.

5.3.1 Transformation

La transformation a comme objectif de passer d'une spécification de présentation *ad hoc* vers une spécification OQLiST compréhensible pour le médiateur (cf. figure 5.7). Chaque type de donnée utilisé dans la spécification est transformé vers un type du modèle spatio-temporel. Ensuite, chaque position absolue et/ou relative (relation spatiale et/ou temporelle) est transformée par une expression OQLiST. Ces transformations se réalisent grâce aux règles implantées par l'adaptateur.

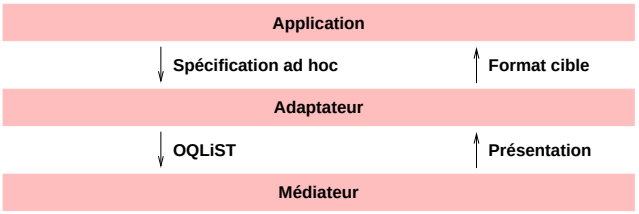


FIG. 5.7 – *Adaptateur*

Par exemple prenons une spécification qui décrit la présentation d'une image de Montmartre composée avec celle de Grenoble (cf. figure 5.8). L'image de Montmartre est affichée à la position (110, 70) avec une durée de 15 secondes. La présentation de l'image de Grenoble est affichée à la position (250, 150) pendant 10 secondes et elle démarre 5 secondes après celle de Montmartre.

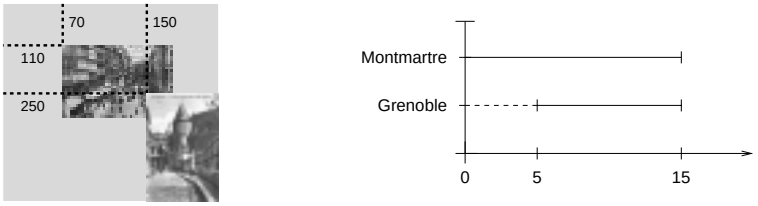


FIG. 5.8 – *Spécification d'une présentation*

Les références aux images (i.e., Montmartre et Grenoble) sont transformées sous la forme de constructeurs d'objets OQLiST de type Image. Cette présentation est transformée comme suit :

Image(URL = "file://localhost/Montmartre.gif") at 110, 70 during 15 compound
Image(URL = "file://localhost/Grenoble.gif") at 250, 150 at 5 during 10

5.3.2 Génération de présentations

La phase de génération des résultats implique la génération d'une présentation construite par le médiateur (représentée dans le modèle spatio-temporel du gestionnaire) vers la représentation d'une application cible. Un adaptateur reçoit la présentation sous forme d'un arbre où les nœuds correspondent aux relations et les feuilles aux objets de type présentation atomique (Atomic).

L'arbre est ensuite traité grâce aux règles de génération implantées par l'adaptateur. Les règles indiquent la correspondance entre le modèle du gestionnaire et le modèle cible. De cette manière, un adaptateur implante :

- Pour chaque type d'objet une règle qui définit la structure sous laquelle l'objet doit être transformé.

- Pour chaque opération associée à un type une règle qui définit quel est son correspondant dans le modèle cible. Si cette opération n'est pas supportée par le modèle cible, l'adaptateur doit exécuter l'opération en question et retourner l'objet résultant.

- Des règles qui spécifient la correspondance entre chaque relation (i.e., spatiales, temporelles, de profondeur, hyperlien) de présentations et sa présentation dans le modèle cible. Par exemple pour les modèles ne supportant pas des relations spatiales et/ou temporelles, les règles transforment les positions relatives à de positions absolues. Précisons que le calcul des positions doit tenir compte des caractéristiques des référentiels par rapport auxquelles les présentations ont été définies. Par exemple si le modèle cible a un référentiel discret, alors les positions spatiales et temporelles doivent être discrétisées.

- Un planificateur qui spécifie la synchronisation des présentations dans le modèle cible.

La figure 5.9 illustre l'arbre d'une présentation qui affiche une image de Montmartre à la position (110, 70) avec une image de Grenoble au sud-est (140 pixels et 80 pixels à partir du coin gauche supérieur de la présentation de Montmartre). Considérons que la plate-forme cible est basée sur SMIL. Dans cette plate-forme, les positions spatiales ne peuvent pas être exprimées en termes de relations. La relation sud-est est donc exprimée par des positions absolues.

Pour notre exemple, considérons les règles de génération suivantes :

- R1 : le type Image est transformé au type <img /> dans le format cible. Par exemple .

- R2 : le type Text est transformé au type <text /> dans le format cible.

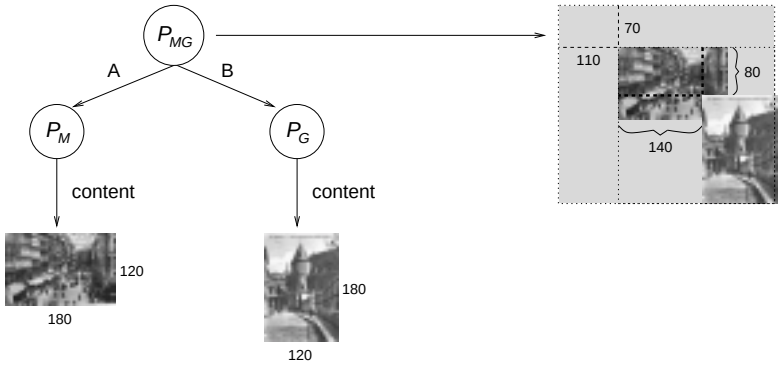


FIG. 5.9 – *Arbre d'une présentation*

– R3 : chaque présentation feuille dans un arbre de présentation (cf. figure 5.9) définit une présentation absolue p avec des caractéristiques spatiales offset_x, offset_y, size_x, size_y. p est transformée vers une région du format cible :

```
<region id=" + newRegion() + "
        left=" + (int) p.offset_x() + "
        top=" + (int) p.offset_y() + "
        width=" + (int) p.size_x() + "
        height=" + (int) p.size_y() + " />
```

Le région possède les mêmes caractéristiques spatiales que la présentation de l'image et elle est identifiée par une chaîne de caractères créée avec la fonction newRegion() qui est implantée par l'adaptateur.

– R4 : pour chaque présentation feuille p, associer son contenu à un début (p.delay()) et à une durée (p.duration()) de présentation et les rassembler dans une région calculée par la fonction getRegion() :

```
<img src=" + p.content.URL + "
     region=" + getRegion() + "
     begin=" + p.delay() + s"
     dur=" + p.duration() + s" />
```

En utilisant ces règles, un planificateur parcourt l'arbre et synchronise les présentations en utilisant les opérateurs <seq> </seq> pour les nœuds utilisant les relations temporelles before, meet, bi, mi et <par> </par> pour les nœuds implantant les autres

relations. Par exemple, pour la présentation des images de Grenoble et Montmartre, le résultat de la génération vers une présentation SMIL est la suivante :

```
<smil>
  <head> <layout> <root-layout width="370" height="330" />
      <region id="region1" left="110" top="70"
                           width="180" height="120" />
      <region id="region2" left="250" top="150"
                           width="120" height="180" />
  </layout> </head>
  <body> <par>
      <img src="file://localhost/Montmartre.gif"
           region="region1" begin="0.0s" dur="15.0s" />
      <img src="file://localhost/Grenoble.gif"
           region="region2" begin="5.0s" dur="10.0s" />
  </par> </body>
</smil>
```

Elle est composée par deux régions region1 et region2 qui spécifient la position et la taille pour présenter les images de Grenoble et de Montmartre. Les images sont synchronisées avec l'opérateur <par> </par> par rapport à leurs durées décrites sous forme d'intervalle (par ex. `begin="5.0s" dur="10.0s"`).

Dans certains cas, la plate-forme cible peut ne pas supporter toutes les opérations associées aux types faisant partie d'une présentation résultante. Ceci implique qu'il n'existe pas de règle de correspondance implantée par l'adaptateur. Par exemple, dans une plate-forme SMIL, il n'y a pas d'opération crop associée aux images. Dans ce cas, l'adaptateur exécute d'abord l'opération pour obtenir un objet d'un type qui soit supporté par la plate-forme cible. Dans le cas des images, il applique la méthode crop à l'objet Image et ensuite il transforme l'image résultante vers un objet image du modèle cible.

Enfin, pour les cas où le contenu d'une présentation atomique est lui même une présentation, les adaptateurs implantent une règle de correspondance spécifique. Afin d'expliquer ces aspects, considérons la présentation P_{TP} (cf. figure 5.10) qui combine un texte à la position (0, 0) et la présentation de Montmartre et Grenoble P_{MG} définie précédemment à la position (0, 80).

Les composants $\mathcal{P}_{MG}$ – qui font partie de $\mathcal{P}_{TP}$ – sont organisés par rapport à l'origine de la présentation $\mathcal{P}_{MG}$. Pour les intégrer, il faut alors recalculer leurs positions

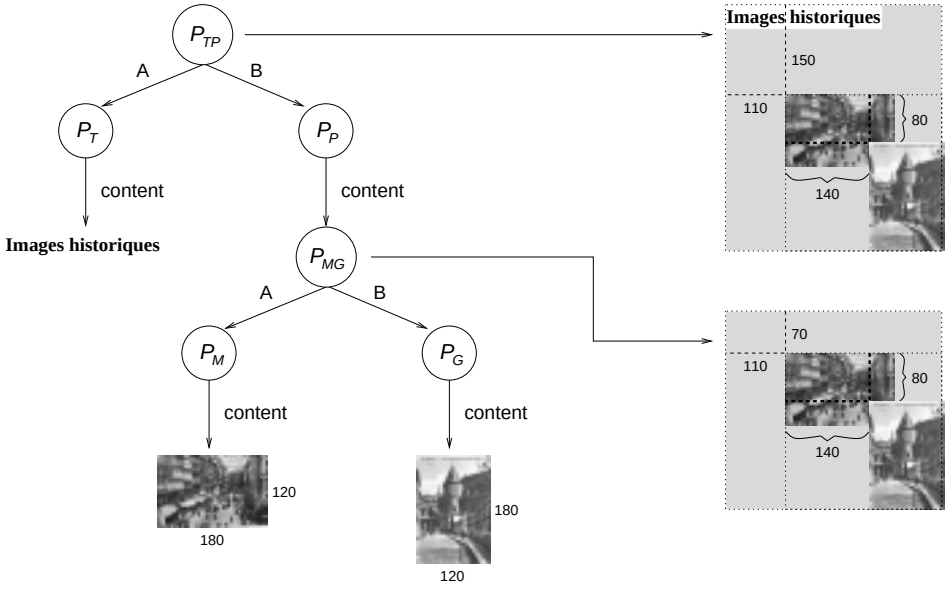


FIG. 5.10 – *Adaptation de présentations*

par rapport au nouveau référentiel (cf. figure 5.10) et obtenir $\mathcal{P}_{TP}$.

5.4 Construction de présentations

La construction de présentations est réalisée par le médiateur à partir d'une expression OQLiST. Ce processus se fait en deux phases : (1) récupération de données auprès des sources et du serveur de présentations et ; (2) intégration de données dans une présentation.

5.4.1 Récupération de données

La récupération de données se fait entre le médiateur et les extracteurs et le serveur de présentations (cf. figure 5.11).

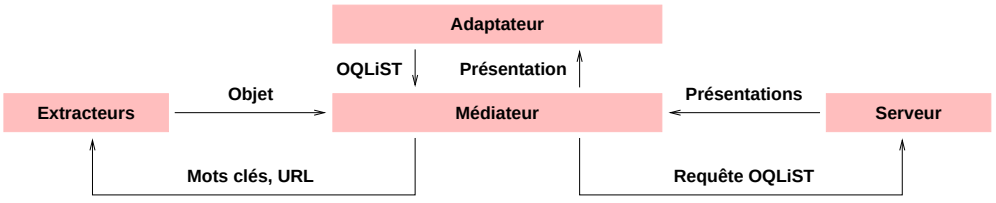


FIG. 5.11 – *Récupération de données*

Réutilisation de présentations Tout d'abord le médiateur fait évaluer des expressions OQLiST auprès de son serveur de présentations pour déterminer si tout ou partie d'entre elle a déjà été créée. Les présentations récupérées peuvent être réutilisées complètement ou partiellement dans des nouvelles présentations.

Extraction de données Un extracteur récupère des données en utilisant leurs URL, auprès des bases de données ou des moteurs de recherche (par ex. AltaVista, NorthernLight, etc.). La récupération de données est guidée par les règles d'interaction implantées pour un extracteur donné (cf. section 5.2).

Chaque extracteur est spécialisé pour interagir avec un type de source spécifique, il connaît son adresse, son protocole de communication, le type et le format du résultat. Afin d'exécuter des requêtes, un extracteur reçoit le type d'objets à récupérer (images, textes, sons, vidéos, présentations SMIL, etc.), des contraintes décrivant des critères de sélection (mots clés) et, éventuellement, le nombre maximal d'éléments à récupérer.

Une fois que les données ont été récupérées, l'extracteur les transforme vers des objets du modèle de données du gestionnaire. Les règles spécifient comment calculer les caractéristiques spatiales et temporelles de chaque type de donnée pour ensuite en définir un objet. Les résultats de requêtes des extracteurs sont des collections de données. Les règles d'extraction spécifient comment transformer les collections en des collections d'objets du modèle.

5.4.2 Intégration de présentations

L'intégration de présentations se fait par rapport à une expression OQLiST. Toute présentation est construite par rapport à un contexte de présentation. Le médiateur utilise les règles de construction du contexte de présentation pour intégrer les objets, c'est-à-dire chaque objet est associé à sa présentation par défaut. Enfin, la présentation est envoyée à l'adaptateur correspondant pour la mise en forme (cf. sous-section 5.3.2).

5.5 Mise en place d'un gestionnaire

La figure 5.12 présente l'architecture de JAGUAR qui permet de spécifier des gestionnaires de présentations. La mise en place d'un gestionnaire est réalisée en spécifiant un contexte de présentation, un schéma de données et des règles de transforma-

tion et de génération. À partir de ses informations, JAGUAR génère un gestionnaire de présentations adapté pour un ensemble d'applications et d'extracteurs de sources.

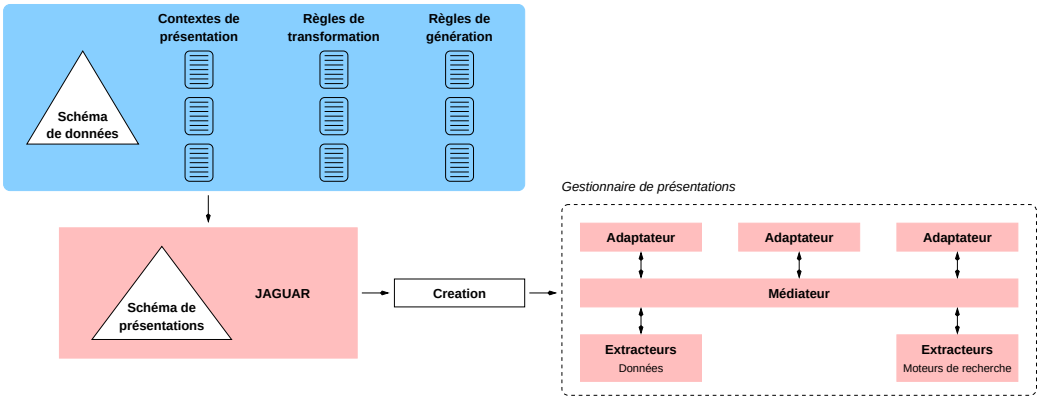


FIG. 5.12 – *Architecture de* JAGUAR

5.5.1 Modèle de Jaguar

JAGUAR implante un modèle générique qui permet de spécifier les données gérées par un gestionnaires. La figure 5.13 illustre ce modèle sous forme d'un diagramme UML. Le modèle fournit deux classes principales, la classe générique Presentation qui permet d'associer des présentations à tout objet du modèle et la classe Intrel qui caractérise une relation d'intervalle[1].

Classe présentation Les présentations peuvent être atomiques et composites (Atomic, Compound) avec des caractéristiques spatiales (Spatial 2D, Spatial 3D), temporelles (Temporal) et spatio-temporelles (Spatial 2D et Temporal, Spatial 3D et Temporal) différentes.

Rappelons que la composition de présentations se fait par des relations temporelles et spatiales entre des présentations atomiques et composites. Les relations sont toutes exprimées de manière homogène avec des relations d'intervalles. Chaque type de présentation composite, contient la définition des relations temporelles et/ou spatiales en termes de relations d'intervalles telles que nous les avons présentées dans le chapitre 3.

1. Rappelons que nous adoptons des relations d'intervalles pour décrire les relations entre les objets cf. chapitre 3.

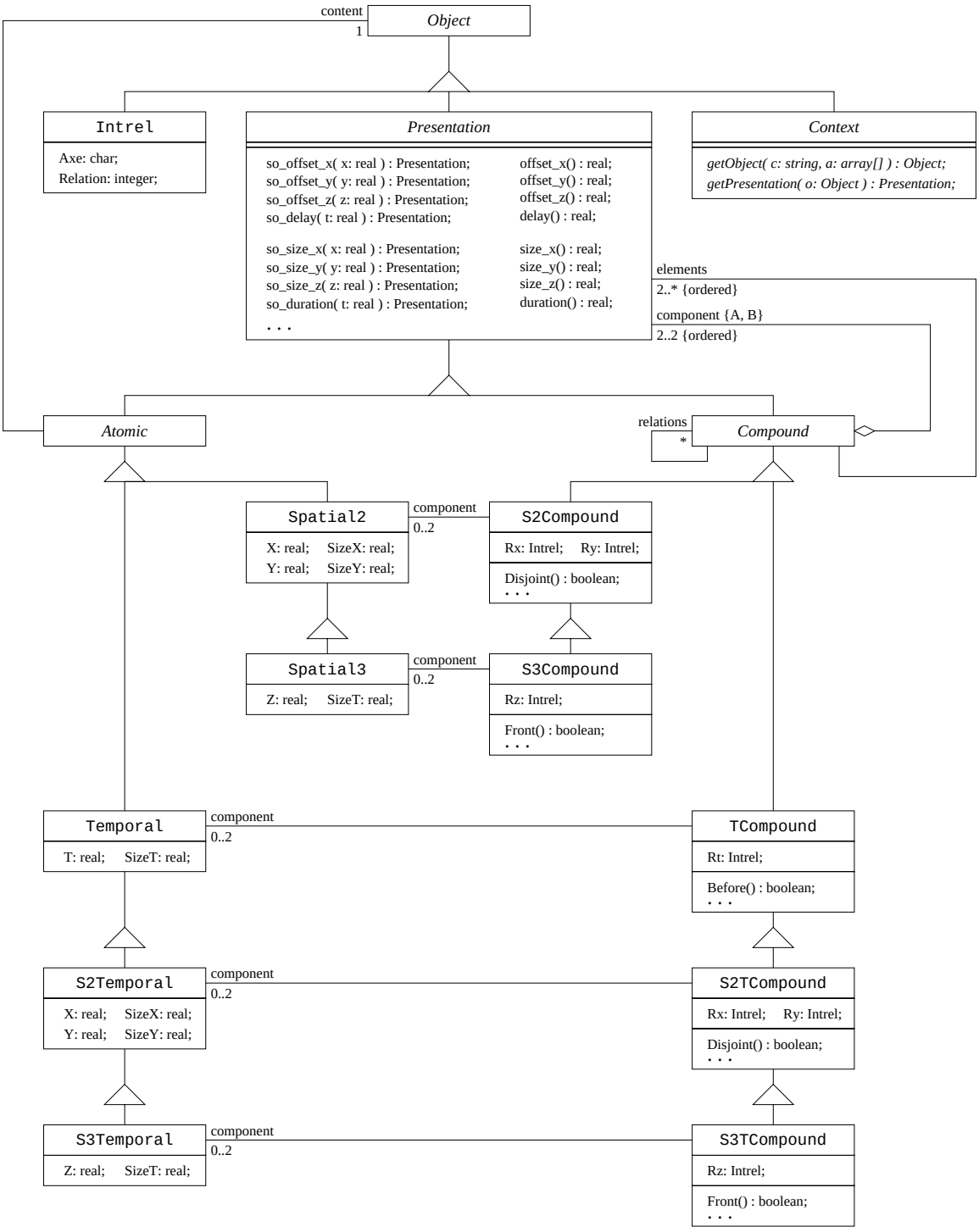


FIG. 5.13 – *Modèle de présentations de* JAGUAR

Présentation atomique Tout type de base (entier, réel, texte, image, document) est associé à différents types de présentations atomiques décrits dans le modèle spatio-temporel. Par exemple une image peut être associée à une présentation Spatiale 2D alors qu'un son est associé à une présentation Temporelle. Ceci n'empêche pas de composer des présentations en combinant des présentations de types différents .

Présentation composite La composition de présentations est guidée par des règles. Comme nous l'avons vu au chapitre 3, nous définissons des relations d'ordre entre des types de présentations pour ensuite spécifier des règles de composition. Dans le cas d'un modèle spatio-temporel (Spatiale 2D et Temporelle) comme celui de notre exemple, l'ordre des types est défini comme suit : Temporelle $<$ (Spatiale 2D et Temporelle), Spatiale 2D $<$ (Spatiale 2D et Temporelle). Les règles de composition de présentations sont :

- T_2 si $T_1 \leq T_2$;
- T_1 si $T_1 > T_2$;
- $T_1 + T_2$ si $T_1 \not\leq T_2$ et $T_1 \not> T_2$.

Par exemple si nous combinons une présentation Temporelle avec une présentation Spatiale 2D, nous obtenons une présentation composite Spatiale 2D et Temporelle. Ces règles sont implantées par le médiateur d'un gestionnaire de présentations.

Classe Intrel Une relation d'intervalle est caractérisée par la classe Intrel. Elle concerne un Axe qui décrit la ligne spatiale ou temporelle où l'intervalle se projette. L'attribut Relation est un entier représentant une des treize relations d'Allen. Les relations sont ordonnées dans une énumération où chaque position dénote une relation.

5.5.2 Spécification des contextes de présentation

Le modèle de données générique de JAGUAR est utilisé pour définir un contexte de présentation.

Classe contexte La classe abstraite Context implante une association entre les types décrits par le schéma de données de l'application et des types de présentation du modèle spatio-temporel. Elle définit un ensemble de méthodes abstraites représentant

des opérations pour interagir avec des extracteurs (getObject) et pour obtenir les présentations par défaut des objets (getPresentation).

La classe Context est spécialisée pour définir les contextes de présentation. Un gestionnaire peut utiliser plusieurs contextes pour gérer les présentations d'une application.

Interaction avec les applications Les interfaces entre un gestionnaire et les types de plates-formes et de sources avec lesquelles il interagit doivent aussi être spécifiées. Il s'agit de définir les protocoles d'interaction et les modèles de données d'échange. Pour cela, le programmeur d'un gestionnaire de présentations doit définir des règles de transformation, de génération et d'extraction de données.

5.5.3 Spécification d'un schéma de données

Les objets d'un gestionnaire sont décrits dans un schéma de données. Ils sont spécifiées par spécialisation de la classe Object. Des classes doivent être définies pour chaque type du schéma utilisé (i.e., valeurs, collections, documents, etc.).

Le modèle fournit un ensemble de classes prédéfinies avec des constructeurs par défaut. Ces classes sont classifiées en : (1) valeurs atomiques (par ex. Integer, Real, Char, String, Collection, etc.) ; et (2) objets multimédias Document (par ex. Text, Image, Audio, Video, SMIL, etc.) (cf. figure 5.14).

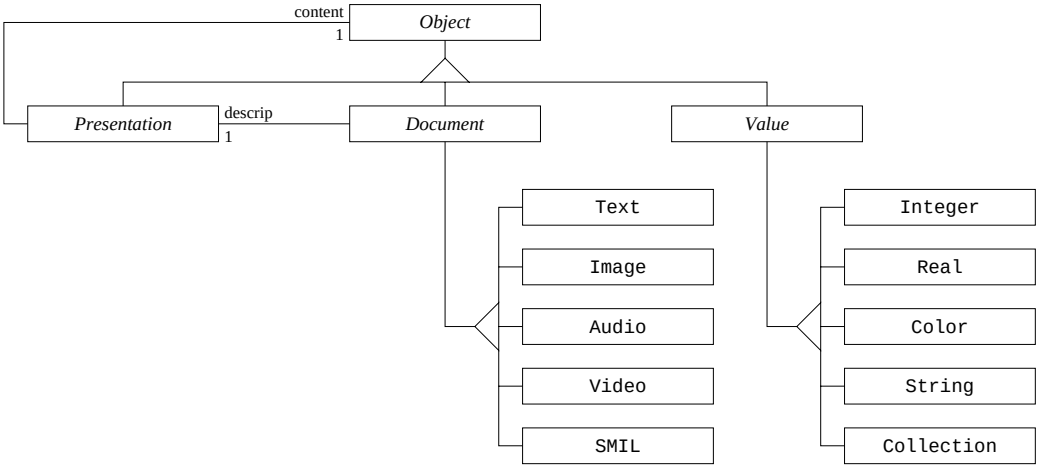


FIG. 5.14 – *Types prédéfinis dans le modèle de* JAGUAR

Ces classes peuvent être spécialisées pour associer un nouveau constructeur à un

type pour un contexte de présentation donné. Par exemple pour définir une nouvelle implantation des opérations associés à un objet.

Classe document associe des caractéristiques spatiales et/ou temporelles à un objet référencé par une URL. Selon le type de donnée référencée, le constructeur accède au fichier et calcule la taille spatiale et/ou temporelle d'un objet. Par exemple Image(URL = "file://www-lsr.imag.fr/FrenchMap.gif") définit un objet de type Image. La taille est calculée par les gestionnaires en utilisant une librairie de classes spécialisée.

Classe valeur (Integer, Real, Char, String). Le constructeur associe à cette classe calcule la taille de la valeur. Pour cela, une valeur de type entier, caractère, flottant est transformée en chaîne de caractères d'une police particulière. La taille de la valeur est ensuite calculée selon le nombre de caractères qui la composent. Par exemple en supposant une police mono-espace (7×14 pixels par caractère), la taille du caractère "5" est de 7×14 pixels.

Classe collection définit un constructeur pour les collections de types de données. Par exemple une liste peut être présentée soit comme une liste d'éléments dans une colonne, dans une ligne, dans un tableau, soit comme le résultat d'une opération d'agrégation (cardinalité, max, min, etc.). Précisons que les types du modèle de données d'un gestionnaire peuvent être associés à des types de présentations atomiques ou composites.

5.5.4 Génération

La figure 5.15 illustre le processus de génération implanté par JAGUAR. Étant données un schéma de données, au moins un contexte de présentation et des règles d'interaction avec les applications, JAGUAR génère :

- une librairie des classes qui implantent des types d'adaptateurs définis par le constructeur d'un gestionnaire ;

- une interface applicative (API) associé aux classes générées qui inclue l'API du médiateur qui permet aux adaptateurs d'interagir avec le gestionnaire de

présentations. L'API générée fournit :

- les interfaces des adaptateurs pour permettre aux applications de communiquer avec le gestionnaire pour faire évaluer leurs requêtes ;
- les interfaces devant être utilisées par le gestionnaire pour interagir avec les extracteurs des sources.

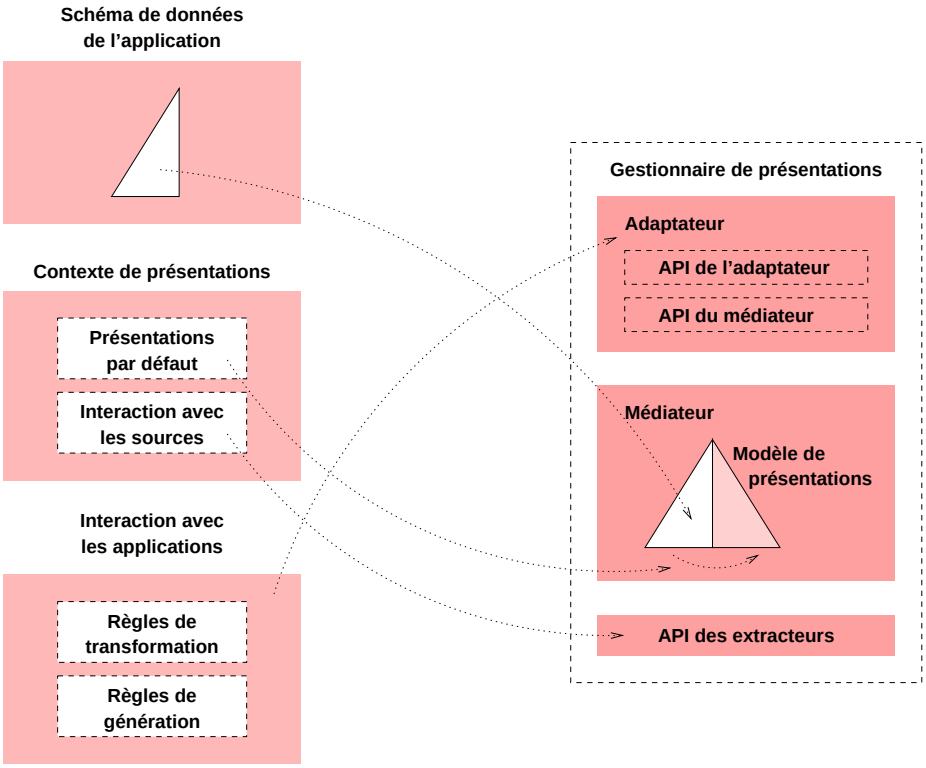


FIG. 5.15 – *Processus de génération*

Il est couplé avec l'API du médiateur qui fournit l'interface suivante :

```
public interface Mediator {
  void setContext( Context context );
  Presentation evaluate( String oqlistExpression );
}
```

La génération rassemble alors les classes dans une librairie que toute application utilisant un gestionnaire de présentations doit importer. L'API associée à la librairie de classes est utilisée par les applications pour mettre un place un contexte de

présentation (setContext) par rapport auquel le gestionnaire va évaluer des requêtes (evaluate) pour construire les présentations spécifiées.

5.6 Conclusion

Nous avons présenté JAGUAR en montrant comment notre infrastructure permet de spécifier des gestionnaires de présentations. Un gestionnaire fédère des bases de données hétérogènes et distribuées (Web, SGBD, bases d'images, etc.) et des applications multimédias (exécuteurs, systèmes auteurs). Les gestionnaires peuvent interagir avec des sources hétérogènes et distribuées gérant des données de formats différents. Ensuite, les données sont intégrées dans des présentations grâce à un modèle de données spatio-temporel qui traite les données de manière homogène. Il s'agit d'un modèle pivot qui intègre des données hétérogènes dans des présentations pour les transformer vers des contextes applicatifs différents. Les résultats sont enfin mis en forme selon des formats adaptés à des plates-formes spécifiques.

La contribution de notre approche consiste à offrir des gestionnaires qui modularisent la gestion de présentations et rendent transparents les problèmes de récupération et de mise en forme de présentations. Ils traitent de manière homogène les médias et les documents, représentés dans des standards permettant donc leur réutilisation et leur intégration dans des nouvelles présentations.

Notre approche se veut générique mais l'infrastructure que nous proposons nécessite l'intervention d'un programmeur pour générer un gestionnaire spécifique et des classes correspondantes. Des extensions peuvent être envisagées pour les gestionnaires concernant le partage de données entre les applications. À l'heure actuelle, seuls les applications utilisant le même contexte de présentations peuvent partager des données. L'intégration des données définies dans des contextes différents est un problème d'intégration de schémas qui doit être exploré.

Par ailleurs, la coopération entre des gestionnaires de présentations est aussi une extension qui permettrait le partage et la réutilisation de données. Des gestionnaires pourraient être spécialisés par type d'application, par type de source ou par types de données gérées. Ces gestionnaires pourraient être accédés par des applications selon des besoins particuliers.

Chapitre 6

Implantation d'un gestionnaire de présentations

Ce chapitre présente la mise en œuvre d'un gestionnaire de présentations. L'objectif est de valider notre proposition et particulièrement la médiation entre des applications et des sources. Nous nous sommes intéressés à l'implantation d'un gestionnaire adapté pour des applications construites sur des plates-formes standards et pour des sources accessibles à travers le Web.

Le gestionnaire implanté est un environnement client-serveur qui supporte un modèle spatio-temporel et qui fournit des outils spécifiques pour stocker, récupérer et visualiser des présentations. Il communique avec un serveur bases de données multimédias à objets, une extension spatio-temporelle que nous avons implanté sur le SGBD O_2 [BDK92, AC93]. Il utilise des langages standards tels que :

- OQL pour définir OQLiST,
- Java et SMIL pour l'exécuter de présentations avec les applications,
- Java pour programmer le médiateur.

Finalement, il utilise des outils tels que *ImageMagick* [Cri] pour appliquer des opérations telles que *geometry*, *crop*, *blur* sur les images et, enfin, *Java Media Framework* [GT99, SI] (JMF) et SOJA [Fou] pour l'exécution des présentations.

La section 6.1 présente l'architecture du gestionnaire et ses fonctions principales. Elle décrit également le modèle que le gestionnaire implante (schéma de données et contexte de présentation). La section 6.2 décrit le mécanisme de médiation implanté. Ensuite, la section 6.3 présente la génération de présentations pour les plates-formes

JMF et SMIL. La section 6.4 décrit les expériences d'utilisation du gestionnaire. Enfin dans la section 6.5, nous concluons ce chapitre en discutant les résultats et les perspectives envisageables.

6.1 Architecture

La figure 6.1 montre l'architecture du gestionnaire de présentations implanté qui est une instance de JAGUAR. Il est composé d'un médiateur, des extracteurs de données, d'un adaptateur JMF et d'un adaptateur SMIL. Les adaptateurs fournissent une interface OQLiST qui permet d'interagir avec les applications pour définir des présentations, et produisent un programme JMF ou SMIL qui peut être exécuté par une application. Les extracteurs utilisent un cache dans l'espace de travail pour stocker et réutiliser les objets récupérés, par exemple les médias provenant du Web. L'espace de travail stocke trois types d'objets (fichiers) : locaux associés aux applications ; récupérés auprès des sources ; et transformés suite à la construction d'une présentation SMIL. Il laisse ces objets accessibles aux adaptateurs pour les réutiliser lors de la génération et l'exécution d'autres présentations.

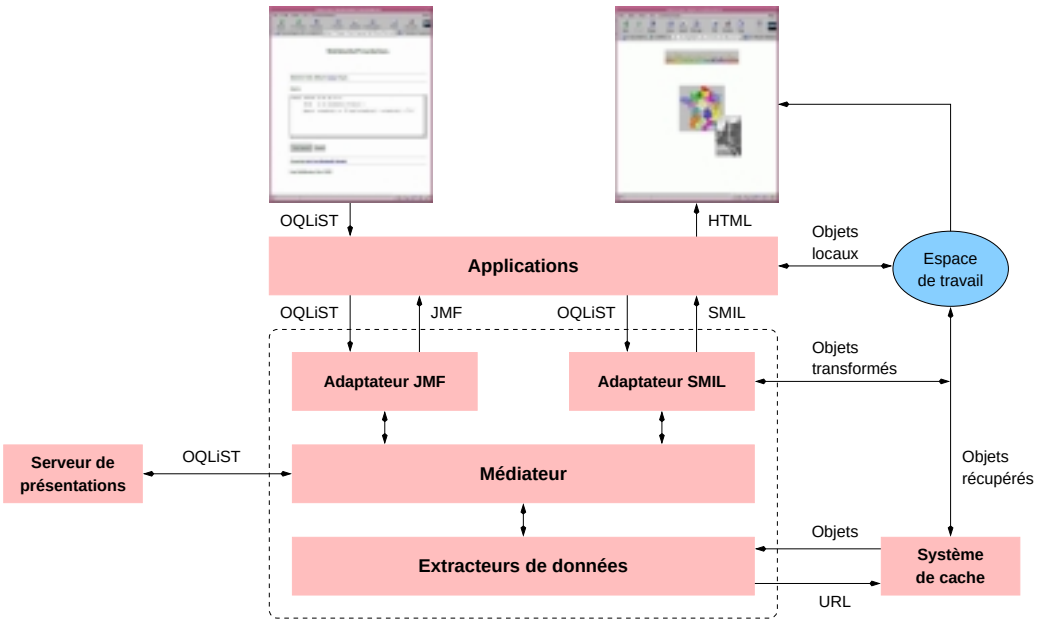


FIG. 6.1 – *Architecture du gestionnaire*

6.1.1 Fonctions implantées

La version actuelle de JAGUAR implante des mécanismes de base pour la spécification et l'exécution de présentations spatio-temporelles. L'interaction avec le gestionnaire se fait à travers une interface OQLiST qui permet de :

- construire des présentations spatio-temporelles à partir de présentations existantes et des médias définis dans différents formats,
- stocker des présentations,
- interroger les attributs des présentations,
- spécifier les résultats des requêtes comme des présentations avec des caractéristiques spatio-temporelles explicites ou implicites. Ces caractéristiques peuvent être modifiées selon les besoins des utilisateurs.

Le gestionnaire fournit aussi une interface applicative (API) qui permet de configurer l'adresse et le port d'écoute du serveur de présentations, le contexte de présentation et le type d'adaptateur (i.e., JMF ou SMIL). Toute application peut changer son contexte et son adaptateur en cours d'exécution selon les types de données qu'elle veut présenter et les présentations par défaut qu'elle associe à ses objets. Les schémas de données et les contextes de présentation sont définis à l'avance par les programmeurs d'applications.

6.1.2 Schéma de données spatio-temporel

Le noyau de JAGUAR implante un ensemble de classes décrivant un schéma spatio-temporel générique à partir duquel un programmeur peut définir le schéma de données qu'il souhaite utiliser pour un ensemble d'applications. Le schéma implanté décrit des objets de type Presentation, Document et Value (cf. figure 6.2).

Classe abstraite présentation La classe Presentation spécifie les méthodes abstraites setBorder et setBackground qui permettent d'associer des éléments de décor à une présentation (le cadre et l'arrière plan). La méthode Hyperlink associe un hyperlien à une présentation vers une page Web ou vers une autre présentation.

```
public abstract class Presentation implements SpatioTemporal {
    public abstract void setBorder( double x, double y );
```

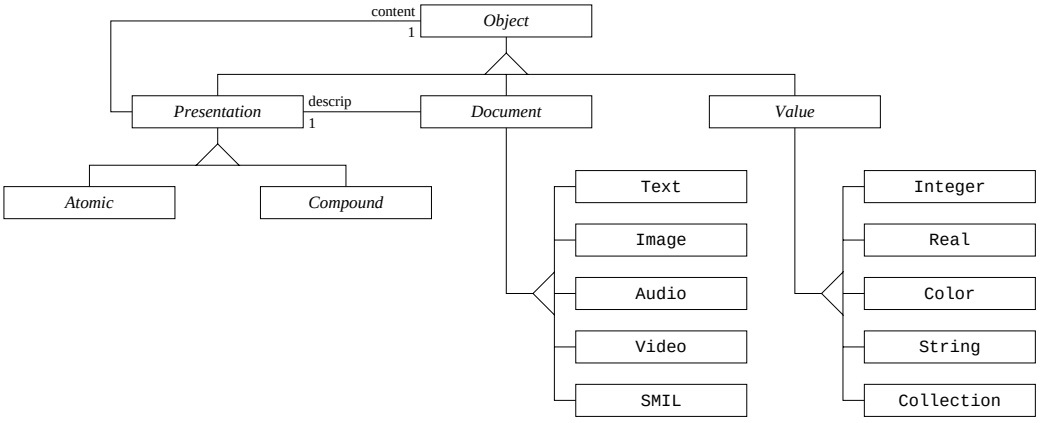


FIG. 6.2 – *Schéma de données du gestionnaire*

```
    public abstract void setBackground( Presentation so );
    public abstract void setHyperLink( Object hyperlink );
}
```

Cette classe implante l'interface SpatioTemporal qui fournit un ensemble de constants énumérant les axes (x, y, z, t). La constante FREE est la valeur associée à un attribut lorsqu'il est indéterminé. Par exemple puisque les sons n'ont pas de propriétés spatiales inhérentes, la présentation spatio-temporelle d'un son à une position et une taille indéterminées dans l'espace. Elle fournit aussi les méthodes getOffset, getSize qui permettent de récupérer la position et la taille d'une présentation sur une dimension (axis) et les méthodes setOffset, setSize qui associent une position et une taille à une présentation.

```
public interface SpatioTemporal {
    public final static double FREE = Double.MAX_VALUE;
    public final static char X = 'x'; /* axes spatiaux: x, y, z */
    public final static char Y = 'y';
    public final static char Z = 'z';
    public final static char T = 't'; /* axe temporel: t */
    public double getOffset( char axis ); /* décalage et taille */
    public double getSize( char axis );
    public void setOffset( char axis, double value );
    public void setSize( char axis, double value );
}
```

Types de présentations Les classes Atomic et Compound sont des spécialisations de la classe Presentation. Les présentations de type Atomic peuvent être associées à tout type d'objet. La classe Compound décrit une relation spatio-temporelle entre deux présentations. Elle définit des méthodes booléens qui permettent de vérifier si deux présentations ont une relation spécifique. Nous avons implanté des relations topologiques et directionnelles pour un espace 2D. Nous combinons les relations spatiales avec les relations temporelles pour spécifier et interroger des présentations Spatiale 2D et Temporelle. Pour cela, les relations spatiales sont implantées en termes de relations d'intervalles d'Allen.

Types de relations Une présentation composite Spatiale 2D et Temporelle dispose de :

- 13 méthodes implantant les relations temporelles: `Before()`, `Meet()`, `Overlap()`, `Start()`, `During()`, `Finish()`, `Equal()`, `Bi()`, `Mi()`, `Oi()`, `Si()`, `Di()`, `Fi()`;
- 2 méthodes implantant les relations temporelles génériques : `Seq()`, `Par()` ;
- 8 méthodes implantant les relations topologiques : `Disjoint()`, `Touch()`, `Cover()`, `Covered_by()`, `Inside()`, `Contain()`, `Spatial_Equal()`, `Spatial_Overlap()` ;
- 8 méthodes implantant les relations directionnelles : `North()`, `South()`, `West()`, `East()`, `Northeast()`, `Southeast()`, `Northwest()`, `Southwest()`.

6.1.3 Contextes de présentation

Le gestionnaire de présentations fournit l'interface Context qui définit l'interaction avec les extracteurs des objets et la construction de présentations.

```
public interface Context {
  public Object getObject( String constructor, Object [] args );
  public Presentation getPresentation( Object object );
}
```

La méthode getObject spécifie l'interaction avec les extracteurs d'objets. Elle détermine leurs caractéristiques spatiales et/ou temporelles selon leur format de définition. La méthode getPresentation associe une présentation par défaut à un objet selon son type. Le code qui implante cette interface est spécifié par un programmeur.

Dans notre prototype, nous avons implanté trois contextes de présentation, tous adaptés pour interagir avec des extracteurs qui récupèrent des objets à partir des adresses URL. Le contexte JMF est adapté pour construire des présentations avec des objets de type atomique (i.e., Value), texte, image, son et vidéo. Le contexte SMIL permet en plus d'exécuter des présentations de documents SMIL, mais pas d'objets vidéos. Enfin, le troisième contexte définit des présentations adaptées pour la présentation de données d'un entrepôt de données.

6.2 Mécanisme de médiation

Le médiateur intègre dans des présentations des objets multimédias provenant des sources différentes (cf. figure 6.3). Cette intégration est exprimée en OQLiST grâce aux constructeurs définis dans le contexte de présentation. Le langage permet de définir déclarativement des présentations en intention qui combinent des présentations existantes et/ou qui introduisent des nouveaux objets disponibles sur le Web à travers leurs URL. Le résultat final peut être stocké dans un serveur bases de données comme une nouvelle présentation (cf. figure 6.4).

6.2.1 Extracteurs de données

La récupération est exécutée par des extracteurs qui réalisent la recherche des objets explicitement spécifiés dans la définition d'une présentation. Étant donnée une adresse URL, un extracteur de données interagie avec un système de cache qui vérifie l'existence et la disponibilité des objets. Les extracteurs identifient les types de données récupérés et déterminent leurs caractéristiques spatiales et/ou temporelles (par ex. taille, durée) en utilisant des outils comme *ImageMagick* pour les images, *Java Media Framework* pour les vidéos et les sons, un analyseur syntaxique XML[1] pour les objets SMIL. L'interface des extracteurs et les paramètres dont ils ont besoin pour lancer des recherches est décrite dans le contexte de présentation utilisé par le gestionnaire (cf. figure 6.3).

1. SAX (*http://www.megginson.com/SAX*) permet d'analyser les présentations SMIL.

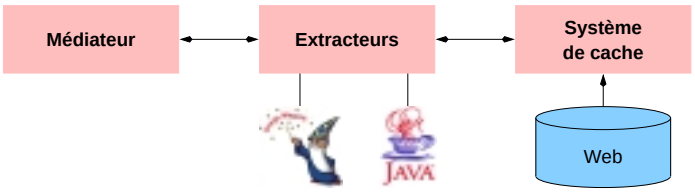


FIG. 6.3 – *Extracteurs de données*

6.2.2 Serveur de présentations

Nous avons implanté une extension spatio-temporelle de O_2 pour intégrer les aspects spatio-temporels des présentations multimédias. Pour cela, nous avons défini un schéma de présentations qui implante le modèle spatio-temporel présenté dans le chapitre 3. Nous avons également implanté une extension spatio-temporelle du langage OQL, le langage OQLiST pour la construction, le stockage et l'interrogation des présentations multimédias. Le serveur de présentations assure le permanence des présentations construites par le médiateur (cf. figure 6.4).

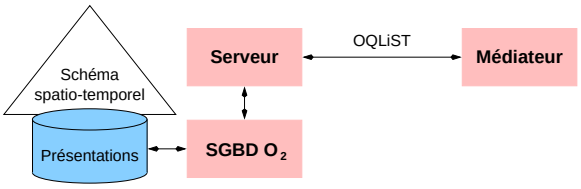


FIG. 6.4 – *Serveur de présentations*

6.3 Adaptateurs

La figure 6.5 présente l'architecture générale d'un adaptateur composé (1) d'un transformateur qui traduit la spécification d'une présentation exprimée dans un langage *ad-hoc* (utilisé par une application) et (2) d'un générateur qui traduit une présentation construite par le médiateur vers un programme exécutable par une plate-forme spécifique.

Dans la version actuelle de notre prototype, les présentations sont spécifiées en OQLiST raison pour laquelle les adaptateurs que nous avons mis en œuvre n'implantent pas les transformateurs. Pour les adaptateurs de notre prototype, nous avons implanté des générateurs spécialisés pour la génération de présentations sur des plates-

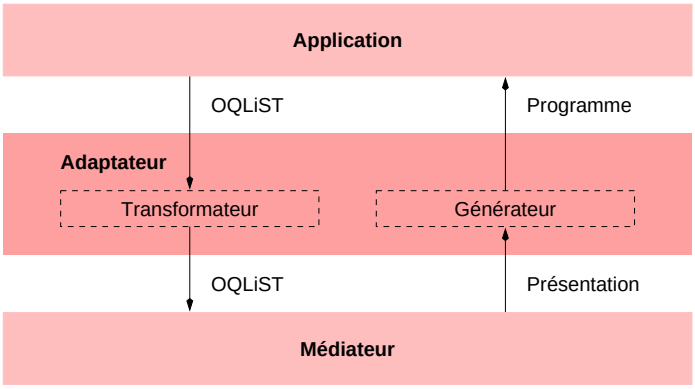


FIG. 6.5 – *Architecture générale d'un adaptateur*

formes JMF et SMIL.

Le but de la génération est de construire une spécification qui exécute une présentation. Selon la plate-forme cible, soit elle est un programme Java, soit un document SMIL. Les phases du processus de génération sont les suivantes :

1. Préparation des structures composant le programme. Par exemple les en-têtes (*headers*) d'un programme Java ou d'un document SMIL.

2. Construction du corps du programme. Ce processus consiste à parcourir un arbre de présentation de manière récursive afin de créer :

   – pour chaque nœud $\mathcal{PC}$ (présentation composite) une structure spécifiant une synchronisation entre ses fils ;

   – pour chaque feuille $\mathcal{PA}$ (présentation atomique) une structure décrivant les caractéristiques spatiales et temporelles de l'objet $\mathcal{O}$ contenu dans $\mathcal{PA}$ et l'adresse où $\mathcal{O}$ est physiquement disponible (par ex. dans l'espace de travail du gestionnaire).

3. Assemblage des entêtes avec les corps pour construire un programme ou un document.

Notre prototype fournit une classe abstraite Generator (cf. ci-dessous) qui spécifie trois méthodes : program, process_presentation, object. La méthode program doit implanter la construction d'une spécification pour une plate-forme cible (phases 1 et 3). La méthode process_presentation implante le parcours d'un arbre de présentation et la

méthode object implante la transformation des objets contenus dans les présentations feuilles (phase 2) .

```
public abstract Generator {
  public abstract String program( String name, Presentation so ) throws Exception;
  protected abstract void object( Object object,
                                  Size size, Time time ) throws Exception;
  protected void process_presentation( Presentation so ) throws Exception;
}
```

Une implantation de la classe Generator doit mettre en œuvre les méthodes program et object. La méthode process_presentation est fournie par le système car le parcourt d'un arbre est indépendant des plates-formes d'exécution. L'implantation des autres méthodes dépend de la plate-forme et elle est faite par un générateur. Nous décrivons dans la suite l'implantation des générateurs pour JMF et SMIL.

6.3.1 Générateur JMF

La figure 6.6 présente l'architecture du générateur JMF qui est basé sur l'API *Java Media Framework* [GT99]. Le programme JMF généré est compilé pour construire un *Applet* qui peut être exécuté sur un navigateur Web.

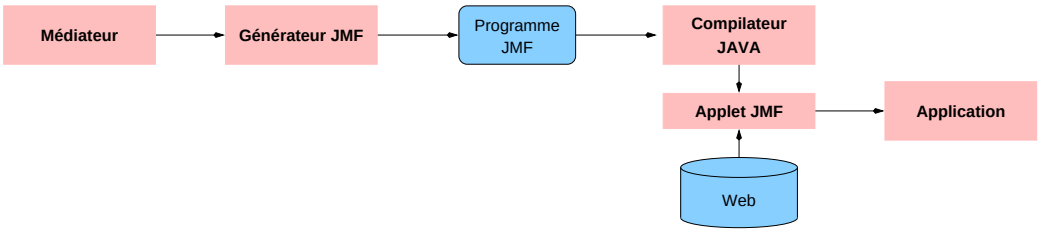


FIG. 6.6 – *Générateur JMF*

Structure d'un programme JMF

Un programme JMF est défini par une classe Java qui spécialise la classe Applet, il implante une présentation d'objets médias. Conceptuellement, il est organisé en trois parties :

– initialisation : elle définie les structures qui vont contenir les objets et des méthodes nécessaires pour construire le plan d'exécution ;

- administration : elle implante les méthodes qui permettent de gérer (démarrer, arrêter et détruire) chaque objet de la présentation ;

- synchronisation : elle implante le plan d'exécution des présentations. Par ailleurs, l'exécution des sons et vidéos est guidée par la production d'événements, par ex. une vidéo est prête pour être présentée (*prefetch*), l'exécution d'une vidéo a fini, etc. Ils sont détectés par un "module d'écoute" implanté pour tout objet de ces types.

Génération

Le processus de génération d'un programme JMF implanté par la méthode program met en place les en-têtes du programme, les structures et les variables nécessaires pour construire son corps. Il traite ensuite l'arbre de présentation avec la méthode process_presentation.

Pour chaque présentation feuille de l'arbre, il faut construire son code d'exécution et l'inscrire dans le plan d'exécution de la présentation. Cette opération est mise en œuvre par la méthode object du générateur qui utilise trois méthodes[2] :

```
String vars_code( Object object, String label );
String init_code( Object object, String label, Size size ) throws Exception;
String player_code( Object object, String label, Size size, Time time );
```

Les méthodes vars_code et init_code construisent respectivement le code de création et d'initialisation d'un objet (parties administration et initialisation). Ces méthodes utilisent des outils JMF lorsqu'il faut appliquer des opérations de transformation sur un objet[3]. La méthode player_code construit les modules d'écoute des sons et des vidéos.

Une fois construites les méthodes qui composent les trois parties du programme (initialisation, administration et synchronisation) program les rassemble et génère la classe Java qui met en oeuvre la présentation.

2. Notons que ces méthodes implantent les règles de génération de l'adaptateur JMF.
3. Pour l'application d'opérations de transformation en Java, il existe des librairies spécialisées telles que `java.awt` et `javax.media`.

6.3.2 Générateur SMIL

La figure 6.7 présente l'architecture du générateur SMIL. Il prend en entrée une présentation et génère un document SMIL, il est couplée avec un *Applet* SOJA pour être exécutée sur un navigateur Web.

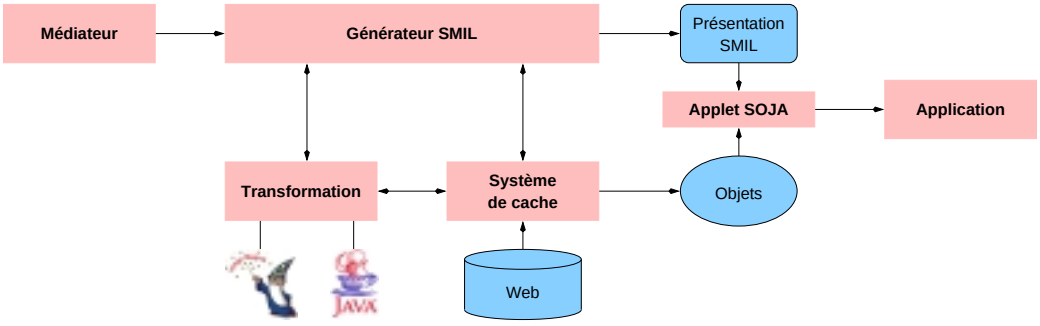


FIG. 6.7 – *Générateur SMIL*

Structure d'une présentation SMIL

De manière générale, la structure d'une présentation SMIL est organisée en deux parties :

- la section <layout> </layout> spécifie l'ordonnancement spatial des présentations. Cette partie est composée d'un ensemble de structures <region ... />, une pour chaque objet composant une présentation. Une telle structure décrit les attributs spatiaux (position et taille) de la région où l'objet sera présenté :

```
SD ::= <region id="<IdRegion>"
               left="<Integer>"
               top="<Integer>"
               width="<Integer>"
               height="<Integer>" />
```

- la section <body> </body> pour l'ordonnancement temporel des présentations. Elle spécifie une expression de synchronisation en composant des descriptions temporelles des objets avec des opérateurs. La description temporelle TD d'un objet a la forme suivante :

```
TD ::= <MediaType> src="<URL>"
                   region="<IdRegion>"
                   begin="<Real>s"
                   dur="<Real>s" />
```

Elle caractérise un média (<MediaType>) par son adresse URL, sa région de présentation, son début et sa durée de présentation (begin, dur). Notons que la région de présentation correspond à une des régions spécifiées par des descriptions spatiales SD dans la section layout.

Une expression de synchronisation est de la forme :

```
Exp ::= <TD> | <Operator> <Exp> <Exp> </Operator>
```

Elle associe des définitions temporelles avec des opérateurs de synchronisation : <par> </par> ou <seq> </seq>.

Génération

En tenant compte cette structure, la méthode program du générateur SMIL construit les en-têtes d'un document et traite l'arbre de présentation avec la méthode process_presentation. Pour chaque présentation feuille de l'arbre, elle fait construire son code de présentation par la méthode object du générateur.

La méthode object construit le code de présentation d'un objet en utilisant les méthodes [4] suivantes :

```
String layout_code( Object object, String label, Size size );
String body_code( Object object, String label, Size size ) throws Exception;
```

Les méthodes layout_code et body_code implantent respectivement las construction des structures des descriptions spatiales (SD) et temporelles (TD). Enfin, la méthode program construit les sections layout et body en rassemblant les spécifications spatiales et temporelles calculées par object. Ensuite il les intègre avec les en-têtes et construit le document SMIL.

4. Ces méthodes implantent les règles de génération de l'adaptateur SMIL.

6.4 Expérimentation

L'objectif de nos expériences a été de montrer l'utilisation de notre gestionnaire de présentations pour aider à la visualisation de données multimédias dans des applications différentes. Un deuxième objectif a été de montrer l'utilisation de notre langage OQLiST pour définir des présentations d'objets. Dans un premier temps, nous avons choisi une application touristique où des images et des informations textuelles doivent être présentées dans des cas de figures différents et dans un contexte Web. Dans un deuxième temps, nous avons défini des outils pour la visualisation de données pour un entrepôt de données OLAP [Cod93, CD97].

6.4.1 Application touristique

Nous avons spécifié et implanté une application touristique qui permet de visualiser différentes données décrivant des sites de France. Les présentations des applications peuvent être exécutées sur un environnement *Java Media Framework* ou SMIL. Le gestionnaire de présentations permet d'intégrer les applications avec des données accessibles sur l'internet. Selon le type de plate-forme utilisée, les applications implantent des contextes de présentation différents.

Régions françaises L'application présente une image de la carte de France associée aux noms des régions (cf. annexe B). Pour cela, le contenu de la carte est décrit par une présentation composite des régions françaises (cf. figure 6.8). Chaque région est une sous-image décrite par sa position, sa taille et son nom. L'expression OQLiST suivante spécifie cette présentation :

```
select e.content border 2, 2 background black
                    using e at 2 * e.offset_x, 2 * e.offset_y
        compound ((Document) e.content).getName
                     fit e.size_x + 100, 14 during e.duration
        touch, centered, south synchronize equal
from    e in descrip( FrenchMap ).elements order by *
```

Dans notre prototype, la description du contenu des médias est faite manuellement. Néanmoins, il est envisageable de faire coopérer un système de reconnaissance de contenu d'images avec le médiateur de JAGUAR. Une fois décrite, nous pouvons utiliser

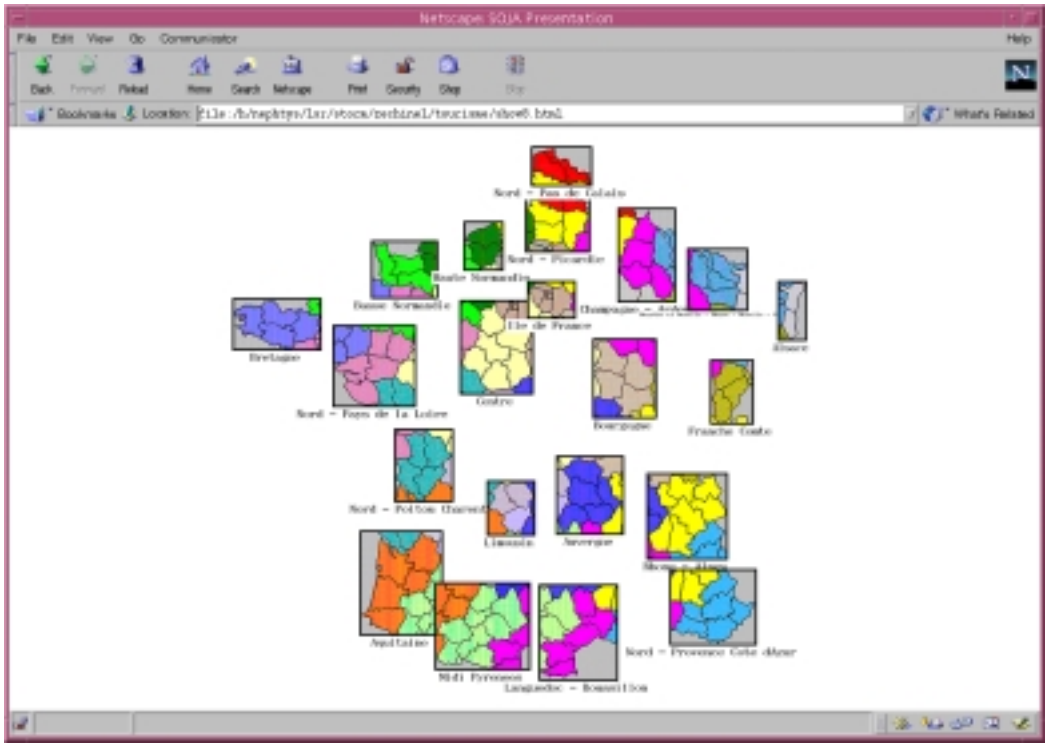


FIG. 6.8 – *Description de la carte de France*

l'information contenue dans la description de la carte pour définir une présentation des noms des régions de la France comme suit :

```
select last( e compound ((Document) e.content).getName
                        with e.size_x – 2, 14 during e.duration
             inside, centered, centered synchronize equal )
from  e in descrip( FrenchMap ).elements order by *
```

En analysant l'expression OQLiST, nous récupérons le nom (cf. getName) et la position de chaque région e de la description de l'image FrenchMap. Pour chaque nom, nous changeons sa taille (cf. with e.size_x – 2, 14) et sa durée (cf. during e.duration) et nous le plaçons au centre de la région correspondante (cf. inside, centered, centered). Enfin, nous spécifions que toutes les présentations des noms doivent commencer et finir en même temps (cf. synchronize equal). Ensuite, nous spécifions la construction d'une liste de présentations (cf. order by *) pour les composer dans une nouvelle présentation. La figure 6.9 présente le résultat. Un utilisateur peut ainsi choisir et repérer la région qu'il/elle souhaite visiter.

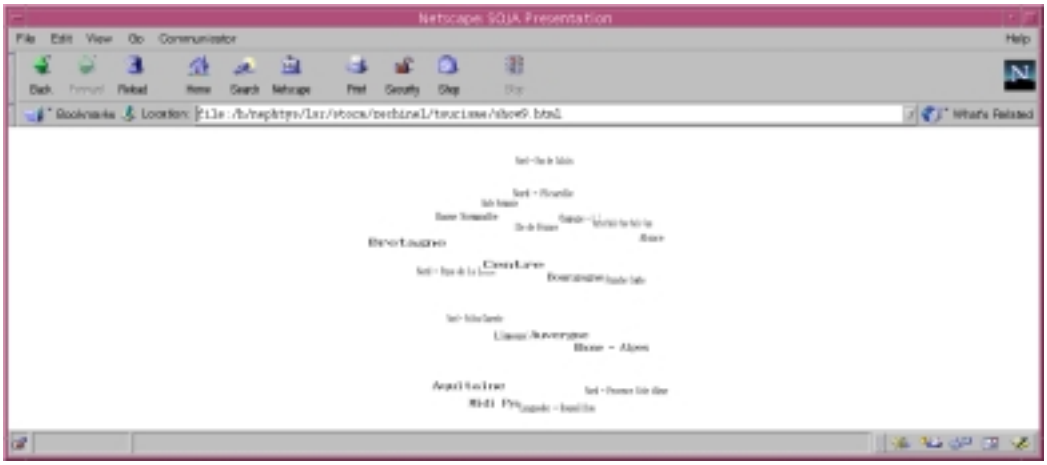


FIG. 6.9 – *Noms des régions de la France*

Stations de ski Les stations de skis sont l'un de plus connus des endroits touristiques en France. Un visiteur pourrait afficher des images de ces stations et les présenter dans un tableau. D'autres informations peuvent aussi être affichées, par exemple des images montrant les activités de loisir que l'on peut réaliser dans les stations (cf. figure 6.10). Pour compléter cette information, une image de la région où les stations peuvent être visitées pourrait être affichée avec les tableaux d'image. Cette région est récupérée à partir de la description de la carte de France. Enfin, chaque image présentée dans ces tableaux peut être associée à un hyperlien de type bouton. En cliquant, l'image est affichée indépendamment du tableau. L'expression OQLiST correspondante est :

```
define Table1 as
        atomic( select e button e.URL from e in
                select Image( URL = h ) from h in directory( "file://localhost/h" )
                table 4 border 1, 1 );
define Table2 as
        atomic( select e button e.URL from e in
                select Image( URL = v ) from v in directory( "file://localhost/v" )
                table 5 border 1, 1 with 700, 200 );
define Region as
        element( select e from e in descrip( FrenchMap ).elements
                  where ((Document) e.content).getName = "Rhone - Alpes"
                ).content;
```

```
define LogoRA as
        atomic( Region
                compound String( Value = " Les Deux Alpes en Rhone Alpes " )
                        fit 2.3 * Region.getWidth, 2.3 * Region.getHeight
                disjoint 44, centered, south at 0, 0 );
define Table3 as
        atomic( LogoRA border 20, 60 compound Table1
                table 2 with 700, 350 );
define RhoneA as
        atomic( Table3 compound Table2 touch, centered, south )
        border 1, 1 background lightGray
```

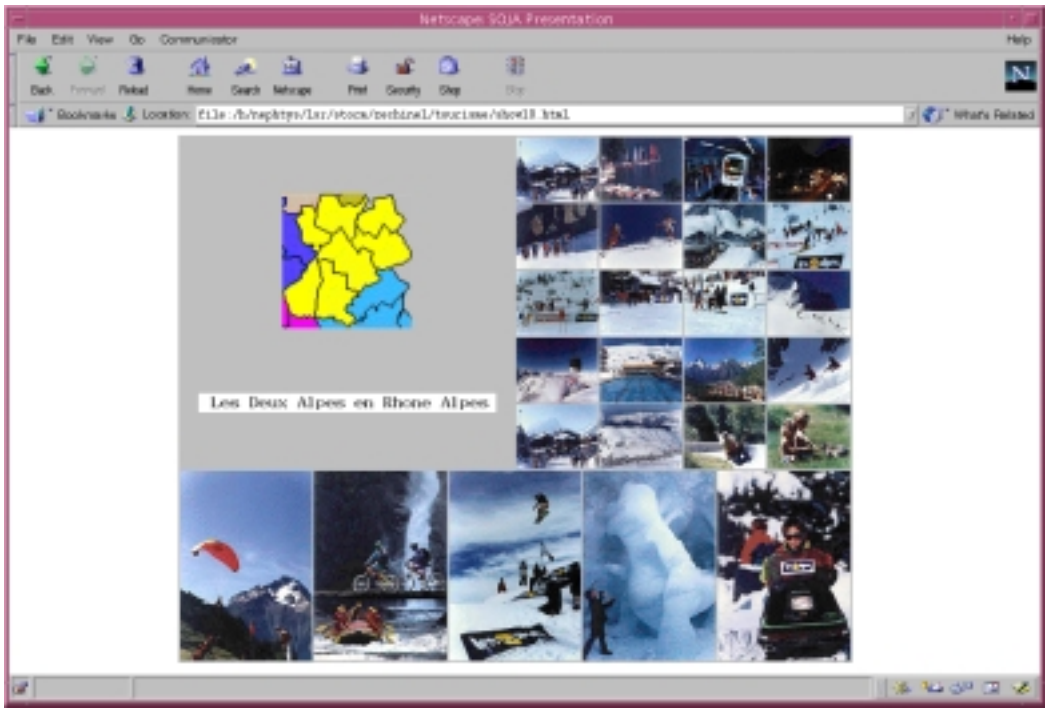


FIG. 6.10 – *Stations de ski et activités*

Notons que pour définir cette présentation, nous utilisons le constructeur directory. Il reçoit en entrée l'adresse URL d'un répertoire et retourne la liste des adresses des fichiers qu'il contient. Dans cette requête, nous récupérons un ensemble d'adresses URL correspondant aux images des stations contenues dans le répertoire "file://localhost/h" et "file://localhost/v".

Création de présentations enchaînées Des présentations peuvent être construites et enchaînées en utilisant des hyperliens. Afin d'illustrer ces aspects, considérons qu'un utilisateur de notre application touristique souhaite connaître les noms des régions voisines de l'Île de France.

Pour afficher ces informations, nous prenons l'Île de France comme référentiel et nous définissons un tableau contenant quatre hyperliens nommées NORTH, SOUTH, WEST, EAST (cf. annexe B). Chaque hyperlien point sur un tableau qui groupe les noms des régions en accord à leur position géographique par rapport l'Île de France. Finalement, chaque nom de région est un hyperlien vers une présentation qui montre de manière graphique la relation directionnelle entre l'Île de France et la région (cf. figure 6.11). L'expression OQLiST suivante définie le tableau d'hyperliens :

```
define Regions as
        String( Value = " Ile de France " ).setBackground( yellow ) fit 320, 50
        compound( select atomic( yellow compound e touch, centered, east )
                from e in list( String( Value = " NORTH " ) link North,
                                String( Value = " SOUTH " ) link South,
                                String( Value = " WEST " ) link West,
                                String( Value = " EAST " ) link East )
                table fit 4 fit 520, 38 arrange all meet + 20 on x )
        disjoint 80, centered, south at 0, 0
```

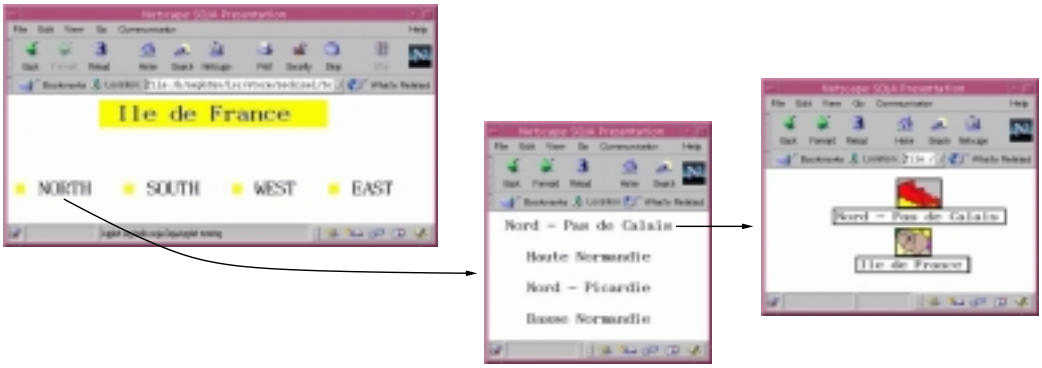


FIG. 6.11 – *Présentations enchaînées*

Chaque lien pointe vers une présentation qui correspond à une région placée au nord de l'Île de France. Pour retrouver les présentations correspondantes, nous interrogeons les relations spatio-temporelles de la description de la carte de France (cf. la définition détaillée dans l'annexe B).

6.4.2 Support à la présentation des données pour le décisionnel

Un entrepôt de données est une collection de données destinée à être exploitée par les applications d'aide à la décision [Inm92, JLVV00, BCA01]. Le contenu d'un entrepôt de données est le résultat de l'intégration de multiples sources ; ainsi, toutes les données nécessaires pour réaliser une analyse particulière se trouvent dans l'entrepôt. L'objectif des entrepôts de données est d'alimenter les systèmes d'aide à la décision en particulier ceux du traitement analytique en ligne OLAP (*OnLine Analytical Processing*). Les systèmes OLAP se caractérisent par une vision multidimensionnelle des données et leur analyse pour des interrogations interactives.

Les données sont perçues par l'utilisateur comme étant organisées en cubes dont les cellules contiennent des objets d'analyse ou mesures. Le cube est le concept central du modèle de données des systèmes multidimensionnels. Un cube organise les données en une ou plusieurs dimensions qui déterminent une mesure d'intérêt. Une dimension spécifie la manière dont on regarde les données pour les analyser, alors qu'une mesure est un objet d'analyse.

Gestion de données sur la météo Considérons un entrepôt de données contenant des informations concernant la météo en Europe : images des cartes des régions, températures, vitesse du vent, horaires du levé et des couches de soleil, qualité du temps (orageux, pluvieux, neigé), etc. Des applications d'analyse des tendances météorologiques sont couplées à cet entrepôt. Elles sont utilisées par différents organismes par exemple les offices de tourisme des régions pour décider de programmer la mise en place de moyens pour accueillir des touristes.

Nous supposons que les données de cet entrepôt sont organisées selon un *schéma en étoile* et que des collections des mesures et dimensions sont visualisées. La figure 6.12[5] présente le schéma de l'entrepôt de données météorologiques.

Dans le schéma en étoile, les mesures sont représentées par une relation de faits météo (Weather) et chaque dimension par une relation (Season, Time, Zone). La relation de faits référence les relations de dimension, en utilisant des identificateurs d'objets pour chacune d'elles, et stocke les valeurs de mesures pour la combinaison de ces identificateurs.

Les mesures d'intérêt dans notre application sont les températures moyennes dé-

5. Nous montrons le schéma des classes en anglais et les commentaires en français.

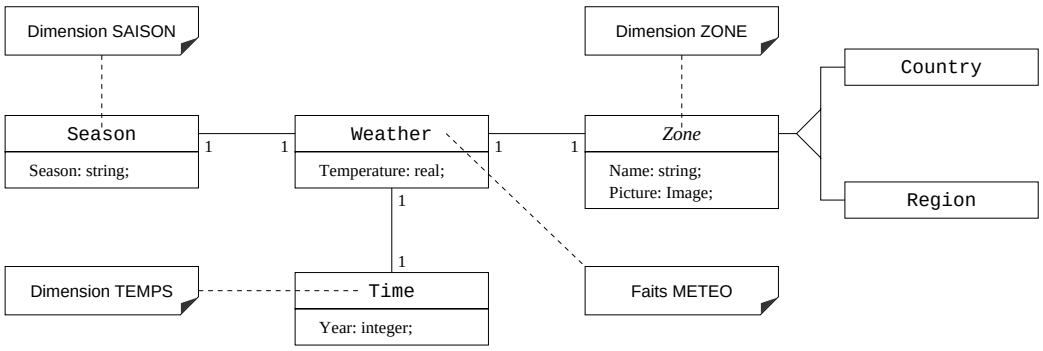


FIG. 6.12 – *Schéma en étoile des données météorologiques*

terminées au cours du temps par saison (hiver, printemps, été, automne) et par zone. La figure 6.13 présente le cube METEO qui est une collection d'objets de type Weather de trois régions de France. Il montre que la température moyenne dans la région Rhône Alpes en hiver 1999 était de 0C. Associée à la dimension ZONE, on trouve la hiérarchie Région → Pays. Une instance de cette hiérarchie peut regrouper par exemple les régions de Île de France, Bretagne et Rhône Alpes.

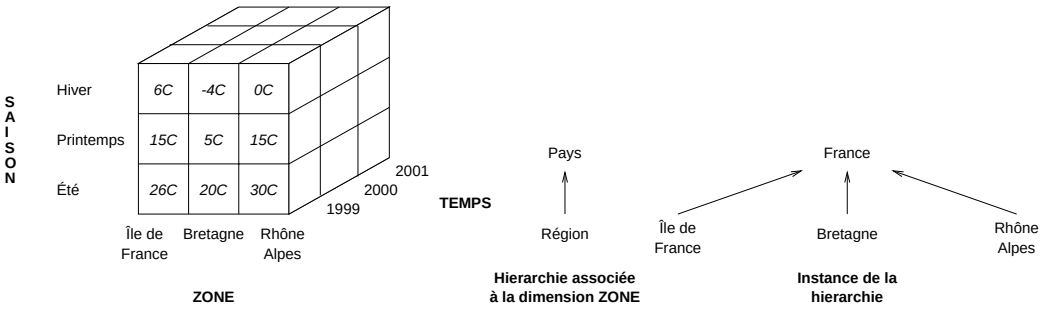


FIG. 6.13 – *Cube* METEO

L'analyse de l'entrepôt peut utiliser des techniques classiques issues des statistiques ou d'autres comme celle qui transforment les données en présentations graphiques. Nous avons spécifié un schéma de données et un contexte de présentation adaptés à la présentation des résultats de requêtes OLAP.

Schéma de présentations d'un cube

La figure 6.14 présente le schéma de présentations définies pour un cube. De manière générale, un cube est défini comme une liste de plans, chacun étant lui-même une liste de cellules.

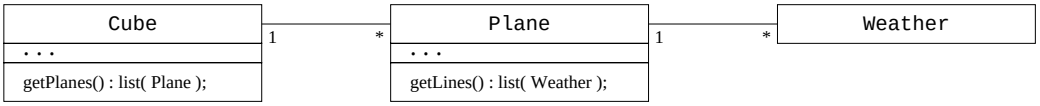


FIG. 6.14 – *Schéma de présentations d'un cube*

Présentation par défaut d'une cellule Une cellule représente une mesure par rapport à des dimensions du cube. Afin de pouvoir être affichée, elle doit être associée à une présentation. Dans le cas, du cube METEO, une cellule est associée à une présentation qui affiche l'image d'une zone avec la température moyenne au centre (cf. figure 6.15). L'expression OQLiST suivante définit le constructeur de cette présentation :

```
atomic( Weather.Zone.Picture border 1, 1 background black with 50, 50
        compound Integer( Value = Weather.Temperature ) fit 48, 20
        inside, centered, centered )
```

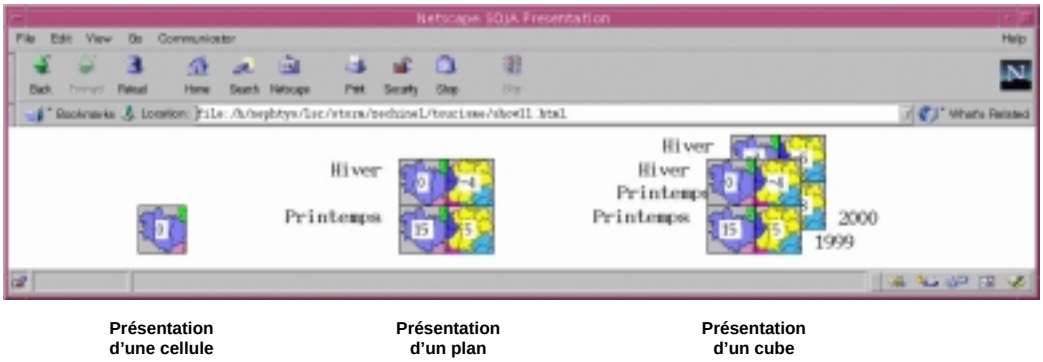


FIG. 6.15 – *Présentations des éléments d'un cube*

Présentation par défaut d'un plan Un plan est une liste de cellules à laquelle il faut aussi associer une présentation. Dans le cas du cube METEO, la présentation d'un plan est un tableau (une présentation composite) de cellules avec le nom de la saison à côté du coin supérieur gauche de la première cellule de chaque ligne du tableau. L'expression OQLiST suivante définit le constructeur de la présentation d'un plan :

```
select atomic( line[0].Season.Season
               compound ( line arrange all meet − 1 on x )
               disjoint, meet + 10, beginning, southeast )
from  line in Plane.getLines order by *
arrange all end on x
arrange all meet − 1 on y
```

Présentation par défaut d'un cube Elle spécifie la présentation d'une liste de plans. Dans notre exemple, la présentation d'un cube METEO affiche une collection de présentations de plans les uns devant les autres avec un décalage à droite de la moitié de la taille d'une cellule. L'expression OQLiST suivante définit le constructeur de la présentation d'un cube :

```
select atomic( plane.getLines[0][0].Time.Year
               compound plane
               disjoint, meet + 10, end, northwest at 0, 0 )
from  plane in Cube.getPlanes order by *
arrange all beginning + 25 on x
arrange all beginning − 25 on y at 0, 0 inverse
```

Interrogation du cube

L'interrogation vise la manipulation du cube selon différents niveaux de détail (selon le niveau d'agrégation des dimensions). Étant donnée la représentation d'un entrepôt sous la forme d'un cube, plusieurs opérations comme *slice-n-dice*, *roll-up*, *drill-down*, *pivot*, etc. ont été proposées [KEN95, PIL98, BCA01].

– *slice-n-dice* permet de restreindre les valeurs dans une ou plusieurs dimensions. Par exemple, présenter les températures moyennes en hiver et au printemps dans les régions Bretagne et Rhône Alpes en 1999 et 2000 :

```
select  M
from    METEO M
where (M.Zone.Name = "Bretagne" or M.Zone.Name = "Rhône - Alpes")
      and (M.Season.Season = "Hiver" or M.Season.Season = "Printemps")
      and (M.Time.Year = 1999 or M.Time.Year = 2000)
```

Du point de vue de la présentation des résultats, ceci revient à afficher des plans en appliquant un filtre de sélection sur les lignes et les colonnes des plans de la présentation du cube METEO. La spécification suivante présente les plans du cube dans une seule colonne avec une séparation de 20 pixels (cf. figure 6.16) :

```
Cube( select  M
      from   METEO M
      where (M.Zone.Name = "Bretagne" or M.Zone.Name = "Rhône - Alpes")
            and (M.Season.Season = "Hiver" or M.Season.Season = "Printemps")
            and (M.Time.Year = 1999 or M.Time.Year = 2000)
    )
table 1 arrange all meet + 20 on y
```

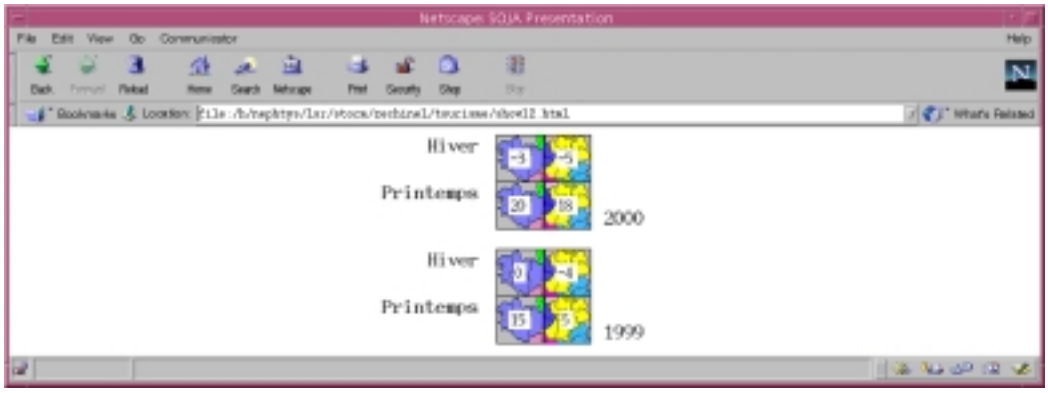


FIG. 6.16 – *Présentations des plans d'un cube*

– *roll-up* effectue l'agrégation des mesures d'un niveau particulier vers un niveau plus général des hiérarchies associées à une dimension. Par exemple, présenter la moyenne des températures en France en hiver 2000 sur une image de la carte de France (cf. figure 6.17) :

```
Cube( set( Weather(
      Season = Season( Season = "Hiver" ),
      Time = Time( Year = 2000 ),
      Zone = Country( Name = "France",
                      Picture = Image(URL = "file://localhost/FrenchMap.gif") )
      Temperature = avg( select M.Temperature from METEO M
                         where M.Season.Season = "Hiver"
                               and M.Time.Year = 2000 ) ) ) )
```

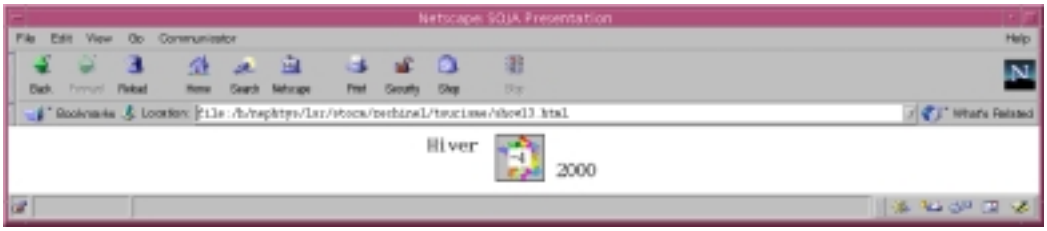


FIG. 6.17 – *Opération roll-up*

– *drill-down* fait l'opération inverse.

– *pivot* inverse les colonnes et les lignes des plans du cube. Cet opérateur active un contexte de présentation où la présentation par défaut d'un plan est définie comme suit (cf. figure 6.18) :

```
select atomic( line[0].Season.Season with 50, 14
               compound ( line arrange all meet – 1 on y )
               touch, centered, south )
from  line in Plane.getLines order by *
arrange all meet – 1 on x
arrange all end on y
```

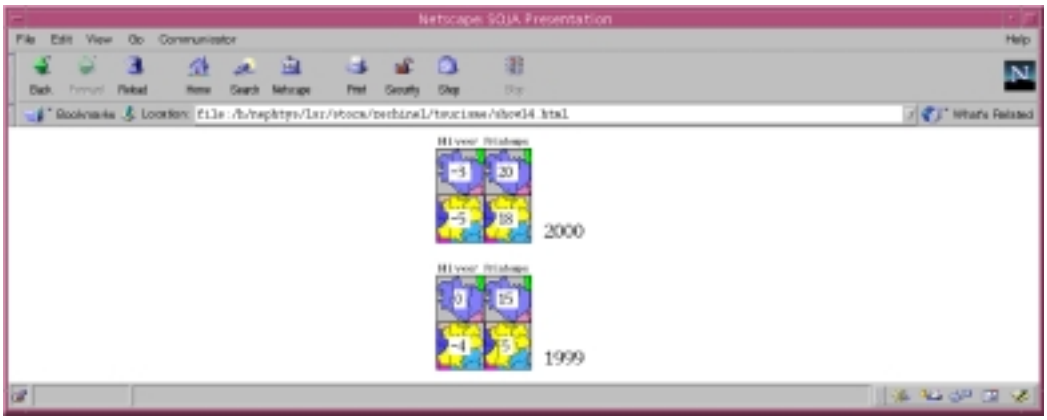


FIG. 6.18 – *Opération pivot*

6.5 Conclusion

Nous avons décrit les expérimentations que nous avons menées afin de valider l'utilisation du langage OQLiST et de notre gestionnaire de présentations. Le ges-

tionnaire est un outil qui fournit un support pour la spécification de présentations et l'intégration de données provenant des sources hétérogènes. Nous avons détaillé notre démarche de construction du gestionnaire qui est adapté pour exécuter des présentations spatio-temporelles dans des plates-formes JMF et SMIL.

En ce qui concerne OQLiST, nous avons montré de manière pragmatique son pouvoir d'expression pour la spécification de configurations spatiales, temporelles et spatio-temporelles d'objets. Les objets de type différents sont traités et combinés de manière homogène avec ce langage sans devoir se soucier de leur localisation, de leur format d'expression, ni de leur type. Les opérateurs liés à l'interaction et au décor des présentations permettent de traiter des aspects présents dans des documents spécifiés par des langages standards tels que SMIL.

Dans l'implantation de notre prototype, nous avons utilisé des stratégies de gestion de cache pour permettre le partage et la réutilisation de données dans un gestionnaire. D'autres stratégies telles que celles proposées dans [CRS99] doivent être étudiées.

Les contextes de présentation nous permettent de définir des outils pour la visualisation. Par exemple nous avons choisi une visualisation en forme de cube, mais il est possible de spécifier des contextes de présentation définissant d'autres présentations (par ex. en utilisant des graphiques ou des métaphores [Man99]). Une fois qu'une présentation par défaut est construite, OQLiST permet de la manipuler (par ex. les plans du cube sont présentés dans un tableau).

Par ailleurs, concernant le gestionnaire de présentations que nous avons spécifié et implanté, nous remarquons que la valeur ajoutée de cet outil est sa capacité de traiter des contextes de présentation de manière transparente. Les applications qui l'utilisent peuvent changer de support d'exécution de présentations dynamiquement.

Chapitre 7

Conclusions et perspectives

Aujourd'hui la mise à disposition de données classiques et multimédias et des outils (plates-formes, langages de spécification) pour les exploiter encourage le partage des informations. La diversité des modèles, des prototypes et des standards adaptés pour définir et exécuter des présentations multimédias est une réalité. Elle met en évidence le besoin de proposer des infrastructures de médiation entre les données et les applications. Pour pouvoir répondre à un tel besoin il faut tenir compte des caractéristiques des données (volumineuses, multiforme, multimédias, évolutives, mobiles, etc.) dans les modèles de description, de proposer des langages et de spécifier des mécanismes facilitant le développement des applications utilisant des données multimédias et multiformes.

Partant de ces constatations, nous avons proposé JAGUAR une infrastructure logicielle pour la spécification de gestionnaires de présentations de données. Les gestionnaires servent de couche médiatrice entre des sources de données hétérogènes et réparties et des applications qui ont besoin d'accéder et visualiser des données.

7.1 Bilan du travail effectué

7.1.1 Infrastructure de spécification des gestionnaires de présentations

JAGUAR est une infrastructure logicielle pour la spécification de gestionnaires de présentations. Un gestionnaire est un médiateur entre des applications et des sources de données. Il implante un modèle pivot et un langage qui lui permettent d'intégrer des données hétérogènes dans des présentations spatiales, temporelles et

spatio-temporelles. Les types de données gérés par un gestionnaire sont décrits dans un schéma qui peut être associé à un ou plusieurs contextes de présentation chacun spécifiant des présentations par défaut pour chaque type de donnée du schéma.

Les applications interagissent avec le gestionnaire au travers d'adaptateurs. Un adaptateur traduit la spécification d'une présentation exprimée dans un langage *ad-hoc* vers une expression du langage du gestionnaire (OQLiST). Une fois construite, le gestionnaire envoi le résultat à l'adaptateur qui la met en forme dans une représentation exécutable par la plate-forme d'exécution de l'application.

Les gestionnaires de présentations sont spécifiés auprès de JAGUAR. Pour cela, un programmeur doit définir un schéma de données, au moins un contexte de présentation en spécifiant les interactions entre les applications et les extracteurs d'un ensemble de sources. JAGUAR génère une interface applicative (API) et les librairies de classes implantant les adaptateurs et la communication avec un gestionnaire de présentations. Ce support est utilisé par les applications pour faire gérer leurs présentations.

7.1.2 Modèle spatio-temporel de présentations

Nous avons proposé un modèle de présentations qui permet de traiter de manière homogène la présentation de médias et des données classiques. Nous avons adopté une approche par intervalles pour représenter les projections d'un objet sur les dimensions d'un espace (i.e., 1D, 2D, 3D, 4D). Nous avons utilisé le concept d'ombre qui représente la projection d'un objet sur un axe i par deux intervalles décrivant respectivement le position de l'objet et sa taille sur i. De manière générale, la présentation d'un objet est alors représentée par une combinaison de n intervalles, où n est le nombre de dimensions de l'espace.

Nous avons étudié de manière approfondie les présentations des objets dans un espace temporel, spatial et spatio-temporel. Les présentations des objets sont atomiques et composites. Une présentation atomique décrit la façon de visualiser un objet (position absolue, taille, durée). Une présentation composite décrit la façon dont un ensemble de présentations sont ordonnées dans l'espace et/ou dans le temps. L'ordonnancement est représenté par des relations spatiales et temporelles définies par des relations d'intervalles.

Les études sur les relations spatiales et temporelles montrent qu'il n'y a pas une sémantique standard pour les relations directionnelles [Ege91, PT97, TPSS98]. Au lieu de fixer une seule spécification pour ces relations, notre modèle adopte l'approche

de [Her94, TPSS98] pour préciser leur sémantique. Notre modèle caractérise ainsi un espace "exhaustif" de relations directionnelles pour représenter ces configurations. Enfin, le modèle caractérise également des relations d'alignement (au centre, au milieu, à gauche, à droite, en bas, en haut) à la façon des outils de dessin.

Enfin, la notion de présentation composite permet de modéliser la composition inter-médias de présentations mais aussi des descriptions intra-médias. En effet, un média peut être décrit par une présentation composite en spécifiant la position et les relations des objets (régions) contenus dans le média.

7.1.3 Le langage OQLiST

OQLiST, le langage de construction et d'interrogation de présentations de notre modèle spatio-temporel, fournit une palette d'opérateurs pour :

- construire des présentations atomiques spatio-temporelles d'objets médias (i.e., texte, image, son, vidéo, document) et de valeurs (i.e., entier, réel, chaîne de caractères) ;
- spécifier la position absolue et la taille d'une présentation (at, fit, with, during) ;
- composer des présentations (compound) avec des opérateurs spatiaux (topologiques et directionnels) et/ou temporels combinés avec des relations d'alignement (beginning, centered, end, middle, meet) et d'ordre (inverse, move, swap) ;
- associer des attributs comme l'arrière plan (background), le cadre (border), un lien (button, link) ;
- associer à un objet la description de son contenu en utilisant une présentation (l'opérateur tilde, descrip).

OQLiST permet également d'interroger et d'associer explicitement une présentation au résultat. Par exemple les collection sont présentées en utilisant des configurations spatiales et temporelles particulières : table et arrange all.

7.1.4 Implantation et expérimentation

Nous avons implanté une infrastructure prototype adaptée pour gérer des données pour des applications utilisant des plates-formes JMF (*Java Media Framework*)

et SMIL exécutables sur la Web. Notre prototype a été implanté en Java et il utilise un serveur de présentations multimédias spatio-temporelles basé sur O_2 comme support de stockage. Le prototype fait également appel à des outils tels que des générateurs, des analyseurs lexicaux et syntaxiques, des librairies de traitement de médias, des navigateurs Web. La version actuelle est en mesure d'intégrer dans une présentation multimédia des objets provenant de sources hétérogènes réparties (par ex. BD relationnelles, BD à objets, serveurs d'images, de sons, et des vidéos).

Dans un premier temps, nous avons utilisé notre prototype pour construire une application touristique qui a besoin de présenter des informations concernant des sites en France. Cette application nous a permis de valider l'utilisation de notre approche pour construire des présentations en accédant à des données disponibles sur l'Internet. Dans un deuxième temps, nous avons implanté un support de visualisation pour les données d'un entrepôt basés sur une approche OLAP. Pour cela, nous avons implanté un contexte de présentation adapté pour la présentation d'un cube de données météorologiques. Des opérations propres au cube ont été utilisées pour montrer la visualisation de ces données. À travers la construction de ces deux applications, nous avons montré de manière pragmatique l'expressivité de notre langage OQLiST.

7.1.5 Contributions majeures

Notre travail est centré sur la spécification d'une infrastructure logicielle permettant l'intégration de données hétérogènes dans des présentations spatio-temporelles. L'originalité de notre travail consiste à avoir abordé la gestion de présentations multimédias d'un point de vue modèle, langage et support logiciel. En effet, la plupart de technologies existantes se focalisent sur un aspect à la fois (langage, modèle, plateforme d'exécution) ce qui fait que les applications doivent mettre en œuvre des solutions *ad-hoc* pour les autres aspects.

Nous souhaitons souligner certains aspects abordés dans cette thèse qui nous paraissent être des contributions importantes :

1. Notre approche prend en compte les aspects spatiaux et temporels avec un modèle assez général pour qu'il puisse servir de modèle pivot aux langages et standards existants.

2. La spécification et la construction d'un médiateur qui sépare la gestion (spécification, réutilisation et stockage) de présentations de données et qui fournit un

support pour fédérer des bases et des applications.

Ces points constituent, nous semble-t-il, un travail original par rapport aux travaux existants concernant la multimédia.

7.2 Perspectives

7.2.1 Modèle

La présence des informations indéterminées est courante dans les présentations multimédias. Par exemple avoir un délai ou une durée indéterminées représentant le fait qu'un événement (par ex. un clic de souris) pourra déclencher ou arrêter une présentation. Nous nous sommes intéressés dans la modélisation des présentations sans considérer des aspects indéterminés. Il sera donc intéressant et utile d'étendre notre modèle pour représenter des configurations imprécises, mais aussi pour intégrer des aspects interactifs des présentations.

L'introduction de valeurs indéterminées dans le modèle aura des impacts sur l'interrogation et la réutilisation des présentations. Il faudra spécifier des stratégies pour interroger et interpréter de telles valeurs.

Dans notre travail nous avons réalisé une étude approfondie des relations spatiales et temporelles en les définissant en termes de relations d'intervalles pour des espaces spatial, temporel et spatio-temporel. Cette étude pourrait être poursuivie pour définir des relations spatiales pour un espace 3D. Ceci permettra de profiter des plates-formes telles que VRML (*Virtual Reality Modeling Language*) [CBM97] pour la visualisation de présentations.

7.2.2 Langage

Dans notre travail nous n'avons pas abordé des problèmes d'optimisation de requêtes. Une étude devra être menée pour déterminer comment optimiser des expressions OQLiST. Cette étude devra considérer les caractéristiques des opérateurs, les structures de requêtes mais aussi les sources (par ex. disponibilité, coût d'évaluation des requêtes, coût de communication). Bien évidemment, les travaux concernant l'optimisation d'OQL comme [BK89, CD92, Fin92] seront des références à prendre en compte.

Enfin, l'extension de notre modèle pour traiter des valeurs indéterminées [Con99] et pour représenter des présentations 4D aura des impacts sur OQLiST qui méritent une étude sérieuse.

7.2.3 Indexation

Proposer des structures pour supporter l'accès efficace des données spatiales et temporelles est une préoccupation de plusieurs travaux [BOSD+97a, BOSD+97b, GG98]. Le caractère volumineux des médias et leurs besoins des qualités de services précis montrent la nécessité d'adapter et de proposer des techniques d'indexation adaptées.

Nous pensons que nous pourrons utiliser les positions absolues mais aussi la structure des présentations définie par des relations pour organiser les données sur des supports physiques. Les travaux de [BOSD+97c, PS94, PD97, TPSS98, PKA99] nous semblent un bon départ pour la proposition d'index adaptés aux présentations.

7.2.4 Vers des services multimédias adaptables

Les dernières décades ont vu l'émergence et l'expansion de systèmes et d'applications manipulant des données fortement structurées, semi-structurées ou non structurées sous divers formats (digitalisées, multimédias). En particulier, les médias (texte, image, son, vidéo) par leur caractère volumineux, hétérogène et leurs besoins spécifiques de qualités de services pour les exploiter sont au centre des préoccupations de plusieurs domaines (réseaux, système, visualisation, bases de données).

Les systèmes construits autour de ces données doivent souvent réaliser des opérations parfois complexes nécessitant de nombreuses interactions entre les utilisateurs et les sources de données dispersées sur le réseau. Les modèles et les algorithmes aujourd'hui disponibles pour la représentation des données en mémoire ou sur disque, l'accès aux données, la gestion des index, etc. doivent donc être reconsidérés [Col00].

La gestion de données multimédias, en particulier le stockage et l'interrogation sont des fonctionnalités propres au SGBDM. Le nouveau scénario d'outils logiciels de plus en plus hétérogène avec des besoins d'intégration et de réutilisation met en cause son architecture. Il est nécessaire de repenser l'ensemble de l'architecture d'un SGBD et de ses composants en fonction des avancées en matière d'architecture matérielle (disque RAID), de systèmes d'exploitation (noyau adaptable, adressage 64 bits), de réseaux (haut débit, qualité de service), d'architecture parallèle (machine SMP),des

outils matériels petits, par exemple le PDA (*Personal Digital Assistant*).

Les services spécifiques aux objets multimédias sont nécessairement reliés à d'autres comme la persistance ou la réplication. De même, les aspects transactionnels doivent être revus dans un cadre différent de celui des modèles de transactions proposés. Ainsi les objets en 3D ou certains aspects spécifiques liés à la nature temps réel des types multimédias comme le son ou la vidéo induisent des problèmes nouveaux en matière de transactions qui ne sont pas uniquement réductibles à ceux de qualité de service lors de la restitution d'un flux de données.

Bibliographie

[AC93] M. Adiba et C. Collet. *Objets et Bases de Données : le SGBD O_2*. Hermès, 1993.

[Adi96] M. Adiba. STORM: an object-oriented multimedia DBMS. Dans K. Nowsu, B. Thuraisingham et B. Berra, éditeurs, *Multimedia Database Management Systems. Design and Implementation Strategies*, chapitre 3. Kluwer Academic Publishers, 1996.

[AH91] D.P. Anderson et G. Homsy. A continuous media I/O server and its synchronization mechanism. *IEEE Computer*, 24(10), 1991.

[All83] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 1983.

[ALMM97] M. Adiba, R. Lozano, H. Martin et F. Mocellin. Management of multimedia data using an object-oriented database system. Dans *Proc. of DEXA'97 Workshop on Database and Expert Systems Applications*, September 1997.

[ASS99] S. Adali, M.L. Sapino et V.S. Subrahmanian. A multimedia presentation algebra. Dans *Proc. of the ACM SIGMOD International Conference on Management of Data*, 1999.

[AZ99a] M. Adiba et J.L. Zechinelli-Martini. Managing spatio-temporal multimedia presentations as database objects. Rapport de recherche RR 1022-I-LSR 10, LSR - IMAG - Université Joseph Fourier, Grenoble - France, July 1999.

[AZ99b] M. Adiba et J.L. Zechinelli-Martini. Spatio-temporal multimedia presentations as database objects. Dans *Proc. of DEXA'99, 10th Interna-*

tional Conference on Databases and Expert Systems Applications, Florence - Italy, August - September 1999.

[AZ00] M. Adiba et J.L. Zechinelli-Martini. JAGUAR: integrating and managing multimedia presentations by a web-based object server. Dans *the 14th European Conference on Object-Oriented Programming (ECOOP 2000), Demonstrations Session*, Sophia Antipolis and Cannes - France, June 2000.

[BBE99] A. Bouguettaya, B. Benatallah et A. Elmagarmid. An overview of multidatabase systems: Past and present. Dans A. Elmagarmid, M. Rusinkiewick et A. Sheth, éditeurs, *Management of Heterogeneous and Autonomous Database Systems*, The Morgan Kaufmann Series in Data Management Systems, chapitre 1. Morgan-Kaufmann, 1999.

[BCA01] E. Benitez-Guerrero, C. Collet et M. Adiba. Entrepôts de données : caractéristiques et problématique. *Technique et science informatiques*, 20(2), 2001.

[BCD93] G.S. Blair, G. Coulson et N. Davies. System support for multimedia applications: An assessment of the state of the art. Rapport de recherche MPG-93-29, Lancaster University, Bailrigg, Lancaster, LA1 4YR, U.K., 1993.

[BCD+99] L. Bouganim, T. Chan-Sine-Ying, T.T. Dang-Ngoc, J.L. Darroux, G. Gardarin et F. Sha. Miro web: Integrating multiple data sources through semistructured data types. Dans *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*. Morgan Kaufmann, 1999.

[BDK92] F. Bancilhon, C. Delobel et P. Kanellakis. *Building an Object-Oriented Database System: The story of O_2*. Morgan Kaufmann, 1992.

[BF98] E. Bertino et E. Ferrari. Temporal synchronization models for multimedia data. *IEEE Transactions on Knowledge and Data Engineering*, 10(4), 1998.

[BK89] E. Bertino et W. Kim. Indexing techniques for queries on nested objects. *IEEE Transactions on Knowledge and Data Enginnering*, 1(2), 1989.

[BOSD+97a] E. Bertino, B.C. Ooi, R. Sacks-Davis, K. Tan, J. Zobel, B. Shidlovsky et B. Catania. *Indexing Techniques for Advanced Database Systems*, chapitre Spatial Databases. Kluwer Academic Publishers, 1997.

[BOSD+97b] E. Bertino, B.C. Ooi, R. Sacks-Davis, K. Tan, J. Zobel, B. Shidlovsky et B. Catania. *Indexing Techniques for Advanced Database Systems*, chapitre Temporal Databases. Kluwer Academic Publishers, 1997.

[BOSD+97c] E. Bertino, B.C. Ooi, R. Sacks-Davis, K. Tan, J. Zobel, B. Shidlovsky et B. Catania. *Indexing Techniques for Advanced Database Systems*, chapitre Image Databases. Kluwer Academic Publishers, 1997.

[Bre99] M. Brelot. *Représentations orientées-objets de scènes visuelles pour la composition flexible*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble - France, mai 1999.

[CBB+00] R.G.G. Cattell, D. Barry, M. Berler, J. Eastman, D. Jordan, C. Russell, O. Schadow, T. Stanienda et F. Velez, éditeurs. *The Object Database Standard: ODMG 3.0*. Morgan Kaufmann, 2000.

[CBM97] R. Carey, G. Bell et C. Marrin. ISO/IEC 14772-1:1997 Virtual Reality Modeling Language (VRML97), 1997. http://www.vrml.org/Specifications/VRML97.

[CC96] R. Cutler et K.S. Candan. Multimedia authoring systems. Dans V.S. Subrahmanian et Sushil Jajodia, éditeurs, *Multimedia Database Systems. Issues and Research Directions*. Springer-Verlag, 1996.

[CD92] S. Cluet et C. Delobel. A general framework for the optimization of object-oriented queries. Dans *Proc. of the ACM SIGMOD International Conference on Management of Data*, June 1992.

[CD97] S. Chaudhuri et U. Dayal. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record*, 26(1), March 1997.

[CG99] B. Czejdo et L. Gruenwald. Schema and language translation. Dans A. Elmagarmid, M. Rusinkiewick et A. Sheth, éditeurs, *Management of Heterogeneous and Autonomous Database Systems*, The Morgan Kaufmann Series in Data Management Systems, chapitre 6. Morgan-Kaufmann, 1999.

[Clu98] S. Cluet. Designing OQL: Allowing objects to be queried. *Information Systems*, 23(5), 1998.

[Cod93] E. Codd. Providing OLAP (On-Line Analytical Processing) to Users-Analysts: An IT Mandate. Rapport de recherche, E.F. Codd and Associates, 1993.

[Col00] C. Collet. The NODS Project: Networked Open Database Services. Dans *Proc. of the 14th European Conference on Object-Oriented Programming (ECOOP 2000) - Symposium on Objects and Databases*, Cannes, France, Juin 2000. http://www-lsr.imag.fr/Les.Groupes/storm/NODS/index.html.

[Con99] F. Connan. *Interrogation flexible de bases de données multimédias*. PhD thesis, Université de Rennes 1, Rennes - France, décembre 1999.

[Cri] J. Cristy. ImageMagick. http://www.imagemagick.org.

[CRS99] B. Chidlovskii, C. Roncancio et M-L. Schneider. Semantic cache mechanism for heterogeneous web querying. *Computer Networks*, 31, 1999.

[DB98] L. De-León-Fernández et D. Burgos. *Macromedia Director 6.x Iniciación y referencia*. McGraw-Hill, 1998.

[DK95] A. Duda et C. Keramane. Structured temporal composition of multimedia data. Dans *Proc. of the IEEE International Workshop on Multi-Media Data Base Management Systems*, August 1995.

[Dum00] M. Dumas. *TEMPOS : une plate-forme pour le développement d'applications temporelles au dessus de SGBD à objets*. PhD thesis, Université Joseph Fourier, Grenoble - France, juin 2000.

[EF91] M. Egenhofer et R. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2), 1991.

[Ege91] M. Egenhofer. Reasoning about binary topological relations. Dans *Proc. of the 2st International Symposium on Large Spatial Databases (SSD), Springer Verlag LNCS*, 1991.

[Ege94] M. J. Egenhofer. Spatial SQL: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, 6(1), 1994.

[EJH+97] A.K. Elmagarmid, H. Jiang, A.A. Helal, A. Joshi et M. Ahmed. *Video Database Systems: Issues, Products, and Applications*. Kluwer Academic Publishers, 1997.

[EMM+98] M. Echiffre, C. Marchisio, P. Marchisio, P. Panicciari et S. Del Rossi. MHEG-5—Aims, concepts, and implementation issues. *IEEE MultiMedia*, 5(1), 1998.

[Fin92] B. Finance. *Une plate-forme pour la génération d'optimiseurs extensibles*. PhD thesis, Université Paris VI, Paris - France, feb 1992.

[FLMM98] J. Freire, R. Lozano, H. Martin et F. Mocellin. A STORM environment for building multimedia presentations. Dans *Proc. of the 12th International Conference on Information Networking*, January 1998.

[Fou] H. Foucher. SOJA. http://www.helio.org/products/smil.

[FSA+95] M. Flickner, H.S. Sawhney, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele et P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9), 1995.

[GBE+00] R.H. Güting, M.H. Böhlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider et M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, 25(1), 2000.

[GG98] V. Gaede et O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2), 1998.

[GQV98] R. Guetari, V. Quint et I. Vatton. Amaya: an authoring tool for the Web. Dans *Proc. of MCSEAI'98, 5th Maghrebian Conference on Computer Sciences*, Tunis, December 1998.

[Gro94] W. Grosky. Multimedia information systems. *IEEE MultiMedia Magazine*, 1(1), 1994.

[GT99] R. Gordon et S. Talley. *Essential JMF: Java Media Framework*. Essential. Prentice Hall PTR, 1999.

[Gut84] A. Guttman. R-trees: A dinamic index structure for spatial searching. Dans *Proc. of the ACM SIGMOD International Conference on Management of Data*, 1984.

[Her94] D. Hernández. *Qualitative Representation of Spatial Knowledge*, volume 804 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1994.

[HMS96] R. Hjelsvold, R. Midtstraum et O. Sandstå. Searching and browsing a shared video database. Dans K.Nowsu, B. Thuraisingham et B. Berra, éditeurs, *Multimedia Database Management Systems. Design and Implementation Strategies*, chapitre 4. Kluwer Academic Publishers, 1996.

[HS95] E. Hwang et V.S. Subrahmanian. Querying video libraries. Rapport de recherche CS-TR-3482, University of Maryland, College Park, MD 20742, March 1995. disponible à http://melvil.cs.umd.edu.

[Inm92] W. Inmon. *Building the Data Warehouse*. QED Technical Publishing Group, Wellesley, Massachusetts, U.S.A., 1992.

[ISO86] Information processing - Text and office systems - Standard Generalized Markup Language: SGML (ISO 8879). International Standards Organization, 1986.

[ISO94] Information processing systems - Computer graphics and image processing - Presentation Environment for Multimedia Objects (PREMO). Working Draft ISO/IEC JTC 1/SC 24 WG 6 N 032, International Standards Organization, 1994.

[ISO97] Information technology - Hypermedia/Time-based structuring language: HyTime (ISO/IEC 10744). International Standards Organization, 1997.

[Jag96] H.V. Jagadish. Indexing for retrieval by similarity. Dans V.S. Subrahmanian et Sushil Jajodia, éditeurs, *Multimedia Database Systems. Issues and Research Directions*. Springer-Verlag, 1996.

[JLVV00] M. Jarke, M. Lenzerini, Y. Vassiliou et P. Vassiliadis. *Fundamentals of Data Warehouses*. Springer Verlag, 2000.

[JME99] H. Jiang, D. Montesi et A. K. Elmagarmid. Integrated Video and Text for Content-based Acces to Video Databases. *Multimedia Tools and Applications*, 9(3), 1999.

[KEN95] Kenan Systems Corporation. *An Introduction to Multidimensional Database Technology*, 1995. White paper.

[Ker97] C. Keramane. *Spécification de présentations multimédia structurées.* PhD thesis, Institut National Polytechnique de Grenoble, Grenoble - France, novembre 1997.

[KR98] S. Kaushik et E.A. Rundensteiner. SVIQUEL: A spatial visual query and exploration language. Dans *Proc. of the 9th Int. Conf. on Database and Expert Systems Applications(DEXA'98)*, August 1998.

[KVS97] I. Kostalas, M. Vazirgiannis et T. Sellis. Spatiotemporal specification and verification for multimedia scenarios. Dans M. Erwig, R.H. Güting, M. Shneider et M. Vazirgiannis, éditeurs, *Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases*, number CH-97-08 in CHOROCHRONOS Technical Report Series, chapitre 3. http://www.dbnet.ece.ntua.gr/∼choros, December 1997.

[Lay97] N. Layaïda. *Madeus: système d'édition et de présentation de documents structurés multimédia.* PhD thesis, Université Joseph Fourier, Grenoble - France, juin 1997.

[Li98] J.Z. Li. Modeling and querying multimedia data. Rapport de recherche TR 98-05, Laboratory for Database Systems Research - Department of Computing Science - University of Alberta, Edmonton - Alberta - Canada, March 1998.

[LM98] R. Lozano et H. Martin. Querying virtual videos using path and temporal expressions. Dans *Proc. of the ACM Symposium on Applied Computing Multimedia Systems Track*, February 1998.

[LOS96a] J.Z. Li, M.T. Özsu et D. Szafron. Modeling of moving objects in a video objectbase management system. Rapport de recherche TR 96-12, Laboratory for Database Systems Research - Department of Computing

Science - University of Alberta, Edmonton - Alberta - Canada, April 1996.

[LOS96b] J.Z. Li, M.T. Özsu et D. Szafron. Spatial reasoning rules in multimedia management system. Dans *Proc. of the International Conference on Multimedia Modeling*, November 1996.

[LOSO97] J.Z. Li, M.T. Özsu, D. Szafron et V. Oria. MOQL: A multimedia object query language. Dans *Proc. of the Third International Workshop on Multimedia Information Systems*, September 1997.

[Loz00] R. Lozano-Espinosa. *Intégration de données vidéo dans un SGBD à objets.* PhD thesis, Université Joseph Fourier, Grenoble - France, avril 2000.

[Man99] T.M. Mann. Visualization of WWW-search results. Dans *Proc. of DEXA'99 Workshop on Database and Expert Systems Applications*, September 1999.

[Mar96a] S. Marcus. Querying multimedia databases in SQL. Dans V.S. Subrahmanian et Sushil Jajodia, éditeurs, *Multimedia Database Systems. Issues and Research Directions.* Springer-Verlag, 1996.

[Mar96b] J. Martinez. The design of an extensible multimedia library for an OODBMS. Dans *Proc. of DEXA'96 Workshop on Database and Expert Systems Applications*, September 1996.

[Mar97] H. Martin. Specification of intentional multimedia presentations using an object-oriented database. Dans *Proc. of the International Symposium on Digital Media Information Base*, November 1997.

[Mar00] H. Martin. Systèmes d'information multimédias : des systèmes aux applications. Rapport de recherche, LSR - IMAG - Université Joseph Fourier, Grenoble - France, août 2000. Mémoire d'habilitation à diriger les recherches.

[Moc97] F. Mocellin. *Gestion de données et de présentations multimédias par un SGBD à objets.* PhD thesis, Université Joseph Fourier, Grenoble - France, decembre 1997.

[MRJ99] P. Missier, M. Rusinkiewicz et W. Jin. Multidatabase languages. Dans A. Elmagarmid, M. Rusinkiewick et A. Sheth, éditeurs, *Management of Heterogeneous and Autonomous Database Systems*, The Morgan Kaufmann Series in Data Management Systems, chapitre 7. Morgan-Kaufmann, 1999.

[MS96] S. Marcus et V.S. Subrahmanian. Towards a theory of multimedia database systems. Dans V.S. Subrahmanian et Sushil Jajodia, éditeurs, *Multimedia Database Systems. Issues and Research Directions.* Springer-Verlag, 1996.

[NL99a] F. Nack et A.T. Lindsay. Everything you wanted to know about MPEG-7 part 1. *IEEE MultiMedia Magazine*, 6(3), 1999.

[NL99b] F. Nack et A.T. Lindsay. Everything you wanted to know about MPEG-7 part 2. *IEEE MultiMedia Magazine*, 6(4), 1999.

[ORS96] B. Ozden, R. Rastogi et A. Silberschatz. The storage and retrieval of continuous media data. Dans V.S. Subrahmanian et Sushil Jajodia, éditeurs, *Multimedia Database Systems. Issues and Research Directions.* Springer-Verlag, 1996.

[OS95] V.E. Ogle et M. Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer*, 28(9), 1995.

[OSEV95] M.T. Özsu, D. Szafron, G. El-Medani et C. Vittal. An object-oriented multimedia database system for a news-on-demand application. *ACM Multimedia Systems*, 3, 1995.

[OTCH98] B.C. Ooi, K. Tan, T.S. Chua et W. Hsu. Fast image retrieval using color-spatial information. *The VLDB Journal*, 0(7), 1998.

[PD97] D. Papadias et B. Delis. Relation-based similarity. Dans *Proc. of the 5th ACM Workshop on GIS*, 1997.

[PIL98] Pilot Software. *An Introduction to OLAP: Multidimensional Terminology and Technology*, 1998. White paper.

[PKA99] D. Papadias, N. Karacapilidis et D. Arkoumanis. Processing fuzzy spatial queries: A configuration similarity approach. *International Journal of Geographic Information Science*, 13(2), 1999.

[PS94] D. Papadias et T. Sellis. On the qualitative representation of spatial knowledge in 2D space. *Very Large Data Bases Journal, Special Issue on Spatial Databases*, 3(4), 1994.

[PT97] D. Papadias et Y. Theodoridis. Spatial relations, minimum bounding rectangles, and spatial data structures. *International Journal of Geographic Information Science*, 11(2), 1997.

[PTS97] D. Papadias, Y. Theodoridis et E. Stefanakis. Multi-dimensional range query processing with spatial relations. *Geographical Systems*, 4(4), 1997.

[PTSE95] D. Papadias, Y. Theodoridis, T. Sellis et M. Egenhofer. Topological relations in the world of minimum bounding rectangles: A study with R-trees. Dans *Proc. of the ACM Conference on the Modelling of Data (SIGMOD)*, May 1995.

[QV97] V. Quint et I. Vatton. An introduction to Amaya. *World Wide Web Journal*, 2(2), 1997.

[RTV99] C. Roisin, T. Tran-Thuong et L. Villard. Integration of structured video in a multimedia authoring system. Dans *Proc. of the Eurographics Multimedia'99 Workshop, Springer Computer Science*, Milan - Italy, September 1999.

[RV93] P.V. Rangan et H.M. Vin. Efficient storage techniques for digital continuous multimedia. *IEEE Transactions on Knowledge and Data Engineering*, 5(4), 1993.

[SD00] D. Sitaram et A. Dan. *Multimedia Servers*. Morgan Kaufmann, 2000.

[SI] Sun Microsystems Inc. et IBM. Java Media Framework 2.0 API. http://java.sun.com/products/java-media/jmf.

[SL90] A.P. Sheth et J.A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3), 1990.

[SLR94] P. Seshadri, M. Livny et R. Ramakrishnan. Sequence query processing. Dans *Proc. of the ACM SIGMOD International Conference on Management of Data*, June 1994.

[SM97] C. Schmid et R. Mohr. Object recognition using local characterization and semi-local constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5), 1997.

[SS99] M. Safar et C. Shahabi. 2D topological and direction relations in the world of minimum bounding circles. Dans *Proc. of IDEAS International Database Engineering and Applications Symposium*, August 1999.

[Sub98] V.S. Subrahmanian. *Principles of Multimedia Database Systems.* Morgan Kaufmann, 1998.

[Tok94] H. Tokuta. *Operating system support for continuous media applications.* Addison-Wesley, 1994.

[TPSS98] Y. Theodoridis, D. Papadias, E. Stefanakis et T. Sellis. Direction relations and two-dimensional range queries: Optimization techniques. *Data and Knowledge Engineering*, 27(3), 1998.

[TS88] D.B. Terry et D.C. Swinehart. Managing stored voice in the etherphone system. *ACM Transactions on Computer Systems*, 6(1), 1988.

[TVS96] Y. Theodoridis, M. Vazirgiannis et T. Sellis. Spatio-temporal indexing for large multimedia applications. Dans *Proc. of the IEEE-International Conference on Multimedia Systems*, 1996.

[Vaz96] M. Vazirgiannis. An object-oriented modeling of multimedia database objects and applications. Dans K.Nowsu, B. Thuraisingham et B. Berra, éditeurs, *Multimedia Database Management Systems. Design and Implementation Strategies*, chapitre 8. Kluwer Academic Publishers, 1996.

[VH95] M. Vazirgiannis et M. Hatzopoulos. Integrated multimedia object and application modelling based on events and scenarios. Dans *Proc. of the 1st IEEE International Workshop for Multimedia DBMS*, August 1995.

[VKS99] M. Vazirgiannis, I. Kostalas et T. Sellis. Specifying and authoring multimedia scenarios. *IEEE MultiMedia*, 6(3), 1999.

[VTS96] M. Vazirgiannis, Y. Theodoridis et T. Sellis. Spatio-temporal composition in multimedia applications. Dans *Proc. ot the International Workshop on Multimedia Software Development (MMSD)*, March 1996.

[VTS98] M. Vazirgiannis, Y. Theodoridis et T. Sellis. Spatio-temporal composition and indexing for large multimedia applications. *ACM/Springer-Verlag Multimedia Journal*, 6(4), 1998.

[W3C98a] Extensible Markup Language (XML). Recommendation REC-xml-19980210, World Wide Web Consortium (W3C), 1998. http://www.w3.org/TR/REC-xml.

[W3C98b] Synchronized Multimedia Integration Language (SMIL). Recommendation REC-smil-19980615, World Wide Web Consortium (W3C), 1998. http://www.w3.org/TR/REC-smil.

[WDG95] R. Weiss, A. Duda et D.K. Gifford. Composition and search with a video algebra. *IEEE Multimedia*, 2(1), 1995.

[Wie92] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3), 1992.

[YY97] B-L. Yeo et M. Yeung. Retrieving and Visualizing Video. *Communications of the ACM*, 40(12), December 1997.

[ZA00] J.L. Zechinelli-Martini et M. Adiba. Integrating and managing multimedia presentations by a web-based object server. Dans *Proc. of the EDBT 2000 PhD Workshop, 7th International Conference on Extending Database Technology*, Konstanz - Germany, March - April 2000.

Annexe A

Définitions des opérateurs OQLiST

Nous présentons ici la définition formelle des opérateurs du langage OQLiST. Cette description utilise une logique de premier ordre. Tout d'abord nous donnons la grammaire du langage sous forme BNF et la priorité de ses opérateurs. Ensuite nous définissons les constructeurs des présentations ainsi que les opérateurs temporels et spatiaux. Enfin, nous donnons des définitions de base qui servent à simplifier les définitions des opérateurs eux mêmes.

Grammaire de OQLiST

```
oqlistPROG ::= oqlistPROG ; oqlistEXPR | oqlistEXPR

oqlistEXPR ::= define IDENTIFIER as QUERY | QUERY ~ QUERY | QUERY

QUERY      ::=

// élémentaire
               nil | true | false | NUMBER | CHAR | STRING | ( QUERY )
               |
               Free | beginning | centered | end | middle | meet
               |
               IDENTIFIER | IDENTIFIER ( ARGS )
               |
               descrip ( QUERY ) | QUERY instanceof STRING
               |
// simple
               QUERY + QUERY | QUERY - QUERY
```

```
                |
                QUERY * QUERY | QUERY / QUERY
                |
                QUERY mod QUERY | - QUERY | abs QUERY | QUERY || QUERY
                |
// comparaison

                QUERY COMPARISON QUERY | QUERY like QUERY
                |
// booléen
                not QUERY | QUERY and QUERY | QUERY or QUERY
                |
// constructeur

                struct ( STRUCT )
                |
                array () | array ( ARGS )
                |
                set () | set ( ARGS ) | bag () | bag ( ARGS )
                |
                list () | list ( ARGS ) | ( QUERY, ARGS )
                |
                list ( QUERY .. QUERY ) | ( QUERY .. QUERY )
                |
// sélecteur
                QUERY DOT IDENTIFIER | QUERY DOT IDENTIFIER ( ARGS ) | * QUERY
                |
                QUERY [ QUERY ] | QUERY [ QUERY : QUERY ]
                |
                first ( QUERY ) | last ( QUERY )
                |
// collection
                for all IDENTIFIER in QUERY : QUERY
                |
                exists IDENTIFIER in QUERY : QUERY
                |
                exists ( QUERY ) | unique ( QUERY ) | QUERY in QUERY
                |
                QUERY COMPARISON QUANTIFIER QUERY
                |
                count ( * ) | count ( QUERY ) | sum ( QUERY )
                |
```

```
                min ( QUERY ) | max ( QUERY ) | avg ( QUERY )
                |
// select-from-where

                select SELECT from FROM
                |
                select SELECT from FROM CONSTRAINT
                |
                select distinct SELECT from FROM
                |
                select distinct SELECT from FROM CONSTRAINT
                |
// ensembles
                QUERY intersect QUERY | QUERY union QUERY | QUERY except QUERY
                |
// conversion
                listtoset ( QUERY ) | element ( QUERY )
                |
                distinct ( QUERY ) | flatten ( QUERY ) | ( IDENTIFIER ) QUERY
                |
// constructeur de présentations

                atomic QUERY | QUERY compound QUERY
                |
                QUERY link QUERY | QUERY button QUERY
                |
                QUERY border QUERY, QUERY | QUERY background QUERY
                |
// positionnement et taille de présentations

                QUERY at QUERY, QUERY | QUERY with QUERY, QUERY
                |
                QUERY at QUERY | QUERY during QUERY
                |
                QUERY fit QUERY, QUERY | QUERY using QUERY
                |
// présentation des collections

                QUERY table
                |
                QUERY table NUMBER | QUERY table ( QUERY )
                |
```

```
                QUERY table fit
                |
                QUERY table fit NUMBER | QUERY table fit ( QUERY )
                |
                QUERY arrange all QUERY on AXIS
                |
// relation spatiale

                QUERY disjoint QUERY, QUERY, DREL1
                |
                QUERY disjoint, QUERY, QUERY, DREL2
                |
                QUERY touch, QUERY, DREL1
                |
                QUERY touch, QUERY, QUERY, DREL2
                |
                QUERY overlap QUERY, QUERY, DREL1
                |
                QUERY overlap, QUERY, QUERY, DREL2
                |
                QUERY cover, QUERY, DREL1
                |
                QUERY cover, DREL2 | QUERY cover, QUERY, QUERY
                |
                QUERY covered by, QUERY, DREL1
                |
                QUERY covered by, DREL2 | QUERY covered by, QUERY, QUERY
                |
                QUERY inside, QUERY, QUERY | QUERY contain, QUERY, QUERY
                |
                QUERY spatial equal
                |
// relation temporelle

                QUERY synchronize before, QUERY | QUERY synchronize bi, QUERY
                |
                QUERY synchronize meet | QUERY synchronize mi
                |
                QUERY synchronize overlap, QUERY | QUERY synchronize oi, QUERY
                |
                QUERY synchronize start | QUERY synchronize si
                |
```

```
                QUERY synchronize during, QUERY | QUERY synchronize di, QUERY
                |
                QUERY synchronize finish | QUERY synchronize fi
                |
                QUERY synchronize equal
                |
// relation d'ordre

                QUERY DOT inverse | QUERY inverse
                |
                QUERY move QUERY to QUERY
                |
                QUERY swap QUERY, QUERY

ARGS       ::= ARGS, QUERY | QUERY

COMPARISON ::= QUERY = QUERY | QUERY != QUERY
                |
                QUERY < QUERY | QUERY > QUERY
                |
                QUERY <= QUERY | QUERY >= QUERY

QUANTIFIER ::= some | any | all

STRUCT     ::= STRUCT, IDENTIFIER : QUERY | IDENTIFIER : QUERY

DOT        ::= . | ->

SELECT     ::= ATTR | *

ATTR       ::= ATTR, PROJ | PROJ

PROJ       ::= QUERY | QUERY as IDENTIFIER | IDENTIFIER : QUERY

FROM       ::= FROM, VAR | VAR

VAR        ::= QUERY IDENTIFIER | QUERY as IDENTIFIER | IDENTIFIER in QUERY

CONSTRAINT ::= CONSTRAINT CONSTRAINT
                |
                where QUERY
                |
```

```
                group by ATTR | having QUERY
                |
                order by * | order by ORDERBY

ORDERBY     ::= ORDERBY, ORDERBY | QUERY | QUERY asc | QUERY desc

AXIS        ::= x | y | z | t

DREL1       ::= north | south | west | east

DREL2       ::= northwest | southeast | northeast | southwest
```

Priorités des opérateurs

Nous présentons dans la suite les opérateurs de OQLiST. Ils sont organisés par ordre descendant de priorités. Ceux qui sont regroupés sur une même ligne ont le mêmes priorités et leur associativité à gauche. Enfin, les opérateurs qui sont "en gras" concernent directement les présentations.

() [] . -> **atomic**
not abs - (unary)
in **instanceof**
* / mod intersect
\+ - union except ||
< > <= >= <some <any <all etc.
= != like
and exists for
or
.. :
asc desc
, (IDENTIFIER)
at with during fit using link button border background
compound
synchronize inverse move swap disjoint etc. **spatial north** etc.
order
having
group
where
from
select
table arrange

Constructeurs de présentations

Définitions préliminaires

Nous utilisons six opérateurs pour définir les constructeurs de présentations. Trois pour spécifier la taille, de la forme $\mathsf{D}\ \pounds_i\ \mathcal{P}$ avec $i \in \{\mathsf{x}, \mathsf{y}, \mathsf{t}\}$ (i.e., un pour chaque dimension de l'espace et le temps). Trois pour spécifier la position de la forme $\mathsf{D}\ @_i\ \mathcal{P}$. Par exemple :

$$\mathsf{D}\ @_x\ (\mathbf{OST}_1 = (\delta_x, \delta_y, \delta_t, d_x, d_y, d_t), \mathcal{O}) = (\mathbf{OST}_2 = (\mathsf{D}, \delta_y, \delta_t, d_x, d_y, d_t), \mathcal{O})$$

L'opérateur @ reçoit un réel D et une présentation atomique $(\mathbf{OST}_1 = (\delta_x, \delta_y, \delta_t, d_x, d_y, d_t), \mathcal{O})$ en entrée. Il donne en résultat une nouvelle présentation seulement si $\delta_x \neq \mathsf{D}$. De manière analogue, nous utilisons les opérateurs suivants pour dénoter les attributs spatiaux et temporels des présentations : $\mathsf{size}\pounds_i(\ \mathsf{D}\ \pounds_i\ \mathcal{P}\ ) = \mathsf{D}$ et $\mathsf{size@}_i(\ \mathsf{D}\ @_i\ \mathcal{P}\ ) = \mathsf{D}$, avec $i \in \{\mathsf{x}, \mathsf{y}, \mathsf{t}\}$.

Pour les définitions suivantes, nous utilisons les formules abs, comp_i, size, getA, getB, rels, foldA, inv, rest. Leurs définissions sont données à la fin de cet annexe.

Présentations atomiques et composites

La construction de présentations de type atomique et composite en OQLiST se fait respectivement au travers des opérateurs atomic et compound. Des sélecteurs sur les positions, sur leur contenu, sur leur relations des présentations ainsi que sur des collections de présentations sont définis dans la suite :

- $\mathsf{atomic}\ \mathcal{O} = (\mathbf{OST}, \mathcal{O})$.
- $\mathcal{O}_1\ \mathsf{compound}\ \mathcal{O}_2 = (\mathbf{OST}, \mathsf{A}\ \sigma\tau\ \mathsf{B})$ avec $\sigma\tau = (r_x, r_y, r_t) \leftrightarrow$

$$\begin{aligned}\mathbf{OST} = {} & (\mathsf{size@}_x(\mathsf{abs}(r_x, \mathsf{A}, \mathsf{B})), \mathsf{size@}_y(\mathsf{abs}(r_y, \mathsf{A}, \mathsf{B})), \mathsf{size@}_t(\mathsf{abs}(r_t, \mathsf{A}, \mathsf{B})), \\ & \mathsf{size}(r_x, \mathsf{A}, \mathsf{B}), \mathsf{size}(r_y, \mathsf{A}, \mathsf{B}), \mathsf{size}(r_t, \mathsf{A}, \mathsf{B})) \\ \wedge\ & \mathsf{comp}_x(\mathcal{P}_5, \mathcal{P}_6) = (r_x, \mathsf{A}, \mathsf{B}) \\ \wedge\ & \mathsf{comp}_y(\mathcal{P}_3, \mathcal{P}_4) = (r_y, \mathcal{P}_5, \mathcal{P}_6) \wedge \mathsf{comp}_t(\mathcal{P}_1, \mathcal{P}_2) = (r_t, \mathcal{P}_3, \mathcal{P}_4).\end{aligned}$$

- $(\mathbf{OST}, \mathcal{O}).\mathsf{content} = \mathcal{O}$.
- $\mathcal{P}.\mathsf{offset\_x} = \mathsf{size@}_x(\mathcal{P})$.

- $\mathcal{P}$.offset_y = size@$_y(\mathcal{P})$.
- $\mathcal{P}$.size_x = size$\mathcal{L}_x(\mathcal{P})$.
- $\mathcal{P}$.size_y = size$\mathcal{L}_y(\mathcal{P})$.
- $\mathcal{P}$.delay = size@$_t(\mathcal{P})$.
- $\mathcal{P}$.duration = size$\mathcal{L}_t(\mathcal{P})$.
- $\mathcal{P}$.get(i) = $\mathcal{P}$.elements[i].
- first($\mathcal{P}$) = $\mathcal{P}$.get(0).
- last($\mathcal{P}$) = $\mathcal{P}$.get(count($\mathcal{P}$.elements) − 1).
- (**OST**, $\mathcal{O}$).elements = set((**OST**, $\mathcal{O}$))

  (**OST**, A $\sigma\tau$ B).elements = getA((**OST**, A $\sigma\tau$ B)).elements
  + getB((**OST**, A $\sigma\tau$ B)).elements.

- (**OST**, $\mathcal{O}$).relations = set()

  (**OST**, A $\sigma\tau$ B).relations = rels(size@$_x$((**OST**, A $\sigma\tau$ B)), r_x,
  rels(size@$_y$((**OST**, A $\sigma\tau$ B)), r_y,
  rels(size@$_t$((**OST**, A $\sigma\tau$ B)), r_t, A, B), B), B)
  + rels(size@$_x$((**OST**, A $\sigma\tau$ B)), r_x,
  rels(size@$_y$((**OST**, A $\sigma\tau$ B)), r_y,
  rels(size@$_t$((**OST**, A $\sigma\tau$ B)), r_t, B, A), A), A)
  + foldA(getA((**OST**, A $\sigma\tau$ B)).elements,
  getB((**OST**, A $\sigma\tau$ B)).elements).

Positionnement et taille des présentations

Comme nous l'avons vu dans le chapitre 4, OQLiST fournit des constructeurs de présentations. Un constructeur de ce type définit une nouvelle présentation en modifiant la position (par rapport à un référentiel absolu), la taille, le délai et la durée d'une présentation existante. Soient X, Y, D, W, H des expressions de type réel, $\mathcal{O}$ une expression de type objet et $\mathcal{P}$ une expression de type présentation :

- $\mathcal{O}$ at X, Y = X @$_x$ Y @$_y$ $\mathcal{O}$.

- $\mathcal{O}$ at D = D $@_t$ $\mathcal{O}$.

- $\mathcal{O}$ with W, H = W $\pounds_x$ H $\pounds_y$ $\mathcal{O}$.

- $\mathcal{O}$ fit W, H = H * size$\pounds_x$(atomic $\mathcal{O}$) / size$\pounds_y$(atomic $\mathcal{O}$) $\pounds_x$ H $\pounds_y$ $\mathcal{O}$ $\leftrightarrow$
  H * size$\pounds_x$(atomic $\mathcal{O}$) / size$\pounds_y$(atomic $\mathcal{O}$) $\leq$ W

  $\mathcal{O}$ fit W, H = W * size$\pounds_y$(atomic $\mathcal{O}$) / size$\pounds_x$(atomic $\mathcal{O}$) $\pounds_y$ W $\pounds_x$ $\mathcal{O}$ $\leftrightarrow$
  W * size$\pounds_y$(atomic $\mathcal{O}$) / size$\pounds_x$(atomic $\mathcal{O}$) $<$ H.

- $\mathcal{O}$ during D = D $\pounds_t$ $\mathcal{O}$.

- $\mathcal{O}$ using $\mathcal{P}$ = size$@_x$($\mathcal{P}$) $@_x$ size$\pounds_x$($\mathcal{P}$) $\pounds_x$
  size$@_y$($\mathcal{P}$) $@_y$ size$\pounds_y$($\mathcal{P}$) $\pounds_y$ size$@_t$($\mathcal{P}$) $@_t$ size$\pounds_t$($\mathcal{P}$) $\pounds_t$ $\mathcal{O}$.

Opérateurs spatiaux

Les opérateurs spatiaux permettent de décrire l'ordonnancement spatial entre les éléments d'une présentation composite $\mathcal{PC}$ en spécifiant n'importe quelle configuration parmi les 169 configurations (cf. annexe B). Selon la forme et la sémantique de paramètres des opérateurs spatiaux, nous distinguons trois groupes d'opérateurs :

Premier groupe d'opérateurs spatiaux Combinent d'une relation topologique avec une relation directionnelle drel $\in$ { north, south, east, west }.

- $\mathcal{PC}$ disjoint D, Q, drel :

  - $\mathcal{PC}$ disjoint D, Q, north = first($\mathcal{PC}$) compound
    size$@_x$(first($\mathcal{PC}$)) + Q $@_x$
    size$@_y$(first($\mathcal{PC}$)) −
    size$\pounds_y$(rest($\mathcal{PC}$)) − D $@_y$ rest($\mathcal{PC}$) $\leftrightarrow$
    D $> 0 \wedge$ (Q $\neq 0 \vee$ size$\pounds_x$(first($\mathcal{PC}$)) = size$\pounds_x$(rest($\mathcal{PC}$))) $\wedge$
    (Q $\geq 0 \vee$ size$\pounds_x$(first($\mathcal{PC}$)) $\leq$ size$\pounds_x$(rest($\mathcal{PC}$)) + Q) $\wedge$
    (Q $\leq 0 \vee$ size$\pounds_x$(first($\mathcal{PC}$)) $\geq$ size$\pounds_x$(rest($\mathcal{PC}$)) + Q).

  - $\mathcal{PC}$ disjoint D, Q, south = first($\mathcal{PC}$) compound
    size$@_x$(first($\mathcal{PC}$)) + Q $@_x$
    size$@_y$(first($\mathcal{PC}$)) +

$$\text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) + \text{D } @_y \text{ rest}(\ \mathcal{PC}\ ) \leftrightarrow$$
$$\text{D} > 0 \wedge (\text{Q} \neq 0 \vee \text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) = \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC}))) \wedge$$
$$(\text{Q} \geq 0 \vee \text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) \leq \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) + \text{Q}) \wedge$$
$$(\text{Q} \leq 0 \vee \text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) \geq \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) + \text{Q}).$$

– $\mathcal{PC}$ disjoint D, Q, west = first($\mathcal{PC}$) compound

$$\text{size}@_x(\text{first}(\mathcal{PC})) -$$
$$\text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) - \text{D } @_x$$
$$\text{size}@_y(\text{first}(\mathcal{PC})) + \text{Q } @_y \text{ rest}(\ \mathcal{PC}\ ) \leftrightarrow$$
$$\text{D} > 0 \wedge (\text{Q} \neq 0 \vee \text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) = \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC}))) \wedge$$
$$(\text{Q} \geq 0 \vee \text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) \leq \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) + \text{Q}) \wedge$$
$$(\text{Q} \leq 0 \vee \text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) \geq \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) + \text{Q}).$$

– $\mathcal{PC}$ disjoint D, Q, east = first($\mathcal{PC}$) compound

$$\text{size}@_x(\text{first}(\mathcal{PC})) +$$
$$\text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) + \text{D } @_x$$
$$\text{size}@_y(\text{first}(\mathcal{PC})) + \text{Q } @_y \text{ rest}(\ \mathcal{PC}\ ) \leftrightarrow$$
$$\text{D} > 0 \wedge (\text{Q} \neq 0 \vee \text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) = \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC}))) \wedge$$
$$(\text{Q} \geq 0 \vee \text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) \leq \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) + \text{Q}) \wedge$$
$$(\text{Q} \leq 0 \vee \text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) \geq \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) + \text{Q}).$$

- $\mathcal{PC}$ touch, Q, drel :

– $\mathcal{PC}$ touch, Q, north = first($\mathcal{PC}$) compound

$$\text{size}@_x(\text{first}(\mathcal{PC})) + \text{Q } @_x$$
$$- \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \ @_y \text{ rest}(\ \mathcal{PC}\ ) \leftrightarrow$$
$$(\text{Q} \neq 0 \vee \text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) = \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC}))) \wedge$$
$$(\text{Q} \geq 0 \vee \text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) \leq \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) + \text{Q}) \wedge$$
$$(\text{Q} \leq 0 \vee \text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) \geq \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) + \text{Q}).$$

– $\mathcal{PC}$ touch, Q, south = first($\mathcal{PC}$) compound

$$\text{size}@_x(\text{first}(\mathcal{PC})) + \text{Q } @_x$$
$$\text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \ @_y \text{ rest}(\ \mathcal{PC}\ ) \leftrightarrow$$
$$(\text{Q} \neq 0 \vee \text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) = \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC}))) \wedge$$
$$(\text{Q} \geq 0 \vee \text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) \leq \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) + \text{Q}) \wedge$$
$$(\text{Q} \leq 0 \vee \text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) \geq \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) + \text{Q}).$$

- $\mathcal{PC}$ touch, Q, west = first($\mathcal{PC}$) compound
  $-$ size$\pounds_x$(rest($\mathcal{PC}$)) @$_x$
  size@$_y$(first($\mathcal{PC}$)) + Q @$_y$ rest($\mathcal{PC}$) $\leftrightarrow$
  (Q $\neq 0 \vee$ size$\pounds_y$(first($\mathcal{PC}$)) = size$\pounds_y$(rest($\mathcal{PC}$))) $\wedge$
  (Q $\geq 0 \vee$ size$\pounds_y$(first($\mathcal{PC}$)) $\leq$ size$\pounds_y$(rest($\mathcal{PC}$)) + Q) $\wedge$
  (Q $\leq 0 \vee$ size$\pounds_y$(first($\mathcal{PC}$)) $\geq$ size$\pounds_y$(rest($\mathcal{PC}$)) + Q).

- $\mathcal{PC}$ touch, Q, east = first($\mathcal{PC}$) compound
  size$\pounds_x$(rest($\mathcal{PC}$)) @$_x$
  size@$_y$(first($\mathcal{PC}$)) + Q @$_y$ rest($\mathcal{PC}$) $\leftrightarrow$
  (Q $\neq 0 \vee$ size$\pounds_y$(first($\mathcal{PC}$)) = size$\pounds_y$(rest($\mathcal{PC}$))) $\wedge$
  (Q $\geq 0 \vee$ size$\pounds_y$(first($\mathcal{PC}$)) $\leq$ size$\pounds_y$(rest($\mathcal{PC}$)) + Q) $\wedge$
  (Q $\leq 0 \vee$ size$\pounds_y$(first($\mathcal{PC}$)) $\geq$ size$\pounds_y$(rest($\mathcal{PC}$)) + Q).

- $\mathcal{PC}$ overlap D, Q, drel :

  - $\mathcal{PC}$ overlap D, Q, north = first($\mathcal{PC}$) compound
    size@$_x$(first($\mathcal{PC}$)) + Q @$_x$
    size@$_y$(first($\mathcal{PC}$)) $-$ D @$_y$ rest($\mathcal{PC}$) $\leftrightarrow$
    D $>$ $-$ size$\pounds_y$(rest($\mathcal{PC}$)) $\wedge$ D $< 0 \wedge$
    (Q $\neq 0 \vee$ size$\pounds_x$(first($\mathcal{PC}$)) = size$\pounds_x$(rest($\mathcal{PC}$))) $\wedge$
    (Q $\geq 0 \vee$ size$\pounds_x$(first($\mathcal{PC}$)) $\leq$ size$\pounds_x$(rest($\mathcal{PC}$)) + Q) $\wedge$
    (Q $\leq 0 \vee$ size$\pounds_x$(first($\mathcal{PC}$)) $\geq$ size$\pounds_x$(rest($\mathcal{PC}$)) + Q).

  - $\mathcal{PC}$ overlap D, Q, south = first($\mathcal{PC}$) compound
    size@$_x$(first($\mathcal{PC}$)) + Q @$_x$
    size@$_y$(first($\mathcal{PC}$)) + D @$_y$ rest($\mathcal{PC}$) $\leftrightarrow$
    D + size$\pounds_y$(rest($\mathcal{PC}$)) $>$ size$\pounds_y$(first($\mathcal{PC}$)) $\wedge$ D $<$ size$\pounds_y$(first($\mathcal{PC}$)) $\wedge$
    (Q $\neq 0 \vee$ size$\pounds_x$(first($\mathcal{PC}$)) = size$\pounds_x$(rest($\mathcal{PC}$))) $\wedge$
    (Q $\geq 0 \vee$ size$\pounds_x$(first($\mathcal{PC}$)) $\leq$ size$\pounds_x$(rest($\mathcal{PC}$)) + Q) $\wedge$
    (Q $\leq 0 \vee$ size$\pounds_x$(first($\mathcal{PC}$)) $\geq$ size$\pounds_x$(rest($\mathcal{PC}$)) + Q).

  - $\mathcal{PC}$ overlap D, Q, west = first($\mathcal{PC}$) compound
    size@$_x$(first($\mathcal{PC}$)) $-$ D @$_x$
    size@$_y$(first($\mathcal{PC}$)) + Q @$_y$ rest($\mathcal{PC}$) $\leftrightarrow$

$\mathsf{D} > -\ \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC})) \wedge \mathsf{D} < 0 \wedge$
$(\mathsf{Q} \neq 0 \vee \mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) = \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC}))) \wedge$
$(\mathsf{Q} \geq 0 \vee \mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) \leq \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) + \mathsf{Q}) \wedge$
$(\mathsf{Q} \leq 0 \vee \mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) \geq \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) + \mathsf{Q})$.

– $\mathcal{PC}$ overlap D, Q, east = first($\mathcal{PC}$) compound
$\mathsf{size}@_x(\mathsf{first}(\mathcal{PC})) + \mathsf{D}\ @_x$
$\mathsf{size}@_y(\mathsf{first}(\mathcal{PC})) + \mathsf{Q}\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
$\mathsf{D} + \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC})) > \mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) \wedge \mathsf{D} < \mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) \wedge$
$(\mathsf{Q} \neq 0 \vee \mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) = \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC}))) \wedge$
$(\mathsf{Q} \geq 0 \vee \mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) \leq \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) + \mathsf{Q}) \wedge$
$(\mathsf{Q} \leq 0 \vee \mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) \geq \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) + \mathsf{Q})$.

- $\mathcal{PC}$ cover, Q, drel :

– $\mathcal{PC}$ cover, Q, north = first($\mathcal{PC}$) compound
$\mathsf{size}@_x(\mathsf{first}(\mathcal{PC})) + \mathsf{Q}\ @_x$
$\mathsf{size}@_y(\mathsf{first}(\mathcal{PC}))\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
$\mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) \geq \mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) \wedge$
$(\mathsf{Q} \neq 0 \vee \mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) = \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC}))) \wedge$
$(\mathsf{Q} \leq 0 \vee \mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) \geq \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC})) + \mathsf{Q})$.

– $\mathcal{PC}$ cover, Q, south = first($\mathcal{PC}$) compound
$\mathsf{size}@_x(\mathsf{first}(\mathcal{PC})) + \mathsf{Q}\ @_x$
$\mathsf{size}@_y(\mathsf{first}(\mathcal{PC})) +$
$\mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC}))\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
$\mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) \geq \mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) \wedge$
$(\mathsf{Q} \neq 0 \vee \mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) = \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC}))) \wedge$
$(\mathsf{Q} \leq 0 \vee \mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) \geq \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC})) + \mathsf{Q})$.

– $\mathcal{PC}$ cover, Q, west = first($\mathcal{PC}$) compound
$\mathsf{size}@_x(\mathsf{first}(\mathcal{PC}))\ @_x$
$\mathsf{size}@_y(\mathsf{first}(\mathcal{PC})) + \mathsf{Q}\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
$\mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC})) \geq \mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) \wedge$
$(\mathsf{Q} \neq 0 \vee \mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) = \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC}))) \wedge$
$(\mathsf{Q} \leq 0 \vee \mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) \geq \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) + \mathsf{Q})$.

- $\mathcal{PC}$ cover, Q, east = first($\mathcal{PC}$) compound
  $\text{size}@_x(\text{first}(\mathcal{PC})) +$
  $\text{size}\pounds_x(\text{first}(\mathcal{PC})) - \text{size}\pounds_x(\text{rest}(\mathcal{PC}))\ @_x$
  $\text{size}@_y(\text{first}(\mathcal{PC})) + \text{Q}\ @_y\ \text{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
  $\text{size}\pounds_x(\text{rest}(\mathcal{PC})) \geq \text{size}\pounds_x(\text{first}(\mathcal{PC})) \wedge$
  $(\text{Q} \neq 0 \vee \text{size}\pounds_y(\text{first}(\mathcal{PC})) = \text{size}\pounds_y(\text{rest}(\mathcal{PC}))) \wedge$
  $(\text{Q} \leq 0 \vee \text{size}\pounds_y(\text{first}(\mathcal{PC})) \geq \text{size}\pounds_y(\text{rest}(\mathcal{PC})) + \text{Q}).$

- $\mathcal{PC}$ covered by, Q, drel :

  - $\mathcal{PC}$ covered by, Q, north = first($\mathcal{PC}$) compound
    $\text{size}@_x(\text{first}(\mathcal{PC})) + \text{Q}\ @_x$
    $\text{size}@_y(\text{first}(\mathcal{PC}))\ @_y\ \text{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
    $\text{size}\pounds_y(\text{rest}(\mathcal{PC})) \leq \text{size}\pounds_y(\text{first}(\mathcal{PC})) \wedge$
    $(\text{Q} \neq 0 \vee \text{size}\pounds_x(\text{first}(\mathcal{PC})) = \text{size}\pounds_x(\text{rest}(\mathcal{PC}))) \wedge$
    $(\text{Q} \geq 0 \vee \text{size}\pounds_x(\text{first}(\mathcal{PC})) \leq \text{size}\pounds_x(\text{rest}(\mathcal{PC})) + \text{Q}).$

  - $\mathcal{PC}$ covered by, Q, south = first($\mathcal{PC}$) compound
    $\text{size}@_x(\text{first}(\mathcal{PC})) + \text{Q}\ @_x$
    $\text{size}@_y(\text{first}(\mathcal{PC})) +$
    $\text{size}\pounds_y(\text{first}(\mathcal{PC})) - \text{size}\pounds_y(\text{rest}(\mathcal{PC}))\ @_y\ \text{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
    $\text{size}\pounds_y(\text{rest}(\mathcal{PC})) \leq \text{size}\pounds_y(\text{first}(\mathcal{PC})) \wedge$
    $(\text{Q} \neq 0 \vee \text{size}\pounds_x(\text{first}(\mathcal{PC})) = \text{size}\pounds_x(\text{rest}(\mathcal{PC}))) \wedge$
    $(\text{Q} \geq 0 \vee \text{size}\pounds_x(\text{first}(\mathcal{PC})) \leq \text{size}\pounds_x(\text{rest}(\mathcal{PC})) + \text{Q}).$

  - $\mathcal{PC}$ covered by, Q, west = first($\mathcal{PC}$) compound
    $\text{size}@_x(\text{first}(\mathcal{PC}))\ @_x$
    $\text{size}@_y(\text{first}(\mathcal{PC})) + \text{Q}\ @_y\ \text{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
    $\text{size}\pounds_x(\text{rest}(\mathcal{PC})) \leq \text{size}\pounds_x(\text{first}(\mathcal{PC})) \wedge$
    $(\text{Q} \neq 0 \vee \text{size}\pounds_y(\text{first}(\mathcal{PC})) = \text{size}\pounds_y(\text{rest}(\mathcal{PC}))) \wedge$
    $(\text{Q} \geq 0 \vee \text{size}\pounds_y(\text{first}(\mathcal{PC})) \leq \text{size}\pounds_y(\text{rest}(\mathcal{PC})) + \text{Q}).$

  - $\mathcal{PC}$ covered by, Q, east = first($\mathcal{PC}$) compound
    $\text{size}@_x(\text{first}(\mathcal{PC})) +$
    $\text{size}\pounds_x(\text{first}(\mathcal{PC})) - \text{size}\pounds_x(\text{rest}(\mathcal{PC}))\ @_x$

$$\mathsf{size@}_y(\mathsf{first}(\mathcal{PC})) + \mathsf{Q}\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow$$
$$\mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \leq \mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) \wedge$$
$$(\mathsf{Q} \neq 0 \vee \mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) = \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC}))) \wedge$$
$$(\mathsf{Q} \geq 0 \vee \mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) \leq \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) + \mathsf{Q}).$$

Deuxième groupe d'opérateurs spatiaux Combinent une relation topologique avec une relation directionnelle drel $\in \{$ northwest, southeast, northeast, southwest $\}$.

- $\mathcal{PC}$ disjoint, P, Q, drel :

  - $\mathcal{PC}$ disjoint, P, Q, northwest = first($\mathcal{PC}$) compound

    $$\mathsf{size@}_x(\mathsf{first}(\mathcal{PC})) - \mathsf{P}\ @_x$$
    $$\mathsf{size@}_y(\mathsf{first}(\mathcal{PC})) - \mathsf{Q}\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow$$
    $$(\mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) < \mathsf{P} \vee \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) < \mathsf{Q}) \wedge$$
    $$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{P} \neq 0) \wedge$$
    $$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) < \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{P} \leq 0) \wedge$$
    $$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee$$
    $$\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \geq \mathsf{P}) \wedge$$
    $$(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{Q} \neq 0) \wedge$$
    $$(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) < \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{Q} \leq 0) \wedge$$
    $$(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee$$
    $$\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \geq \mathsf{Q}).$$

  - $\mathcal{PC}$ disjoint, P, Q, southeast = first($\mathcal{PC}$) compound

    $$\mathsf{size@}_x(\mathsf{first}(\mathcal{PC})) + \mathsf{P}\ @_x$$
    $$\mathsf{size@}_y(\mathsf{first}(\mathcal{PC})) + \mathsf{Q}\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow$$
    $$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) < \mathsf{P} \vee \mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) < \mathsf{Q}) \wedge$$
    $$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{P} \neq 0) \wedge$$
    $$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{P} \geq 0) \wedge$$
    $$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) < \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee$$
    $$\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \leq \mathsf{P}) \wedge$$
    $$(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{Q} \neq 0) \wedge$$
    $$(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{Q} \geq 0) \wedge$$
    $$(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) < \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee$$
    $$\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \leq \mathsf{Q}).$$

  - $\mathcal{PC}$ disjoint, P, Q, northeast = first($\mathcal{PC}$) compound

$$\begin{aligned}
&\mathsf{size@}_x(\mathsf{first}(\mathcal{PC})) + \mathsf{P}\ @_x \\
&\mathsf{size@}_y(\mathsf{first}(\mathcal{PC})) - \mathsf{Q}\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow \\
&(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) < \mathsf{P} \vee \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) < \mathsf{Q}) \wedge \\
&(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{P} \neq 0) \wedge \\
&(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{P} \geq 0) \wedge \\
&(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) < \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \\
&\ \mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \leq \mathsf{P}) \wedge \\
&(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{Q} \neq 0) \wedge \\
&(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) < \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{Q} \leq 0) \wedge \\
&(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \\
&\ \mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \geq \mathsf{Q}).
\end{aligned}$$

– $\mathcal{PC}$ disjoint, P, Q, southwest = first($\mathcal{PC}$) compound

$$\begin{aligned}
&\mathsf{size@}_x(\mathsf{first}(\mathcal{PC})) - \mathsf{P}\ @_x \\
&\mathsf{size@}_y(\mathsf{first}(\mathcal{PC})) + \mathsf{Q}\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow \\
&(\mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) < \mathsf{P} \vee \mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) < \mathsf{Q}) \wedge \\
&(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{P} \neq 0) \wedge \\
&(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) < \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{P} \leq 0) \wedge \\
&(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \\
&\ \mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \geq \mathsf{P}) \wedge \\
&(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{Q} \neq 0) \wedge \\
&(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{Q} \geq 0) \wedge \\
&(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) < \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \\
&\ \mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \leq \mathsf{Q}).
\end{aligned}$$

- $\mathcal{PC}$ touch, P, Q, drel :

– $\mathcal{PC}$ touch, P, Q, northwest = first($\mathcal{PC}$) compound

$$\begin{aligned}
&\mathsf{size@}_x(\mathsf{first}(\mathcal{PC})) - \mathsf{P}\ @_x \\
&\mathsf{size@}_x(\mathsf{first}(\mathcal{PC})) - \mathsf{Q}\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow \\
&(\mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) = \mathsf{P} \wedge \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \geq \mathsf{Q} \vee \\
&\ \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) = \mathsf{Q} \wedge \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \geq \mathsf{P}) \wedge \\
&(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{P} \neq 0) \wedge \\
&(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) < \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{P} \leq 0) \wedge \\
&(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \\
&\ \mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \geq \mathsf{P}) \wedge \\
&(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \mathsf{Q} \neq 0) \wedge
\end{aligned}$$

$(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) < \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \leq 0) \wedge$
$(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) > \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee$
$\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) - \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \geq \text{Q}).$

– $\mathcal{PC}$ touch, P, Q, southeast = first($\mathcal{PC}$) compound
$\text{size}@_x(\text{first}(\mathcal{PC})) + \text{P}\ @_x$
$\text{size}@_x(\text{first}(\mathcal{PC})) + \text{Q}\ @_y\ \text{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
$(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) = \text{P} \wedge \text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) \geq \text{Q} \vee$
$\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) = \text{Q} \wedge \text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) \geq \text{P}) \wedge$
$(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) \neq \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee \text{P} \neq 0) \wedge$
$(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) > \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee \text{P} \geq 0) \wedge$
$(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) < \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee$
$\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) - \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \leq \text{P}) \wedge$
$(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) \neq \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \neq 0) \wedge$
$(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) > \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \geq 0) \wedge$
$(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) < \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee$
$\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) - \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \leq \text{Q}).$

– $\mathcal{PC}$ touch, P, Q, northeast = first($\mathcal{PC}$) compound
$\text{size}@_x(\text{first}(\mathcal{PC})) + \text{P}\ @_x$
$\text{size}@_x(\text{first}(\mathcal{PC})) - \text{Q}\ @_y\ \text{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
$(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) = \text{P} \wedge \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \geq \text{Q} \vee$
$\text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) = \text{Q} \wedge \text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) \geq \text{P}) \wedge$
$(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) \neq \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee \text{P} \neq 0) \wedge$
$(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) > \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee \text{P} \geq 0) \wedge$
$(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) < \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee$
$\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) - \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \leq \text{P}) \wedge$
$(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) \neq \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \neq 0) \wedge$
$(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) < \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \leq 0) \wedge$
$(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) > \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee$
$\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) - \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \geq \text{Q}).$

– $\mathcal{PC}$ touch, P, Q, southwest = first($\mathcal{PC}$) compound
$\text{size}@_x(\text{first}(\mathcal{PC})) - \text{P}\ @_x$
$\text{size}@_x(\text{first}(\mathcal{PC})) + \text{Q}\ @_y\ \text{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
$(\text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) = \text{P} \wedge \text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) \geq \text{Q} \vee$

$$\begin{aligned}
&\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) = \text{Q} \wedge \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \geq \text{P}) \wedge \\
&(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) \neq \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee \text{P} \neq 0) \wedge \\
&(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) < \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee \text{P} \leq 0) \wedge \\
&(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) > \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee \\
&\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) - \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \geq \text{P}) \wedge \\
&(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) \neq \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \neq 0) \wedge \\
&(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) > \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \geq 0) \wedge \\
&(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) < \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \\
&\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) - \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \leq \text{Q}).
\end{aligned}$$

- $\mathcal{PC}$ overlap, P, Q, drel :

  - $\mathcal{PC}$ overlap, P, Q, northwest = first($\mathcal{PC}$) compound

$$\begin{aligned}
&\text{size}@_x(\text{first}(\mathcal{PC})) - \text{P}\ @_x \\
&\text{size}@_y(\text{first}(\mathcal{PC})) - \text{Q}\ @_y\ \text{rest}(\ \mathcal{PC}\ ) \leftrightarrow \\
&(\text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) > \text{P} \vee \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) > \text{Q}) \wedge \\
&(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) \neq \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee \text{P} \neq 0) \wedge \\
&(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) < \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee \text{P} \leq 0) \wedge \\
&(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) > \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee \\
&\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) - \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \geq \text{P}) \wedge \\
&(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) \neq \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \neq 0) \wedge \\
&(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) < \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \leq 0) \wedge \\
&(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) > \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \\
&\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) - \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \geq \text{Q}).
\end{aligned}$$

  - $\mathcal{PC}$ overlap, P, Q, southeast = first($\mathcal{PC}$) compound

$$\begin{aligned}
&\text{size}@_x(\text{first}(\mathcal{PC})) + \text{P}\ @_x \\
&\text{size}@_y(\text{first}(\mathcal{PC})) + \text{Q}\ @_y\ \text{rest}(\ \mathcal{PC}\ ) \leftrightarrow \\
&(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) > \text{P} \vee \text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) > \text{Q}) \wedge \\
&(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) \neq \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee \text{P} \neq 0) \wedge \\
&(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) > \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee \text{P} \geq 0) \wedge \\
&(\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) < \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \vee \\
&\text{size}\mathcal{L}_x(\text{first}(\mathcal{PC})) - \text{size}\mathcal{L}_x(\text{rest}(\mathcal{PC})) \leq \text{P}) \wedge \\
&(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) \neq \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \neq 0) \wedge \\
&(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) > \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \geq 0) \wedge \\
&(\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) < \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \vee \\
&\text{size}\mathcal{L}_y(\text{first}(\mathcal{PC})) - \text{size}\mathcal{L}_y(\text{rest}(\mathcal{PC})) \leq \text{Q}).
\end{aligned}$$

– $\mathcal{PC}$ overlap, P, Q, northeast = first($\mathcal{PC}$) compound
$\text{size}@_x(\text{first}(\mathcal{PC})) + \text{P}\ @_x$
$\text{size}@_y(\text{first}(\mathcal{PC})) - \text{Q}\ @_y\ \text{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
$(\text{size}\pounds_x(\text{first}(\mathcal{PC})) > \text{P} \vee \text{size}\pounds_y(\text{rest}(\mathcal{PC})) > \text{Q}) \wedge$
$(\text{size}\pounds_x(\text{first}(\mathcal{PC})) \neq \text{size}\pounds_x(\text{rest}(\mathcal{PC})) \vee \text{P} \neq 0) \wedge$
$(\text{size}\pounds_x(\text{first}(\mathcal{PC})) > \text{size}\pounds_x(\text{rest}(\mathcal{PC})) \vee \text{P} \geq 0) \wedge$
$(\text{size}\pounds_x(\text{first}(\mathcal{PC})) < \text{size}\pounds_x(\text{rest}(\mathcal{PC})) \vee$
$\text{size}\pounds_x(\text{first}(\mathcal{PC})) - \text{size}\pounds_x(\text{rest}(\mathcal{PC})) \leq \text{P}) \wedge$
$(\text{size}\pounds_y(\text{first}(\mathcal{PC})) \neq \text{size}\pounds_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \neq 0) \wedge$
$(\text{size}\pounds_y(\text{first}(\mathcal{PC})) < \text{size}\pounds_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \leq 0) \wedge$
$(\text{size}\pounds_y(\text{first}(\mathcal{PC})) > \text{size}\pounds_y(\text{rest}(\mathcal{PC})) \vee$
$\text{size}\pounds_y(\text{first}(\mathcal{PC})) - \text{size}\pounds_y(\text{rest}(\mathcal{PC})) \geq \text{Q})$.

– $\mathcal{PC}$ overlap, P, Q, southwest = first($\mathcal{PC}$) compound
$\text{size}@_x(\text{first}(\mathcal{PC})) - \text{P}\ @_x$
$\text{size}@_y(\text{first}(\mathcal{PC})) + \text{Q}\ @_y\ \text{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
$(\text{size}\pounds_x(\text{rest}(\mathcal{PC})) > \text{P} \vee \text{size}\pounds_y(\text{first}(\mathcal{PC})) > \text{Q}) \wedge$
$(\text{size}\pounds_x(\text{first}(\mathcal{PC})) \neq \text{size}\pounds_x(\text{rest}(\mathcal{PC})) \vee \text{P} \neq 0) \wedge$
$(\text{size}\pounds_x(\text{first}(\mathcal{PC})) < \text{size}\pounds_x(\text{rest}(\mathcal{PC})) \vee \text{P} \leq 0) \wedge$
$(\text{size}\pounds_x(\text{first}(\mathcal{PC})) > \text{size}\pounds_x(\text{rest}(\mathcal{PC})) \vee$
$\text{size}\pounds_x(\text{first}(\mathcal{PC})) - \text{size}\pounds_x(\text{rest}(\mathcal{PC})) \geq \text{P}) \wedge$
$(\text{size}\pounds_y(\text{first}(\mathcal{PC})) \neq \text{size}\pounds_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \neq 0) \wedge$
$(\text{size}\pounds_y(\text{first}(\mathcal{PC})) > \text{size}\pounds_y(\text{rest}(\mathcal{PC})) \vee \text{Q} \geq 0) \wedge$
$(\text{size}\pounds_y(\text{first}(\mathcal{PC})) < \text{size}\pounds_y(\text{rest}(\mathcal{PC})) \vee$
$\text{size}\pounds_y(\text{first}(\mathcal{PC})) - \text{size}\pounds_y(\text{rest}(\mathcal{PC})) \leq \text{Q})$.

- $\mathcal{PC}$ cover, drel :

– $\mathcal{PC}$ cover, northwest = first($\mathcal{PC}$) compound
$\text{size}@_x(\text{first}(\mathcal{PC})) -$
$\text{size}\pounds_x(\text{first}(\mathcal{PC})) + \text{size}\pounds_x(\text{rest}(\mathcal{PC}))\ @_x$
$\text{size}@_y(\text{first}(\mathcal{PC})) -$
$\text{size}\pounds_y(\text{first}(\mathcal{PC})) + \text{size}\pounds_y(\text{rest}(\mathcal{PC}))\ @_y\ \text{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
$(\text{size}\pounds_x(\text{first}(\mathcal{PC})) \neq \text{size}\pounds_x(\text{rest}(\mathcal{PC})) \vee$
$\text{size}\pounds_x(\text{first}(\mathcal{PC})) - \text{size}\pounds_x(\text{rest}(\mathcal{PC})) \neq 0) \wedge$

$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee$
$\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \leq 0) \wedge$
$(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee$
$\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \neq 0) \wedge$
$(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee$
$\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \leq 0).$

– $\mathcal{PC}$ cover, southeast = first($\mathcal{PC}$) compound
$\mathsf{size}@_x(\mathsf{first}(\mathcal{PC}))\ @_x$
$\mathsf{size}@_y(\mathsf{first}(\mathcal{PC}))\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC}))) \wedge$
$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee$
$\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \geq 0) \wedge$
$(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC}))) \wedge$
$(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) < \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee$
$\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \leq 0).$

– $\mathcal{PC}$ cover, northeast = first($\mathcal{PC}$) compound
$\mathsf{size}@_x(\mathsf{first}(\mathcal{PC}))\ @_x$
$\mathsf{size}@_y(\mathsf{first}(\mathcal{PC}))\ -$
$\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) + \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC}))\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC}))) \wedge$
$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee$
$\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \geq 0) \wedge$
$(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee$
$\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \neq 0) \wedge$
$(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee$
$\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \leq 0).$

– $\mathcal{PC}$ cover, southwest = first($\mathcal{PC}$) compound
$\mathsf{size}@_x(\mathsf{first}(\mathcal{PC}))\ -$
$\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) + \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC}))\ @_x$
$\mathsf{size}@_y(\mathsf{first}(\mathcal{PC}))\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow$
$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee$
$\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \neq 0) \wedge$
$(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee$

$$\begin{array}{l}
\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \leq 0) \wedge \\
(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC}))) \wedge \\
(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) < \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \\
\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \leq 0).
\end{array}$$

- $\mathcal{PC}$ covered by, drel :

  - $\mathcal{PC}$ covered by, northwest = first($\mathcal{PC}$) compound

$$\begin{array}{l}
\mathsf{size@}_x(\mathsf{first}(\mathcal{PC}))\ @_x \\
\mathsf{size@}_y(\mathsf{first}(\mathcal{PC}))\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow \\
(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC}))) \wedge \\
(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \\
\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \geq 0) \wedge \\
(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC}))) \wedge \\
(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) < \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \\
\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \leq 0).
\end{array}$$

  - $\mathcal{PC}$ covered by, southeast = first($\mathcal{PC}$) compound

$$\begin{array}{l}
\mathsf{size@}_x(\mathsf{first}(\mathcal{PC})) + \\
\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) + \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC}))\ @_x \\
\mathsf{size@}_y(\mathsf{first}(\mathcal{PC})) + \\
\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) + \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC}))\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow \\
(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \\
\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \neq 0) \wedge \\
(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee \\
\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \leq 0) \wedge \\
(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \\
\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \neq 0) \wedge \\
(\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \vee \\
\mathsf{size}\pounds_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\pounds_y(\mathsf{rest}(\mathcal{PC})) \leq 0).
\end{array}$$

  - $\mathcal{PC}$ covered by, northeast = first($\mathcal{PC}$) compound

$$\begin{array}{l}
\mathsf{size@}_x(\mathsf{first}(\mathcal{PC})) + \\
\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) + \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC}))\ @_x \\
\mathsf{size@}_y(\mathsf{first}(\mathcal{PC}))\ @_y\ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow \\
(\mathsf{size}\pounds_x(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\pounds_x(\mathsf{rest}(\mathcal{PC})) \vee
\end{array}$$

$$\begin{aligned}
&\mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC})) \neq 0) \wedge \\
&(\mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC})) \vee \\
&\mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC})) \leq 0) \wedge \\
&(\mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC}))) \wedge \\
&(\mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) < \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) \vee \\
&\mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) \leq 0).
\end{aligned}$$

– $\mathcal{PC}$ covered by, southwest = first($\mathcal{PC}$) compound

$$\begin{aligned}
&\mathsf{size}@_x(\mathsf{first}(\mathcal{PC})) \ @_x \\
&\mathsf{size}@_y(\mathsf{first}(\mathcal{PC})) + \\
&\mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) + \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) \ @_y \ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow \\
&(\mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC}))) \wedge \\
&(\mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC})) \vee \\
&\mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC})) \geq 0) \wedge \\
&(\mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) \neq \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) \vee \\
&\mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) \neq 0) \wedge \\
&(\mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) > \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) \vee \\
&\mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) - \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) \leq 0).
\end{aligned}$$

Troisième groupe d'opérateurs spatiaux Permettent de définir des configurations qui ne peuvent pas être décrites en utilisant une relation directionnelle.

– $\mathcal{PC}$ cover, P, Q = first($\mathcal{PC}$) compound

$$\begin{aligned}
&\mathsf{size}@_x(\mathsf{first}(\mathcal{PC})) + \mathsf{P} \ @_x \\
&\mathsf{size}@_y(\mathsf{first}(\mathcal{PC})) + \mathsf{Q} \ @_y \ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow \\
&\mathsf{P} \leq 0 \wedge \mathsf{Q} \leq 0 \wedge (\mathsf{P} = 0 \vee \mathsf{Q} = 0 \vee \\
&\mathsf{P} + \mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) = \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC})) \vee \\
&\mathsf{Q} + \mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC})) = \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC}))).
\end{aligned}$$

– $\mathcal{PC}$ covered by, P, Q = first($\mathcal{PC}$) compound

$$\begin{aligned}
&\mathsf{size}@_x(\mathsf{first}(\mathcal{PC})) + \mathsf{P} \ @_x \\
&\mathsf{size}@_y(\mathsf{first}(\mathcal{PC})) + \mathsf{Q} \ @_y \ \mathsf{rest}(\ \mathcal{PC}\ ) \leftrightarrow \\
&\mathsf{P} \geq 0 \wedge \mathsf{Q} \geq 0 \wedge (\mathsf{P} = 0 \vee \mathsf{Q} = 0 \vee \\
&\mathsf{P} + \mathsf{size}\mathcal{L}_x(\mathsf{rest}(\mathcal{PC})) = \mathsf{size}\mathcal{L}_x(\mathsf{first}(\mathcal{PC})) \vee \\
&\mathsf{Q} + \mathsf{size}\mathcal{L}_y(\mathsf{rest}(\mathcal{PC})) = \mathsf{size}\mathcal{L}_y(\mathsf{first}(\mathcal{PC}))).
\end{aligned}$$

- $\mathcal{PC}$ inside, P, Q = first($\mathcal{PC}$) compound
  size@$_x$(first($\mathcal{PC}$)) + P @$_x$
  size@$_y$(first($\mathcal{PC}$)) + Q @$_y$ rest($\mathcal{PC}$) $\leftrightarrow$ P $> 0 \wedge$ Q > 0.

- $\mathcal{PC}$ contain, P, Q = first($\mathcal{PC}$) compound
  size@$_x$(first($\mathcal{PC}$)) + P @$_x$
  size@$_y$(first($\mathcal{PC}$)) + Q @$_y$ rest($\mathcal{PC}$) $\leftrightarrow$ P $< 0 \wedge$ Q < 0.

- $\mathcal{PC}$ spatial equal = first($\mathcal{PC}$) compound
  size@$_x$(first($\mathcal{PC}$)) @$_x$
  size@$_y$(first($\mathcal{PC}$)) @$_y$ rest($\mathcal{PC}$) $\leftrightarrow$
  size$\mathcal{L}_x$(first($\mathcal{PC}$)) = size$\mathcal{L}_x$(rest($\mathcal{PC}$)) $\wedge$
  size$\mathcal{L}_y$(first($\mathcal{PC}$)) = size$\mathcal{L}_y$(rest($\mathcal{PC}$)).

Opérateurs temporels

Les opérateurs temporels en OQLiST expriment la synchronisation de présentations.

- $\mathcal{PC}$ synchronize before, D = first($\mathcal{PC}$) compound
  size@$_t$(first($\mathcal{PC}$)) −
  size$\mathcal{L}_t$(rest($\mathcal{PC}$)) − D @$_t$ rest($\mathcal{PC}$) $\leftrightarrow$
  D $> 0 \wedge$ size@$_t$(first($\mathcal{PC}$)) − size$\mathcal{L}_t$(rest($\mathcal{PC}$)) − D ≥ 0.

- $\mathcal{PC}$ synchronize bi, D = first($\mathcal{PC}$) compound
  size@$_t$(first($\mathcal{PC}$)) +
  size$\mathcal{L}_t$(first($\mathcal{PC}$)) + D @$_t$ rest($\mathcal{PC}$) $\leftrightarrow$ D > 0.

- $\mathcal{PC}$ synchronize meet = first($\mathcal{PC}$) compound
  size@$_t$(first($\mathcal{PC}$)) − size$\mathcal{L}_t$(rest($\mathcal{PC}$)) @$_t$ rest($\mathcal{PC}$) $\leftrightarrow$
  size@$_t$(first($\mathcal{PC}$)) − size$\mathcal{L}_t$(rest($\mathcal{PC}$)) ≥ 0.

- $\mathcal{PC}$ synchronize mi = first($\mathcal{PC}$) compound
  size@$_t$(first($\mathcal{PC}$)) + size$\mathcal{L}_t$(first($\mathcal{PC}$)) @$_t$ rest($\mathcal{PC}$).

– $\mathcal{PC}$ synchronize overlap, D = first($\mathcal{PC}$) compound
size@$_t$(first($\mathcal{PC}$)) − D @$_t$ rest($\mathcal{PC}$) ↔
D $> 0 \wedge$ size$\mathcal{L}_t$(rest($\mathcal{PC}$)) − D < size$\mathcal{L}_t$(first($\mathcal{PC}$)).

– $\mathcal{PC}$ synchronize oi, D = first($\mathcal{PC}$) compound
size@$_t$(first($\mathcal{PC}$)) + D @$_t$ rest($\mathcal{PC}$) ↔
D $> 0 \wedge$ size$\mathcal{L}_t$(rest($\mathcal{PC}$)) + D > size$\mathcal{L}_t$(first($\mathcal{PC}$)).

– $\mathcal{PC}$ synchronize start = first($\mathcal{PC}$) compound
size@$_t$(first($\mathcal{PC}$)) @$_t$ rest($\mathcal{PC}$) ↔
size$\mathcal{L}_t$(rest($\mathcal{PC}$)) < size$\mathcal{L}_t$(first($\mathcal{PC}$)).

– $\mathcal{PC}$ synchronize si = first($\mathcal{PC}$) compound
size@$_t$(first($\mathcal{PC}$)) @$_t$ rest($\mathcal{PC}$) ↔
size$\mathcal{L}_t$(rest($\mathcal{PC}$)) > size$\mathcal{L}_t$(first($\mathcal{PC}$)).

– $\mathcal{PC}$ synchronize during, D = first($\mathcal{PC}$) compound
size@$_t$(first($\mathcal{PC}$)) + D @$_t$ rest($\mathcal{PC}$) ↔
D $> 0 \wedge$ size$\mathcal{L}_t$(rest($\mathcal{PC}$)) + D < size$\mathcal{L}_t$(first($\mathcal{PC}$)).

– $\mathcal{PC}$ synchronize di, D = first($\mathcal{PC}$) compound
size@$_t$(first($\mathcal{PC}$)) − D @$_t$ rest($\mathcal{PC}$) ↔
D $> 0 \wedge$ size$\mathcal{L}_t$(rest($\mathcal{PC}$)) + D > size$\mathcal{L}_t$(first($\mathcal{PC}$)).

– $\mathcal{PC}$ synchronize finish = first($\mathcal{PC}$) compound
size@$_t$(first($\mathcal{PC}$)) +
size$\mathcal{L}_t$(first($\mathcal{PC}$)) − size$\mathcal{L}_t$(rest($\mathcal{PC}$)) @$_t$ rest($\mathcal{PC}$) ↔
size$\mathcal{L}_t$(first($\mathcal{PC}$)) − size$\mathcal{L}_t$(rest($\mathcal{PC}$)) > 0.

– $\mathcal{PC}$ synchronize fi = first($\mathcal{PC}$) compound
size@$_t$(first($\mathcal{PC}$)) +
size$\mathcal{L}_t$(first($\mathcal{PC}$)) − size$\mathcal{L}_t$(rest($\mathcal{PC}$)) @$_t$ rest($\mathcal{PC}$) ↔
size$\mathcal{L}_t$(first($\mathcal{PC}$)) − size$\mathcal{L}_t$(rest($\mathcal{PC}$)) < 0.

– $\mathcal{PC}$ synchronize equal = first($\mathcal{PC}$) compound
size@$_t$(first($\mathcal{PC}$)) @$_t$ rest($\mathcal{PC}$) $\leftrightarrow$
size$\mathcal{L}_t$(rest($\mathcal{PC}$)) = size$\mathcal{L}_t$(first($\mathcal{PC}$)).

Ordre et profondeur des présentations

OQLiST fournit trois opérateurs qui permettent de manipuler l'ordre des éléments dans une présentation composite : inverse, move et swap. Nous présentons ici seulement la définition d'inverse.

– (**OST**, A $\sigma\tau_1$ B) inverse = (**OST**, B $\sigma\tau_2$ A) avec $\sigma\tau_1 = (r_x, r_y, r_t)$ et
$\sigma\tau_2 = (\text{inv}(r_x), \text{inv}(r_y), \text{inv}(r_t))$.

Autres définitions

– abs(r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = $\mathcal{P}_1 \leftrightarrow r \in$ {before, meet, overlap, start, di, fi}

abs(r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = $\mathcal{P}_2 \leftrightarrow r \in$ {bi, mi, oi, si, during, finish, equal}.

– comp$_i$($\mathcal{P}_1$, $\mathcal{P}_2$) = (r_i, $\mathcal{P}_1$, size$_{\delta_i}$($\mathcal{P}_2$) − size$_{\delta_i}$($\mathcal{P}_1$) @$_{\delta_i}$ $\mathcal{P}_2$) $\leftrightarrow$
$r \in$ {before, meet, overlap, start, di, fi} $\wedge$
r_i($\mathcal{P}_1$, size$_{\delta_i}$($\mathcal{P}_2$) − size$_{\delta_i}$($\mathcal{P}_1$) @$_{\delta_i}$ $\mathcal{P}_2$) $\wedge$ size$_{\delta_i}$($\mathcal{P}_1$) < size$_{\delta_i}$($\mathcal{P}_2$)

comp$_i$($\mathcal{P}_1$, $\mathcal{P}_2$) = (r_i, size$_{\delta_i}$($\mathcal{P}_1$) − size$_{\delta_i}$($\mathcal{P}_2$) @$_{\delta_i}$ $\mathcal{P}_1$, $\mathcal{P}_2$) $\leftrightarrow$
$r \in$ {bi, mi, oi, si, during, finish, equal} $\wedge$
r_i(size$_{\delta_i}$($\mathcal{P}_1$) − size$_{\delta_i}$($\mathcal{P}_2$) @$_{\delta_i}$ $\mathcal{P}_1$, $\mathcal{P}_2$) $\wedge$ size$_{\delta_i}$($\mathcal{P}_1$) > size$_{\delta_i}$($\mathcal{P}_2$).

– size(r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = size$_{\delta_i}$($\mathcal{P}_1$) + size$_{d_i}$($\mathcal{P}_1$) $\leftrightarrow r \in$ {bi, mi, oi, si}

size(r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = size$_{d_i}$($\mathcal{P}_1$) $\leftrightarrow r \in$ {di, fi}

size(r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = size$_{\delta_i}$($\mathcal{P}_2$) + size$_{d_i}$($\mathcal{P}_2$) $\leftrightarrow r \in$ {before, meet, overlap, start}

size(r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = size$_{d_i}$($\mathcal{P}_2$) $\leftrightarrow r \in$ {during, finish, equal}.

– getA((**OST**, A (r_x, r_y, r_t) B)) = (off(r_x, A, B) @$_x$ off(r_y, A, B)
@$_y$ off(r_t, A, B) @$_t$ A).

- getB($(\mathbf{OST}$, A (r_x, r_y, r_t) B)) = (off(inv(r_x), B, A) $@_x$ off(inv(r_y), B, A) $@_y$ off(inv(r_t), B, A) $@_t$ B).

- off(r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = size$_{\delta_i}(\mathcal{P}_1) \leftrightarrow$ abs(r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = $\mathcal{P}_1$

  off(r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = size$_{\delta_i}(\mathcal{P}_1)$ + size$_{\delta_i}(\mathcal{P}_2) \leftrightarrow$ abs(r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = $\mathcal{P}_2$.

- rels(D, r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = $\mathcal{P}_1$.relations $\leftrightarrow$ abs(r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = $\mathcal{P}_1$

  rels(D, r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = foldR$_i$(D, $\mathcal{P}_1$.relations) $\leftrightarrow$ abs(r_i, $\mathcal{P}_1$, $\mathcal{P}_2$) = $\mathcal{P}_2$.

- foldR$_i$(D, set()) = set()

  foldR$_i$(D, set($\mathcal{P}$ :: $\mathcal{P}$Set)) = set(D + size$_{\delta_i}(\mathcal{P})$ $@_{\delta_i}$ $\mathcal{P}$) + foldR$_i$(D, $\mathcal{P}$Set).

- inv(before$_i$) = bi$_i$
  inv(meet$_i$) = mi$_i$
  inv(overlap$_i$) = oi$_i$
  inv(start$_i$) = si$_i$
  inv(during$_i$) = di$_i$
  inv(finish$_i$) = fi$_i$
  inv(equal$_i$) = equal$_i$.

- foldA(set(), BSet) = set()

  foldA(set($\mathcal{P}$ :: ASet), BSet) = foldB($\mathcal{P}$, BSet) + foldA(ASet, BSet).

- foldB($\mathcal{P}$, set()) = set()

  foldB($\mathcal{P}_1$, set($\mathcal{P}_2$ :: BSet)) = set($\mathcal{P}_1$ compound $\mathcal{P}_2$) + foldB($\mathcal{P}_1$, BSet).

- rest($(\mathbf{OST}$, $(\mathbf{OST}_1$, $\mathcal{O}_1)$ $\sigma\tau$ $(\mathbf{OST}_2$, A $\sigma\tau_2$ B))) =
  getB($(\mathbf{OST}$, $(\mathbf{OST}_1$, $\mathcal{O}_1)$ $\sigma\tau$ $(\mathbf{OST}_2$, A $\sigma\tau_2$ B)))

  rest($(\mathbf{OST}$, $(\mathbf{OST}_1$, A $\sigma\tau_1$ B) $\sigma\tau$ C)) =
  rest(getA($(\mathbf{OST}$, $(\mathbf{OST}_1$, A $\sigma\tau_1$ B) $\sigma\tau$ C)))
  compound getB($(\mathbf{OST}$, $(\mathbf{OST}_1$, A $\sigma\tau_1$ B) $\sigma\tau$ C)).

Annexe B

Présentations

Description en régions de la carte de France

L'expression suivante définit une description d'une image de la carte de France. Il s'agit d'une présentation composite de sous-images chacune associée à un nom et à une région sur l'image original. Par exemple la sous-image correspondant à la région "Rhone - Alpes" est définie entre les points (223, 176) et (223 + 85, 176 + 91) de l'image FrenchMap :

```
define FrenchMap as Image( URL = "file://localhost/FrenchMap.gif" );

FrenchMap ~
         FrenchMap.crop( 8, 85, 93, 55 ).setName( "Bretagne" )
         at 8, 85
compound FrenchMap.crop( 80, 55, 70, 61 ).setName( "Basse Normandie" )
         at 80, 55
compound FrenchMap.crop( 128, 45, 42, 52 ).setName( "Haute Normandie" )
         at 128, 45
compound FrenchMap.crop( 160, 33, 68, 57 ).setName( "Nord - Picardie" )
         at 160, 33
compound FrenchMap.crop( 163, 6, 64, 42 ).setName( "Nord - Pas de Calais" )
         at 163, 6
compound FrenchMap.crop( 208, 38, 61, 99 ).setName( "Champagne - Ardennes" )
         at 208, 38
compound FrenchMap.crop( 244, 59, 64, 66 )
                  .setName( "Meurthe et Moselle - Meuse - Moselle - Vosges" )
         at 244, 59
```

```
compound FrenchMap.crop( 291, 76, 31, 63 ).setName( "Alsace" )
         at 291, 76
compound FrenchMap.crop( 60, 99, 87, 86 ).setName( "Nord - Pays de la Loire" )
         at 60, 99
compound FrenchMap.crop( 126, 86, 77, 99 ).setName( "Centre" )
         at 126, 86
compound FrenchMap.crop( 161, 76, 51, 40 ).setName( "Ile de France" )
         at 161, 76
compound FrenchMap.crop( 195, 106, 67, 84 ).setName( "Bourgogne" )
         at 195, 106
compound FrenchMap.crop( 256, 117, 46, 68 ).setName( "Franche Comte" )
         at 256, 117
compound FrenchMap.crop( 92, 153, 62, 77 ).setName( "Nord - Poitou Charente" )
         at 92, 153
compound FrenchMap.crop( 140, 180, 51, 58 ).setName( "Limousin" )
         at 140, 180
compound FrenchMap.crop( 176, 167, 71, 83 ).setName( "Auvergne" )
         at 176, 167
compound FrenchMap.crop( 223, 176, 85, 91 ).setName( "Rhone - Alpes" )
         at 223, 176
compound FrenchMap.crop( 74, 206, 87, 110 ).setName( "Aquitaine" )
         at 74, 206
compound FrenchMap.crop( 113, 233, 100, 92 ).setName( "Midi Pyrenees" )
         at 113, 233
compound FrenchMap.crop( 167, 234, 83, 101 )
                  .setName( "Languedoc - Roussillon" )
         at 167, 234
compound FrenchMap.crop( 235, 226, 91, 80 )
                  .setName( "Nord - Provence Cote dAzur" )
         at 235, 226
```

Espace de configurations

La figure B.1 suivante présente une énumération exhaustive des relations pouvant être établies entre deux intervalles dans un espace à deux dimensions (169 relations). Cet espace a été déterminé par [PTSE95, PT97] et nous l'utilisons pour montrer, en extension, l'espace de relations représentées par notre modèle.

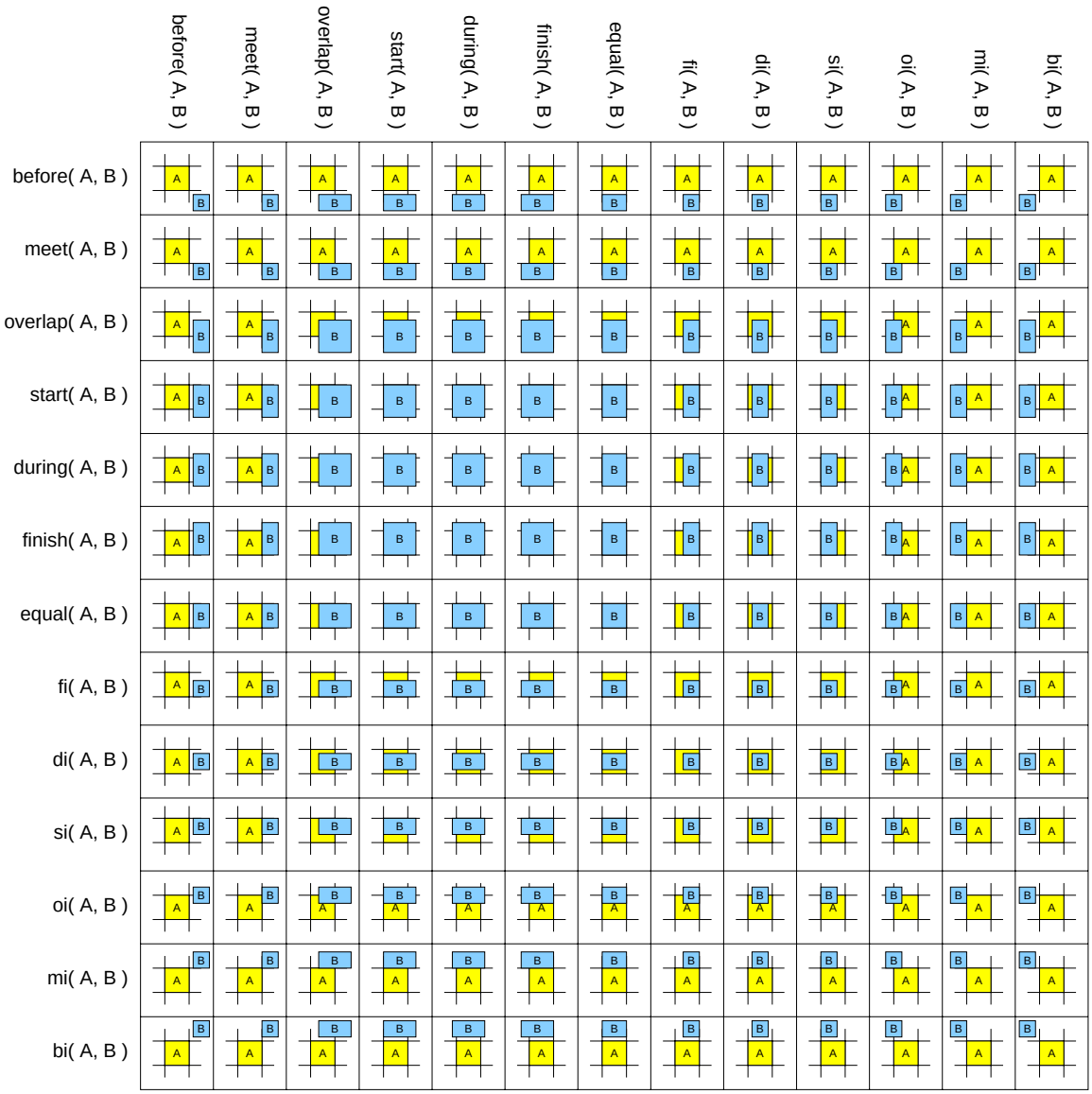


FIG. B.1 – *Espace de configurations*

Composition de tableaux et de médias

La figure B.2 présente une combinaison de tableaux et de médias. Cette présentation est composée de deux tableaux qui montrent des images d'activités de loisir que l'on peut réaliser dans des stations de ski. Ces tableaux sont associés à une image de la région où se trouvent les stations où ont peut réaliser ces activités. L'image est associée à un texte qui fait office de sous-titre. La présentation globale est un tableau composé des éléments (présentations) que nous venons d'énumérer. Nous expliquons dans la suite la construction de cette présentation en utilisant OQLiST.

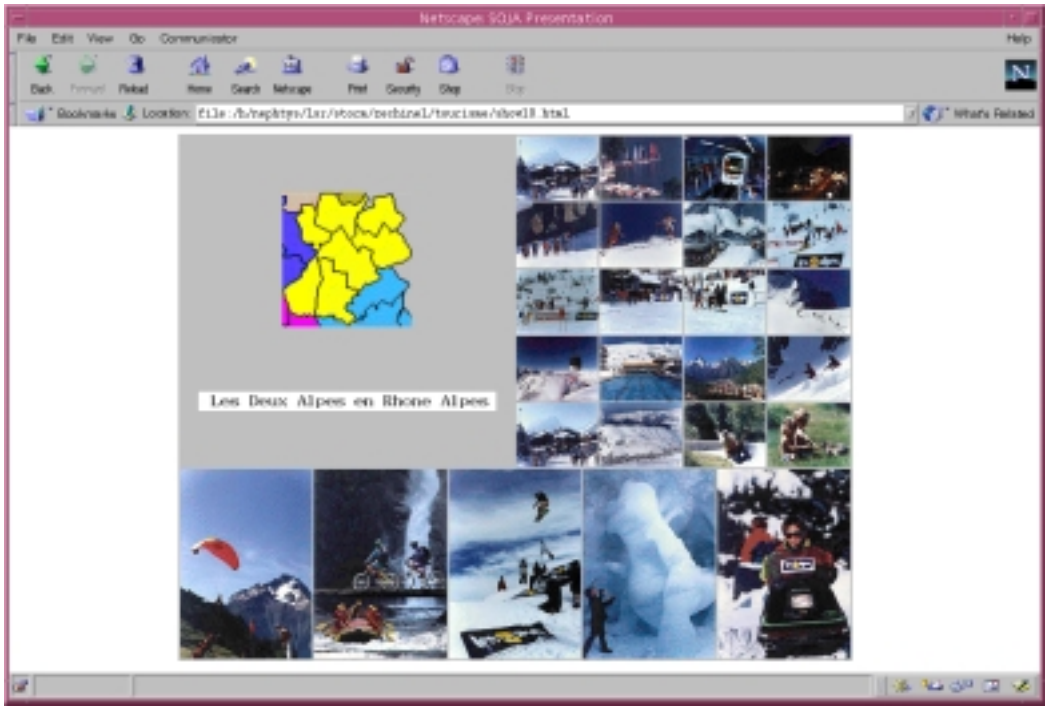


FIG. B.2 – *Stations de ski et activités*

Tableaux d'activités Les images concernant les activités de loisir sont disponibles dans deux répertoires locaux file://localhost/h et file://localhost/v. Nous avons décidé de créer deux tableaux nommées respectivement Table1 et Table2 contenant chacun des images stockés dans un des deux répertoires. Les expressions OQLiST suivantes définissent chacun de ces tableaux :

– Table1 organise les images dans quatre colonnes où chaque cellule a un cadre de 1×1 pixels et un bouton associé qui la reliant avec une présentation de l'image originale (cf. e.URL) :

```
define Table1 as
        atomic( select e button e.URL from e in
                select Image( URL = h ) from h in directory( "file://localhost/h" )
                table 4 border 1, 1 )
```

– Table2 est un tableau de 700×200 pixels à cinq colonnes où chacune de ses cellules a un cadre de 1×1 pixels et un bouton associé :

```
define Table2 as
        atomic( select e button e.URL from e in
                select Image( URL = v ) from v in directory( "file://localhost/v" )
                table 5 border 1, 1 with 700, 200 )
```

Image de la région L'expression suivante définit la présentation Region qui pointe vers un objet de type image. Cet objet est récupéré à partir des éléments contenus dans la description de la présentation de l'image de la carte de France (cf. chapitre 6) :

```
define Region as
        element( select e from e in descrip( FrenchMap ).elements
                   where ((Document) e.content).getName = "Rhone - Alpes"
                 ).content
```

Rappelons que chaque élément dans cette description est lui-même une présentation qui fait référence à une sous-image de l'image de la carte. Un tel élément a une méthode getName qui dénote le nom de la région en question. Le nom est utilisé comme critère de sélection pour trouver la présentation de l'image de la région requise, "Rhone - Alpes", dans notre exemple (cf. la clause where). Le résultat contient une présentation dont le contenu est nommé Region.

Région avec un sous-titre L'expression suivante permet de créer la présentation LogoRA. Elle compose la présentation d'une chaîne de caractères " Les Deux Alpes en

Rhone Alpes " avec celle de l'image Region. La présentation de la chaîne de caractères est centrée, au sud et 44 pixels disjoint de la présentation de Region :

```
define LogoRA as
    atomic( Region
        compound String( Value = " Les Deux Alpes en Rhone Alpes " )
            fit 2.3 * Region.getWidth, 2.3 * Region.getHeight
        disjoint 44, centered, south at 0, 0 )
```

Composition d'une présentation avec un tableau La présentation Table3 est un tableau de 700×350 pixels à deux colonnes. Cette définition associe la présentation LogoRA avec un cadre de 20×60 pixels et elle les compose avec Table1 :

```
define Table3 as
    atomic( LogoRA border 20, 60 compound Table1
        table 2 with 700, 350 )
```

Enfin, la présentation RhoneA est aussi un tableau qui contient une présentation composite qui a un cadre de 1×1 pixels et un arrière plan gris (background lightGray). Dans cette composition, Table3 et Table2 sont centrés, le second est au sud du premier et il n'y a aucune séparation entre eux (touch) :

```
define RhoneA as
    atomic( Table3 compound Table2 touch, centered, south )
    border 1, 1 background lightGray
```

Présentations enchaînées

Les expressions suivantes montrent la création de la présentation d'hyperliens présentée dans le chapitre 6. La présentation principale Regions est composée par un tableau d'hyperliens à quatre colonnes. Les cellules sont tous alignés à gauche avec une séparation de 20 pixels sur l'axe x. Chaque hyperlien point vers une des présentations North, South, West, East :

```
define Regions as
       String( Value = " Ile de France " ).setBackground( yellow ) fit 320, 50
       compound( select atomic( yellow compound e touch, centered, east )
                 from   e in list( String( Value = " NORTH " ) link North,
                                   String( Value = " SOUTH " ) link South,
                                   String( Value = " WEST "  ) link West,
                                   String( Value = " EAST "  ) link East )
                 table fit 4 fit 520, 38 arrange all meet + 20 on x )
       disjoint 80, centered, south at 0, 0
```

Une présentation, North, South, West ou East, est un tableau d'hyperliens dénotant les noms de régions qui sont placés au nord (sud, est, ouest) de l'Île de France. Chaque lien contenu dans ces tableaux est une présentation composite des images qui affiche la relation de l'Île de France avec une autre région. Dans cette présentation, chaque image ainsi que son nom ont un cadre de 2×2 pixels et un arrière plan noir. Les images sont des sous-images contenues dans la description de la présentation de la carte de France :

```
define North as
       atomic( select ((Document) r.A.content).getName
                      link( select e border 2, 2 background black
                                   compound ((Document) e.content).getName
                                            border 2, 2 background black
                                   touch, centered, south
                            from   e in r.inverse.elements order by *
                            background white )
               from   r in ( descrip( FrenchMap ).relations union
                             select r.inverse
                             from r in descrip( FrenchMap ).relations )
               where  ( ((Document) r.B.content).getName = "Ile de France" )
                      and ( r.Northwest or r.North or r.Northeast )
               table fit 1 arrange all meet + 20 on y )
       background white;

define South as
       atomic( select ((Document) r.A.content).getName
                      link( select e border 2, 2 background black
                                   compound ((Document) e.content).getName
                                            border 2, 2 background black
                                   touch, centered, south
```

```
                        from   e in r.inverse.elements order by *
                        background white )
                from   r in ( descrip( FrenchMap ).relations union
                        select r.inverse
                        from r in descrip( FrenchMap ).relations )
                where  ( ((Document) r.B.content).getName = "Ile de France" )
                       and ( r.Southwest or r.South or r.Southeast )
                table fit 1 arrange all meet + 20 on y )
        background white;

define West as
        atomic( select ((Document) r.A.content).getName
                       link( select e border 2, 2 background black
                                    compound ((Document) e.content).getName
                                             border 2, 2 background black
                                    touch, centered, south
                             from   e in r.inverse.elements order by *
                             background white )
                from   r in ( descrip( FrenchMap ).relations union
                        select r.inverse
                        from r in descrip( FrenchMap ).relations )
                where  ( ((Document) r.B.content).getName = "Ile de France" )
                       and ( r.Northwest or r.West or  r.Southwest )
                table fit 1 arrange all meet + 20 on y )
        background white;

define East as
        atomic( select ((Document) r.A.content).getName
                       link( select e border 2, 2 background black
                                    compound ((Document) e.content).getName
                                             border 2, 2 background black
                                    touch, centered, south
                             from   e in r.inverse.elements order by *
                             background white )
                from   r in ( descrip( FrenchMap ).relations union
                        select r.inverse
                        from r in descrip( FrenchMap ).relations )
                where  ( ((Document) r.B.content).getName = "Ile de France" )
                       and ( r.Northeast or r.East or  r.Southeast )
                table fit 1 arrange all meet + 20 on y )
        background white
```

Cadres de présentations

Toute présentation (simple ou composite) peut avoir un cadre associé (cf. chapitre 4). Lorsque deux présentations $\mathcal{P}_1$ et $\mathcal{P}_2$ sont composées, nous considérons chaque présentation et son cadre comme un atome.

Pour les exemples suivants, nous considérons le cas où $\mathcal{P}_1$ et $\mathcal{P}_2$ se touchent sur un de leurs côtés. Dans ce cas, la séparation entre le contenu de $\mathcal{P}_1$ et $\mathcal{P}_2$ est égal à la somme des deux cadres (cf. figure B.3). Considérons par exemple que les présentations $\mathcal{P}_1$ et $\mathcal{P}_2$ ont respectivement des cadres de taille différente D_1 et D_2. Nous définissons une présentation qui les organise dans un tableau à deux colonnes. Dans OQLiST, la séparation entre ces deux éléments est égal à la somme des tailles des cadres des cellules contiguës.

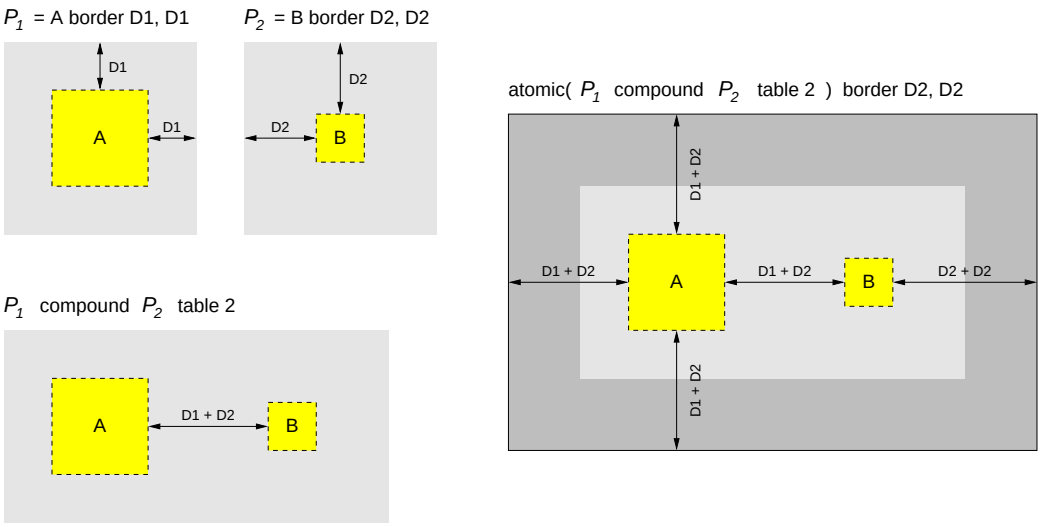


FIG. B.3 – *Composition de présentations avec des cadres*

Composition de médias dans un tableau OQLiST permet de gérer la taille des cadres des cellules d'un tableau. Ceci peut être utile pour définir des présentations où l'arrière plan est aussi une présentation (par ex. la présentation d'un vidéo). Les figures B.4 et B.5 présentent des tableaux avec des caractéristiques différentes présentant des images des planètes du système solaire. Ils ont comme arrière plan des vidéos explicatifs.

FIG. B.4 – *Le soleil et ses planètes*

Finalement, la figure B.6 illustre une composition des deux tableaux précédents. Notons que la taille des séparations en `x` et `y` entre les contenus des cellules des deux tableaux sont égaux.

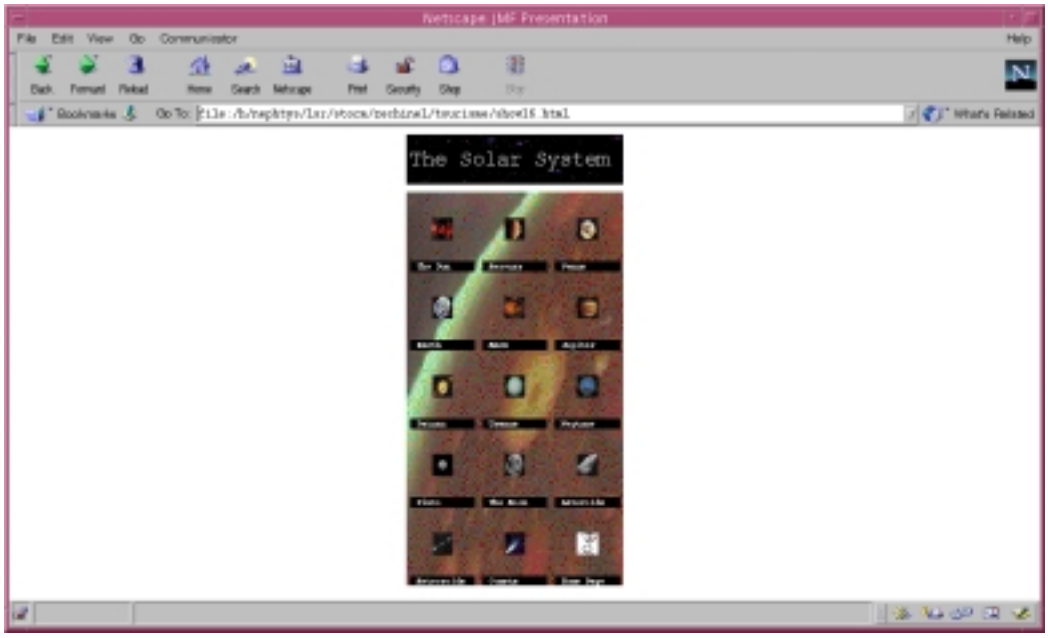


FIG. B.5 – *Le système solaire*

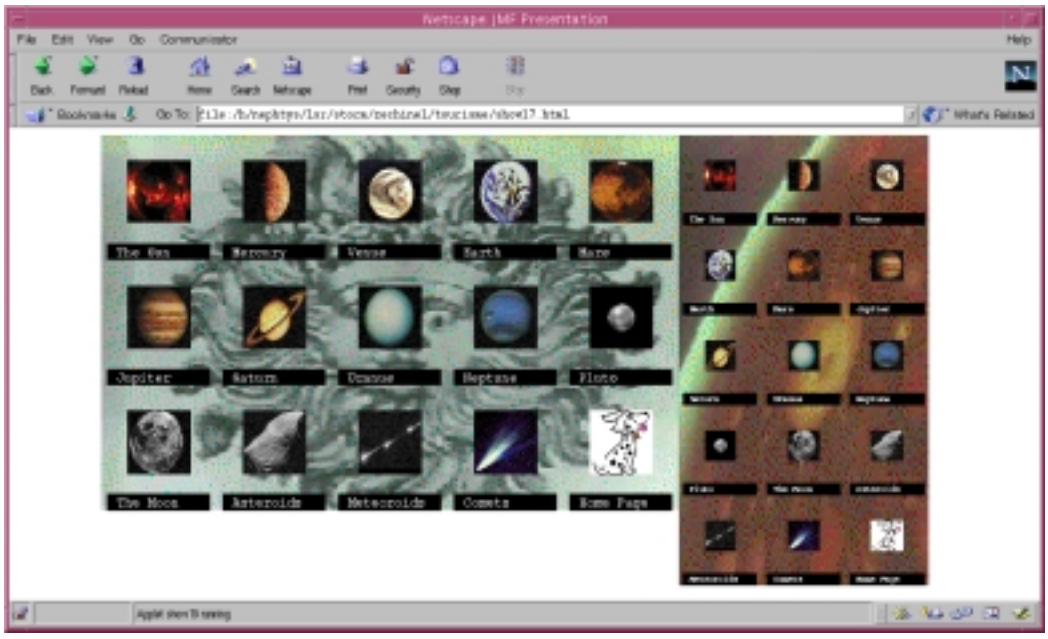


FIG. B.6 – *Composition des tableaux*