**MASTER EN SCIENCES INFORMATIQUES**

# Gestion électronique de document multimédia
# techniques de recherche et d'indexation

Feron, Michel; Delcorde, Paul

*Award date:*
2006

Faculté Universtaire Notre-Dame de la Paix
Institut d'informatique

# Gestion électronique de document multimédia : Techniques de recherche et d'indexation

Michel Feron
Paul Delcorde

Mémoire présenté en vue de l'obtention d'un grade de Maître en Informatique.
Année académique 2005-2006

## *Résumé*

De nos jours, la quantité d'information disponible au format électronique est impressionnante. Au vu de l'émergence actuelle des technologies de l'information et de la télécommunication, ce phénomène ne risque pas de s'atténuer. En particulier, les documents multimédia se font de plus en plus nombreux et nécessitent donc une gestion efficace. En effet, dans bien des domaines, il devient indispensable de posséder des outils permettant une recherche rapide et pertinente de documents multimédia. Bien que certains moteurs de recherche disponibles sur Internet proposent une telle recherche, celle-ci se base uniquement sur des critères externes au document (tels que l'auteur, le titre ou la date de création). Or, dans des domaines comme l'audiovisuel ou l'imagerie médicale, il serait utile de pouvoir rechercher des documents sur base de leur description (i.e. ce que l'on voit sur une photo ou ce que l'on entend sur un document sonore). Nous allons tout au long de ce mémoire aborder les principaux thèmes de la gestion électronique de documents multimédia. Nous nous intéresserons plus particulièrement aux différentes techniques permettant l'encodage, l'indexation et la récupération de ces documents.



Mots-clé : numérisation,encodage, indexation, base de données XML native, recherche par mots-clés, recherche par contenu, MPEG-1, MPEG-2, MPEG-4, MPEG-7, MPEG-21, MXF, Lucene.



## *Abstract*

Nowadays, more and more audio-visual information is available in digital form, in various places around the world, and the current emergence of information tachnology will steadily increase its use. More particularly, multimedia documents are more and more numerous, needing an efficient management. Indeed, in many fields, it is now essential to have tools for a fast and pertinenet search of multimedia documents. Many text-based engines are available on the World Wide Web, but their search is exclusiverly based on external features, like author, title or creation date. Nevertheless, fields like audio-visual applications and medical imagery could use a search of documents based on their content (i.e. what is seen on a picture or heard in an audio document). This dissertation will tackle the main themes of multimedia documents managing, with particular emphasis on the various coding, indexation and retrieval methods available for the documents.



Key words : numerization, coding, indexation, XML native database, search by keywords, search by content, MPEG-1, MPEG-2, MPEG-4, MPEG-7, MPEG-21, MXF, Lucene.

Nous tenons à remercier les différentes personnes qui nous ont soutenus durant la réalisation de ce mémoire ...

Tout d'abord, le professeur Jean-Marie Jacquet, pour ses précieux conseils et sa disponibilité.

Monsieur Alain Goossens pour nous avoir fait confiance dans ce projet.

Monsieur Francis Bodson pour nous avoir accueillis dans ses bureaux et pour sa nombreuse documentation.

Nous voudrions aussi exprimer notre gratitude envers nos familles et amis pour leur confiance et leur soutien, tout particulièrement envers nos pères respectifs pour leurs relectures et corrections orthographiques et grammaticales.

# Table des matières

**Introduction** **1**

**I Gestion électronique de documents** **5**

**1 Numérisation** **7**
1.1 Enjeux . . . . . 7
1.2 Les différents types de documents . . . . . 10
1.2.1 Les documents textuels . . . . . 10
1.2.2 Les documents sonores . . . . . 11
1.2.3 Les images . . . . . 12
1.2.4 La vidéo . . . . . 15
1.3 Références . . . . . 15

**2 Description de contenu** **17**
2.1 Evolutions techniques . . . . . 17
2.2 Méthodes . . . . . 19
2.3 Les principaux travaux . . . . . 20
2.3.1 Les archivistes et les bibliothèques . . . . . 21
2.3.2 Le monde de l'audiovisuel . . . . . 24
2.4 Références . . . . . 27

**II Numérisation** **29**

**3 MPEG-1** **31**
3.1 Historique . . . . . 31
3.2 Techniques de codage de l'image . . . . . 32

3.3 Méthode de compression vidéo . . . . . 33
3.3.1 Les images clés : I . . . . . 35
3.3.2 Les images prédites : P . . . . . 38
3.3.3 Les images bi-directionnelles : B . . . . . 39
3.4 Synchronisation son/vidéo . . . . . 41
3.5 Conclusion . . . . . 43

**4 MPEG-2** **47**
4.1 MPEG-2 Vs MPEG-1 . . . . . 47
4.2 Codage hiérarchique . . . . . 49
4.3 Profils et niveaux . . . . . 50
4.4 Débit fixe ou débit variable . . . . . 52
4.5 L'encodage double passe . . . . . 53
4.6 Conclusion . . . . . 54

**5 MPEG-4** **57**
5.1 Introduction . . . . . 57
5.2 Description et objectifs . . . . . 58
5.2.1 Notion d'objets média . . . . . 58
5.2.2 Description de scènes : BIFS . . . . . 60
5.3 Le codage visuel . . . . . 62
5.3.1 La vidéo naturelle . . . . . 63
5.3.2 Les images fixes et textures . . . . . 63
5.3.3 Les objets synthétiques . . . . . 65
5.4 Les droits de propriété intellectuelle . . . . . 66
5.5 Conclusions . . . . . 67

**III Description de contenu** **69**

**6 Material eXchange Format** **71**
6.1 Introduction . . . . . 71
6.2 Les apports du MXF . . . . . 74
6.3 Description du format . . . . . 77
6.4 Conclusion . . . . . 82

**7 MPEG-7** **85**
7.1 Introduction . . . . . 85
7.2 Principe de MPEG-7 . . . . . 88
7.3 Principales fonctionnalités . . . . . 90
7.3.1 Le DDL . . . . . 90
7.3.2 Le MPEG-7 Visual . . . . . 101
7.3.3 Le MPEG-7 Audio . . . . . 109

7.3.4 Les schémas de description multimédia . . . . . 111
7.4 Conclusion . . . . . 117

**8 MPEG-21** **119**
8.1 Introduction . . . . . 119
8.2 Le modèle “User” . . . . . 120
8.3 Le Digital Item . . . . . 120
8.4 Contenu de MPEG-21 . . . . . 121
8.4.1 MPEG-21 partie 1 : Vision, Technologies et Stratégie . . . . . 121
8.4.2 MPEG-21 partie 2 : Digital Item Declaration . . . . . 121
8.4.3 MPEG-21 partie 3 : Digital Item Identification . . . . . 125
8.4.4 MPEG-21 partie 4 : Intellectual Property Management and Protection . . . . . 128
8.4.5 MPEG-21 partie 5 : Right Expression Language . . . . . 128
8.4.6 MPEG-21 partie 6 : Rigts Data Dictionary . . . . . 130
8.4.7 MPEG-21 partie 7 : Digital Item Adaptation . . . . . 131
8.4.8 MPEG-21 partie 8 : Reference Software . . . . . 133
8.4.9 MPEG-21 partie 9 : File Format . . . . . 133
8.4.10 MPEG-21 partie 10 : Digital Item Processing . . . . . 133
8.4.11 MPEG-21 partie 11 : Evaluation Methods for Persistent Association Technologies . . . . . 135
8.4.12 MPEG-21 partie 12 : Test Bed for MPEG-21 Resource Delivery 135
8.4.13 MPEG-21 partie 13 : Scalable Video Coding . . . . . 135
8.4.14 MPEG-21 partie 14 : Conformance Testing . . . . . 135
8.4.15 MPEG-21 partie 15 : Event Reporting . . . . . 136
8.4.16 MPEG-21 partie 16 : Binary Format . . . . . 137
8.4.17 MPEG-21 partie 17 : Fragment Identification for MPEG Ressources . . . . . 138
8.4.18 MPEG-21 partie 18 : Digital Item Streaming . . . . . 138
8.5 Références . . . . . 138

**IV Multimedia Document Manager** **139**

**9 Description du Multimedia Document Manager** **141**

**10 La production de descripteurs** **149**
10.1 Les descripteurs sémantiques . . . . . 151
10.1.1 L’encodeur sémantique . . . . . 151
10.1.2 Grammaire des descripteurs sémantiques . . . . . 152
10.1.3 Appréciation des descripteurs sémantiques . . . . . 155
10.2 Les descripteurs visuels . . . . . 155
10.2.1 Les histogrammes de couleurs . . . . . 156

10.2.2 Color Layout . . . . . 164
10.2.3 Dominant Color . . . . . 166
10.3 Références . . . . . 166

**11 Stockage des descriptions MPEG-7 169**
11.1 Introduction . . . . . 169
11.2 Types de bases de données supportant XML . . . . . 171
11.2.1 Les bases de données relationnelles étendues XML . . . . . 171
11.2.2 Les bases de données orientées objets . . . . . 178
11.2.3 Les base de données XML natives . . . . . 180
11.3 Le choix des bases de données XML natives . . . . . 185
11.4 Références . . . . . 188

**12 L'indexation et la recherche de document 189**
12.1 Lucene . . . . . 191
12.1.1 Format de stockage des indexes . . . . . 191
12.1.2 La recherche . . . . . 195
12.2 LIRE . . . . . 198
12.3 L'indexation des bases de données XML natives . . . . . 198
12.3.1 Introduction . . . . . 198
12.3.2 Les type d'indexation . . . . . 200
12.3.3 L'indexation avec Berkeley DB XML . . . . . 202
12.4 La recherche via XQuery . . . . . 205
12.5 Références . . . . . 209

**Conclusion 211**

# Table des figures


1.1 Signal analogique VS Signal numérique . . . . . 11
1.2 Echantillonage d'un signal analgique . . . . . 12
1.3 Représentation vectorielle Vs représentation matricielle . . . . . 14

3.1 Format RGB . . . . . 32
3.2 Codage YUV . . . . . 34
3.3 Structure hiérarchique d'une séquence vidéo MPEG-1 . . . . . 34
3.4 Compression d'une image I codée en YUV 4 : 2 : 0 . . . . . 36
3.5 Résultat de la quantification et méthode zig zag . . . . . 37
3.6 Exemple où les images P sont utiles . . . . . 38
3.7 Un signal basique MPEG-1 . . . . . 39
3.8 Exemple de suite d'images MPEG-1 . . . . . 40
3.9 Composition d'un flux MPEG-1 . . . . . 40
3.10 Multiplexage d'une séquence vidéo . . . . . 41
3.11 Démultiplexage d'une séquence vidéo . . . . . 42
3.12 Marqueurs PTS et DTS . . . . . 43
3.13 Image entrelacée . . . . . 44

4.1 Illustration du data partitionning . . . . . 50
4.2 Table de niveaux et profils MPEG-2 . . . . . 52
4.3 Système d'encodage double passe . . . . . 53

5.1 Description d 'une scène MPEG-4 . . . . . 59
5.2 Description d'une scène au format *BIFS* . . . . . 61
5.3 Concept de base du codage *MPEG-4* . . . . . 63
5.4 Compression par ondelettes . . . . . 64
5.5 Technique de maillage 2D/3D . . . . . 66

6.1 Principales exigences de MXF . . . 73
6.2 Etapes classiques de création d'un programme de TV . . . 74
6.3 Exemple d'une métadonnée GPS . . . 75
6.4 Les différentes parties du MXF . . . 78
6.5 Structure KLV d'un fichier MXF . . . 78
6.6 Structure d'un fichier MXF . . . 79
6.7 Packages et pistes d'un fichier MXF . . . 80
6.8 Grille des Operational Patterns . . . 81

7.1 Recherche d'image avec le moteur Google . . . 86
7.2 Portée de MPEG-7 . . . 87
7.3 Eléments principaux de MPEG-7 . . . 88
7.4 Exemple d'application utilisant MPEG-7 . . . 89
7.5 Jeu de balises *XML* ouvrante - fermante . . . 91
7.6 Balise vide . . . 92
7.7 Exemple de document XML . . . 92
7.8 Document XML . . . 94
7.9 Document XML Schema pour la gestion de bon de commande . . . 95
7.10 Prologue d'un schéma *XML* . . . 96
7.11 Déclaration d'éléments . . . 96
7.12 Déclaration d'un type complexe avec séquence . . . 97
7.13 Définition d'un type complexe avec séquence . . . 97
7.14 Définition d'un type simple . . . 98
7.15 Le type liste . . . 98
7.16 Le type union . . . 99
7.17 Les facettes XML Schema . . . 99
7.18 Extensions MPEG7 XML Schema : les matrices . . . 100
7.19 Syntaxes DDL d'un Grid Layout . . . 101
7.20 Classification de la régularité d'une texture . . . 104
7.21 Types de contours de l'histogramme des contours . . . 105
7.22 Formes constituées d'une zone unique, de trous ou des zones disjointes 106
7.23 Représentation de formes similaires . . . 106
7.24 Extraction du contour d'une forme d'une image . . . 107
7.25 Caractéristiques du descripteur de contour . . . 107
7.26 Mouvements basiques de caméra . . . 108
7.27 Descripteurs audio *MPEG-7* de bas niveau . . . 110
7.28 Vue d'ensemble de l'organisation des MDS *MPEG-7* . . . 112
7.29 Caractéristiques spécifiques pour la description d'un segment . . . 113
7.30 Description d'une image par des segments de Régions Immobiles . . . 114
7.31 Exemple de description combinant l'aspect structurel et sémantique . 115

8.1 Relations entre les principaux éléments du modèle DID . . . 124
8.2 Relation entre DI Declaration et DI Identification . . . 126

8.3 Métadata et identifiant d'un album de musique MPEG-21 . . . . . 127
8.4 Le modèle REL . . . . . 129
8.5 Architecture conceptuelle d'un adaptateur de DI . . . . . 132
8.6 Exemple de format de fichier MPEG-21 . . . . . 134
8.7 Modèle de test de conformance . . . . . 136

9.1 Transformation des informations en descripteur *MPEG-7* . . . . . 142
9.2 Descripteur *MPEG-7*de couleur dominante calculé par l'API *Caliph* . . 143
9.3 "Production-Stockage-Indexation" de la solution "full-*XML*" . . . . . 144
9.4 "Production-Stockage-Indexation" de la solution Lucene . . . . . 144
9.5 Chaîne de traitement "Recherche sémantique-Présentation" . . . . . 145
9.6 "Recherche par similarité-Présentation" de la solution "full-*XML*" . . . 146
9.7 "Recherche par similarité-Présentation" de la solution Lucene . . . . . 147

10.1 Exemple de méta-balise HTML . . . . . 150
10.2 L'encodeur sémantique . . . . . 151
10.3 Le descripteur `Term` . . . . . 153
10.4 Le schéma de description `TextAnnotation` . . . . . 153
10.5 Le schéma de description `Creator` . . . . . 153
10.6 Le schéma de description `MediaLocator` . . . . . 154
10.7 Exemple complet d'une description sémantique . . . . . 154
10.8 Exemple de thésaurus . . . . . 155
10.9 Nuances de couleur . . . . . 157
10.10Hexacone représantant l'espace de couleur HSV . . . . . 158
10.11Etablissement d'un histogramme dans l'espace de couleur RGB . . . . 161
10.12Architecture de la recherche par histogramme de couleur . . . . . 163
10.13Technique d'extraction du Color Layout . . . . . 165
10.14Description XML MPEG-7 des descripteurs de couleurs . . . . . 167

11.1 Exemple d'arbre XML . . . . . 170
11.2 Intégration des documents XML . . . . . 172
11.3 Fichier XML dont la structure va être éclatée en tables . . . . . 173
11.4 Table des fichiers . . . . . 173
11.5 Tables de éléments . . . . . 173
11.6 Table des attributs . . . . . 174
11.7 Table des valeurs . . . . . 174
11.8 Stockage structuré de document XML . . . . . 175
11.9 Exemple de fichier XML pour le mapping table-based . . . . . 176
11.10Exemple de fichier XML pour le mapping object-relational . . . . . 179
11.11Exemple d'une requête FLWR . . . . . 183

12.1 Fonctionnement d'un moteur de recherche de page Web . . . . . 190
12.2 Les éléments clés de Lucene . . . . . 191
12.3 Champs différents comportant le même terme . . . . . 192

12.4 Création d'un index Lucene . . . . . 194
12.5 Création d'un index Lucene . . . . . 195
12.6 Exemple de requêtes Lucene valide . . . . . 196
12.7 Recherche sémantique d'une image avec Lucene . . . . . 197
12.8 Exemple de B-tree . . . . . 200
12.9 Division d'un noeud feuille . . . . . 201
12.10 Plan de numérotation DLN . . . . . 203
12.11 Exemple de description basée MPEG-7 . . . . . 206
12.12 Exemple d'une requête FLWR . . . . . 207
12.13 Prototype d'une fonction en XQuery . . . . . 208

# Introduction

De nos jours, la quantité de documents électroniques est devenue impressionnante. Au vu de l'émergence actuelle des technologies de l'information et de la télécommunication, ce phénomène ne risque pas de s'atténuer. En particulier, les documents multimédia font de plus en plus partie intégrante de notre vie. Il suffit de penser aux mots "à la mode" de ce début de $21^{ieme}$ siècle : appareil photo numérique, MP3, TV à la demande, ADSL, etc. Nous assistons donc à l'heure actuel à une révolution numérique et multimédia. En effet, l'avènement de ces technologies et les changements sociétaux qui en découlent génèrent dans le public de nouvelles habitudes de consommation des médias. Devant l'immensité du nombre de documents disponibles, il devient important (voire même crucial) de développer des systèmes permettant de récupérer les documents à l'aide de simples requêtes. Prenons l'exemple d'une site Internet mettant à disposition de ses clients des milliers de photos. Sans un outil de recherche, un utilisateur cherchant une photo représentant un coucher de soleil devrait parcourir l'ensemble des photos mises à sa disposition avant de trouver celle qu'il désire. A contrario, il lui suffirait d'indiquer "coucher de soleil" dans un champ prévu à cet effet pour retrouver les photos correspondant à ce critère. Pour que cette recherche soit pertinente et rapide, il faut au préalable avoir recours à l'indexation.

L'indexation n'est pas apparue avec l'informatique. Depuis des siècles, des bibliothécaires et documentalistes exercent cette pratique en utilisant des outils tels que les thésaurus, les lexiques d'indexation, etc. La problématique fondamentale de l'indexation n'est donc pas nouvelle : des concordances bibliques aux moteurs de recherche actuels, le problème est toujours le même : comment représenter le contenu d'un document, comment le repérer et le retrouver.

A travers ce mémoire, nous tenterons de répondre à ces trois questions en nous focalisant sur la manière dont un document multimédia peut être indexé et récupéré.

Avant de s'attaquer aux problèmes liés à l'indexation et à la recherche de documents multimédia, il est utile de s'intéresser à la manière dont ils sont encodés. Comme nous le verrons dans la première partie de ce mémoire, le terme "multimédia" se réfère à plusieurs types d'information numérisée : le texte, les images (fixes ou animées) et le son. Afin de bien comprendre les problématiques liées à la gestion documentaire informatisée, nous tenterons également de cerner les enjeux de la numérisation et établirons un premier aperçu des différents standards et normes de description de contenu de documents multimédia.

Dans la deuxième partie de ce mémoire, nous étudierons les trois standards de compression audiovisuelle les plus utilisés à savoir le *MPEG-1*, le *MPEG-2* et le *MPEG-4*. En effet, derrière ces standards se cachent (souvent à l'insu de la majorité des gens) de nombreuses technologies utilisées couramment tels que le DVD, la télévision numérique, la vidéo-conférence, le streaming, etc.

Jusqu'il y a peu, les médias audiovisuels étaient considérés comme des objets non décomposables. On voit apparaître aujourd'hui de plus en plus d'outils permettant de manipuler ces informations avec une granularité plus fine. Dans le domaine de l'indexation et de la recherche par le contenu, l'objectif est de permettre aux utilisateurs d'accéder à la structure interne des contenus audiovisuels en se basant sur des descriptions. On pourrait imaginer un moteur de recherche permettant de trouver une scène ou une mélodie particulière dans une base de données audiovisuelles. Pour ce faire, il est nécessaire d'avoir des modèles permettant de définir la syntaxe des descriptions mais aussi leur mode de représentation. C'est ce que nous étudierons dans la troisième partie de ce mémoire à travers deux standards de description de documents multimédia, *MXF* et plus particulièrement *MPEG-7*. Dans cette partie, nous expliquerons également le fonctionnement des langages *XML* et *XML* Schema sur lesquels se base le *MPEG-7*. Ces derniers s'avèrent en effet très intéressants dans le cadre de la production de descriptions multimédia.

Afin de mettre en pratique les techniques et standards étudiés, nous avons, dans la quatrième et dernière partie de ce mémoire, mis au point deux architectures permettant l'indexation et la recherche de photos numériques. La première sera construite autour du moteur d'indexation et de recherche Lucene de la fondation Apache tandis que la deuxième sera réalisée entièrement sur base de technologies liées au *XML* telles que les base de données *XML* Native et l'interpréteur de requêtes XQuery. Deux types de recherche seront proposés, la recherche sémantique et la recherche par contenu. La première approche consiste à interroger le système à l'aide de mots-clés comme c'est le cas dans les moteurs de recherche courants (par exemple, Google ou MSN). La seconde approche, la recherche sur le contenu, permet de récupérer les documents sur base de caractéristiques visuelles. Elle permettra, par exemple, à un utilisateur de retrouver des images similaires (selon un coefficient de similarité défini) à l'image qu'il donne en entrée. Dans cette partie, nous analyserons les moyens à mettre en oeuvre pour réaliser de telles architectures et ce de la production de description *XML* jusqu'à la recherche d'image en passant par l'étude de l'extraction des caractéristiques visuelles d'une image et l'indexation des descriptions.

# Première partie

# Gestion électronique de documents

CHAPITRE 1

# Numérisation

## 1.1 Enjeux

A l'époque de la gestion manuelle des documents, les entreprises étaient confrontées à de nombreux problèmes. Au fil des années les entreprises accumulaient quantité de documents qu'il fallait classer et stocker. Pour que ces document puissent être conservés le plus longtemps possible en bon état, suivant la nature de leur support, ils devaient être stockés dans certaines conditions d'éclairage, d'humidité, de température, etc. Pas question donc d'entreposer ses document dans les sous-sol de l'entreprise, il fallait aménager une pièce prévue spécialement à cet effet, ce qui représentait un coût non négligeable pour l'entreprise. Sans compter qu'une fois l'espace de stockage rempli, l'entreprise devait soit détruire les document les plus anciens, soit réaménager un autre espace de stockage.

Avec un tel système, la recherche de documents était synonyme d'une longue et fastidieuse recherche dans une masse considérable de documents. Le temps de recherche était directement proportionnel à la qualité de la politique de classement des documents. Il était impossible, dans un tel système, que plusieurs personnes aient accès en même temps au même document. Chacun devait attendre que le précédent en termine avec le document avant de pouvoir se l'approprier, ce qui pouvait entraîner des retards dans le travail de chacun. Sans compter que la perte d'un document signifiait sa perte définitive pour l'entreprise.

Ces différents problèmes, ainsi que l'avènement de l'ère numérique, ont poussé les entreprises à abandonner le caractère physique des documents pour un caractère numérique et à passer à une gestion électronique de leurs documents. Ce système résout les problème de stockage, de classement, améliore les recherches, permet un accès concurrentiel aux documents, limite les risque de perte, ...

Cependant comme tous les systèmes, la gestion électronique amène aussi son lot de problèmes. Dans un premier temps il faut numériser tous les documents, ce qui peut prendre un temps considérable. Pour cette raison, beaucoup d'entreprises utilisent un système hybride où cohabitent documents sur support physique et documents numériques. Pour que les recherches soient efficaces, il faut décrire de façon la plus exhaustive possible le contenu de chaque document. Cette tâche s'avère d'autant plus complexe que le document est complexe. On peut aussi citer les problèmes liés à la gestion de droits associés à chaque document, l'authenticité des documents, ...

Au delà des enjeux évidents cités ci-dessus, l'enjeu majeur de la numérisation réside actuellement dans la conservation de notre patrimoine audiovisuel. Celui-ci est constitué de l'ensemble de nos photographies, de nos documents sonores, des nos films, ..., acquis depuis que l'on est capable de les sauvegarder sur des supports "durables". Bien que cela soit peu connu du grand public, la conservation de ce patrimoine est l'enjeu principal du monde audiovisuel.

Dès l'avènement des premiers supports analogiques, on s'est mis à penser que nos documents audiovisuels, émissions de télévision et de radio notamment, accédaient à une nouvelle sorte d'éternité. A l'heure actuelle il n'en est rien. Images, sons, films, vidéos ou bandes magnétiques sont tous actuellement confrontés au caractère éphémère de leur support. Usées par le temps, rongées par l'acide, bientôt illisibles faute de lecteurs adéquat, détruites lors de catastrophes naturelles, ..., nos archives audiovisuelles de par le monde sont en danger. Cependant cette tragédie n'est pas inéluctable.

En effet, grâce à la numérisation, un pan gigantesque de notre patrimoine culturel pourrait être sauvé. Malheureusement la tâche à accomplir est colossale. A l'heure actuelle l'UNESCO estime à 200 millions d'heures le patrimoine audiovisuel mondial. Si l'on ne fait rien, on considère que d'ici une dizaine d'années quatre-vingt pour cent de cette mémoire mondiale aura disparu. Aucun pays, même les plus avancés en matière de conservation et de numérisation, n'est véritablement à l'abri de cette lente destruction. Partout, faute de moyens, ou pire, d'intérêt, des milliers d'heures de radio et de télévision risquent de s'éteindre pour toujours.

Même si la menace est à l'échelle planétaire, toutes les régions du monde ne sont pas frappées avec la même intensité. En effet le numérique et ses nouveaux supports se sont très vite répandus dans nos sociétés industrialisées, alors que dans

bon nombre de pays le numérique est encore absent. Les principales régions en péril sont l'Amérique du Sud, l'Afrique, le Proche et le Moyen-Orient et l'Asie du sud-est. Alors que de nombreux pays européens comme la Grande-Bretagne, l'Italie, la France, l'Allemagne, la Suède, la Norvège ou encore la Finlande ont déjà mis en place des processus de sauvegarde, pour les pays les plus pauvres du sud, la messe semble être déjà dite. Ces pays assistent impuissants à la disparition de leur patrimoine audiovisuel.

Dans nos pays européens, la numérisation de nos archives se heurte encore à une prise de conscience trop lente. Pour les autres formes de culture telles que la littérature, la peinture, la sculpture, l'architecture, ..., les dégradations sont visibles. Lorsque la pollution attaque nos monuments ou nos sculptures, que nos peintures voient leurs couleurs s'estomper, ..., tout le monde trouve légitime de restaurer ce patrimoine, tandis que notre mémoire audiovisuelle subit une dégradation invisible et inaudible pour le grand public. En effet, les image abîmées ne sont, par essence, jamais montrées. Alors que des milliers de boîtes entassées dans des étagères meurent en silence à l'abri des regards, les télévisions diffusent des flux ininterrompus de nouvelle images, nous faisant ainsi oublier la lente agonie de milliers d'heures d'enregistrement.

Aujourd'hui la numérisation est la solution idéale pour sauver notre patrimoine audiovisuel, mais elle nous apporte bien plus. Grâce à la déconnexion physique entre le support et l'utilisateur via les réseaux, le numérique peut s'immiscer dans de nombreux domaines artistiques, culturels, pédagogiques, médiatiques, etc. Donc à l'heure actuelle la conservation matérielle de nos archives audiovisuelles a autant d'importance que la restauration de nos cathédrales, peintures, sculptures et autres formes d'expression culturelle.

Un autre élément qui représente un frein à la numérisation des documents audiovisuels est l'effort financier à fournir. Dans ce domaine on se trouve face à deux problématiques distinctes. Soit les ressources sont insuffisantes car les solutions technologiques actuelles restent relativement chères. C'est notamment le cas pour les pays du Sud. L'enjeu est alors de partager les moyens de façon à rendre plus abordable la numérisation. Soit les ressources sont présentes et c'est le retour sur investissement qui freine les décideurs. C'est notamment le cas aux États-Unis qui n'ont pas encore lancé de plan de numérisation de leurs ressources analogiques. A leurs yeux ces archives audiovisuelles n'ont pas une valeur commerciale justifiant de telles dépenses.

Malheureusement le volet financier n'est pas le seul écueil de taille que rencontrent les projets de numérisation. La formation inadaptée de nos décideurs politiques actuels fait que ceux-ci ont du mal à s'inscrire dans une politique à long terme où les bénéfices des actions prises se ressentent une quinzaines d'années plus tard. En effet la préservation de notre patrimoine audiovisuel nécessite bien plus

de temps que celui couvert par un mandat politique. Les dix ans restant à vivre à la majeure partie du patrimoine audiovisuel semblent encore laisser une bonne marge de manoeuvre et de réflexion, mais il n'en est rien, l'ultime limite a d'ores et déjà été atteinte.

## 1.2 Les différents types de documents

### 1.2.1 Les documents textuels

Les documents textuels peuvent se présenter sous forme manuscrite ou sous forme imprimée. Les documents imprimés ont été créés via un programme de traitement de texte puis imprimés sur papier. En général, avant d'être imprimé, le texte a été sauvegardé sous forme numérique. Dès lors, seuls les documents manuscrits ou les documents imprimés dont on n'a plus la version numérique nécessitent une numérisation.

La méthode la plus simple pour la numérisation d'un document textuel est de le numériser sous forme d'image via un scanner. On va se contenter de numériser chaque page du document. On pourra ensuite consulter le document en lisant les différentes images comme on le ferait sur papier. L'inconvénient de cette méthode est que l'on ne peut pas manipuler le contenu du texte (modification via traitement de texte, extraire de l'information, ...) une fois celui-ci numérisé. Pour cela il faut utiliser la reconnaissance optique de caractères.

Numériser le document avec un logiciel de reconnaissance optique de caractère (ROC) permet d'obtenir un document électronique textuel qui pourra être ensuite manipulé par un programme de traitement de texte. Ce logiciel utilise un fichier image représentant le texte. Il analyse alors les dessins des caractères et permet de construire un fichier au format texte. Cette opération se révèle être assez complexe et le taux d'erreur est encore assez élevé. La reconnaissance n'est pas parfaite et bien entendu les résultats sont meilleurs avec les documents imprimés qu'avec les documents manuscrits. Lorsque l'ordinateur hésite sur un caractère particulier, il le surligne pour qu'un opérateur puisse ensuite relire le texte et indiquer la lettre non reconnue. Plus rarement, l'ordinateur peut substituer une lettre à une autre. C'est pour cela qu'un contrôle doit toujours être effectué par un opérateur après la reconnaissance automatique. Le taux de réussite des programmes d'OCR avoisine les 99% pour les documents imprimés mais cela n'est pas encore assez. Un taux de 99% signifie qu'on aura une quizaine de faute dans le document produit.

### 1.2.2 Les documents sonores

Les représentations analogique ou numérique d'un signal sonore sont deux procédés servant au transport et au stockage de données. L'analogique est apparu au début de l'électricité alors que le numérique est né plus récemment avec l'avènement de l'informatique.

L'analogique reproduit le signal sonore d'entrée sous la même forme sur le support de sortie (magnétique en général). Lorsque l'on veut enregistrer un signal sonore avec un système analogique, l'onde sonore est reproduite le plus fidèlement possible sur la bande magnétique. Les variations de l'onde sonore sont traduites en variations d'intensité du signal électrique. Voir figure 1.1.

Lorsque l'on convertit un signal analogique en signal numérique à l'aide d'un convertisseur analogique-numérique, le signal obtenu est une suite de 0 et de 1, autrement dit un signal qui n'a que deux amplitudes possibles. Voir figure 1.1.

Le signal numérique, contrairement au signal analogique, peut être transmis et/ou copié sans perte d'information. En effet, alors que l'amplitude du signal analogique doit varier fidèlement à l'original, le signal numérique varie entre une amplitude forte et une amplitude faible. Si le signal électrique est perturbé, le système pourra toujours faire la distinction entre signal fort ou faible.

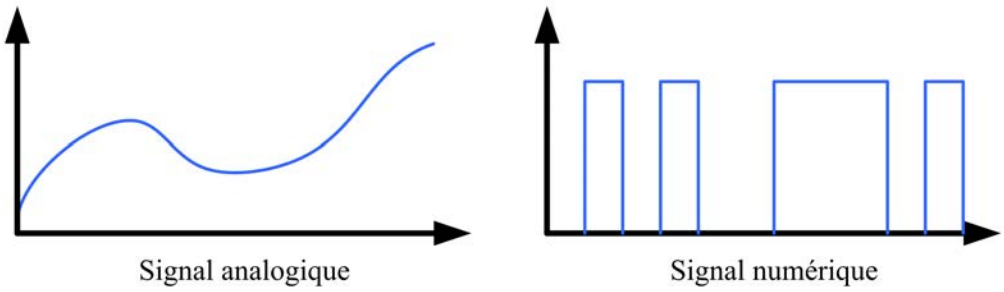


FIG. 1.1 – *Signal analogique VS Signal numérique*

Lorsque l'on numérise un signal analogique, on le convertit en signal numérique. La numérisation se déroule en deux étapes qui sont l'échantillonnage et la quantification. Lors de l'échantillonnage, on va prélever à intervalles de temps réguliers des échantillons du signal analogique. La quantification consiste à assigner à chaque échantillon une valeur numérique dépendant du nombre de bits sur lesquels on va encoder le signal.

La qualité du signal numérique dépend de deux facteurs :

– La fréquence d'échantillonnage ou taux d'échantillonnage : plus la fréquence d'échantillonnage est élevée, plus les intervalles de temps entre chaque échantillon sont courts, plus le signal numérique sera fidèle à l'original et meilleure sera la qualité. Voir figure 1.2.

– La résolution : elle correspond au nombre de bits utilisés pour encoder la valeur de chaque échantillon, c'est-à-dire le nombre de valeurs qu'un échantillon peut prendre. Si on utilise 16 bits, chaque échantillon pourra prendre une valeur entre 0 et $2^{16} = 65535$. Plus la résolution est élevée, plus les variations d'amplitude peuvent être grandes et moins on perd d'information.

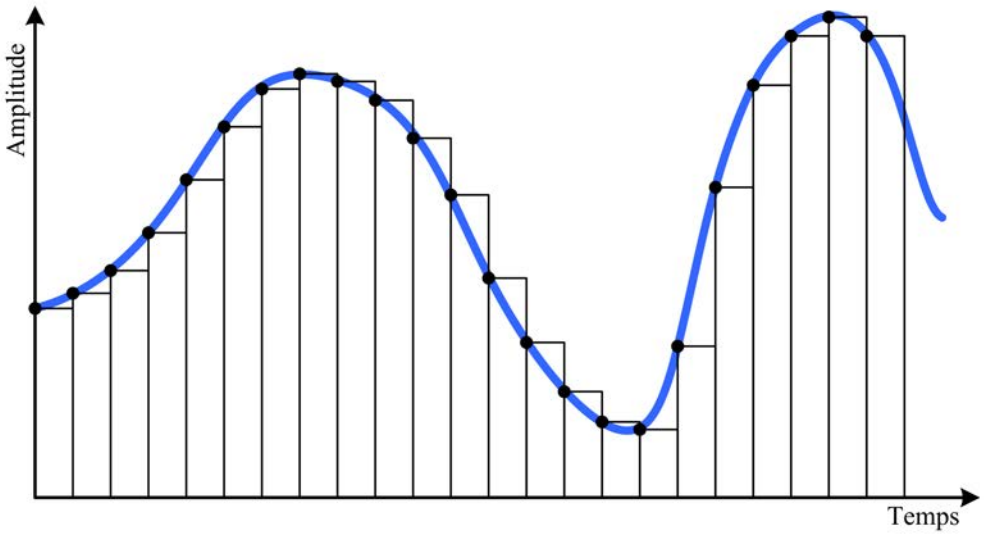


FIG. 1.2 – *Echantillonage d'un signal analgique*

### 1.2.3 Les images

Lorsqu'on numérise une image, on donne une représentation numérique de l'objet réel servant de support (papier, film, dia, négatif, ...). La plupart du temps, cette représentation sera matricielle. L'image sera représentée sous forme de matrice où chaque élément représente un point (pixel) de l'image. Chaque élément exprime la couleur du pixel associé.

La qualité d'une image matricielle est caractérisée par deux paramètres :

– La résolution : La résolution d'un appareil servant à numériser des images exprime le nombre de pixels que cette appareil relève par unité de longeur. La résolution est exprimée en dpi (dots per inch) ou ppp (points par pouce)[1]. Plus

[1] Un pouce vaut 2,54cm

le nombre de pixels est élevé par unité de longueur de l'image à numériser, plus la quantité d'information qui décrit l'image est importante et plus la résolution est élevée. Pour une image de 5 cm sur 5 cm (2 pouces sur 2 pouces) numérisée en 300 dpi, on obtiendra une matrice : 2*300 points sur 2*300 points donc 360.000 points.

– La dynamique ou profondeur de couleur : La dynamique est définie par le nombre de bits utilisés pour représenter chaque pixel. Plus la profondeur est élevée, plus grand sera le nombre de teintes (niveaux de gris ou couleur) représentées. Les images numériques peuvent être produites en noir et blanc (deux couleurs), niveaux de gris ou couleur. Si une couleur est représentée par un seul bit, on aura du noir et blanc. Si une couleur est représentée sur un octet (8 bits), on aura 256 couleurs possibles. C'est le cas des images dites en "fausses couleurs" ou "à palette" (format GIF par exemple) et des images en "niveaux de gris". Enfin, on parle de "vraies couleurs" lorsqu'on utilise un octet pour stocker chacune des composante dans l'espace de représentation des couleurs (Rouge - Vert - Bleu, donc 24 bits), on aura 16 millions de couleurs possibles, mais chaque point sera codé sur 3 octets. Reprenons l'exemple de notre image carrée de 5 cm en 300 dpi : si on la code en 24 bits, la taille de notre image finale sera de 360.000*24 bits soit 8.640.000 bits ou encore 1.080.000 ocets soit plus ou moins 1Mo.

Les principaux inconvénients de cette représentation sont d'une part, lorsque l'on veut agrandir l'image, celle-ci se dégrade, il en va souvent de même lors d'une réduction, d'autre part la taille des fichiers augmente très rapidement avec les dimensions et la dynamique de l'image, c'est pourquoi ce type d'image est très souvent compressé.

La compression se fait avec ou sans perte de qualité. Avec la compression sans perte de qualité, qui est réversible, on va pouvoir par exemple regrouper les pixels de même couleur. A la place de stocker 10 fois la même information correspondant à 10 points de couleur identique côte à côte, on ne stocke que la position du point de départ, du point d'arrivée et la couleur. Bien évidemment, plus l'image est simple, plus le taux de compression obtenu est important.

Pour la compression avec perte de qualité, qui est irréversible, on peut regrouper les pixels qui ont des couleurs proches et leur assigner la même couleur. Le taux de compression est choisi à l'enregistrement. En choisissant une qualité élevée, les différences avec l'original ne sont quasiment pas visibles sur un écran mais le taux de compression n'est pas très élevé.

Cependant la représentation matricielle n'est pas la seule utilisée pour la numérisation des images. On peut utiliser une représentation vectorielle. La description vectorielle consiste en une description géométrique de l'image. Le document numé-

risé contient un ensemble de formules mathématiques décrivant toutes les formes élémentaire constituant l'image (carrés, rectangles, ellipses, cercles, courbes, ...). Ainsi une ligne sera définie par seulement deux points et leurs coordonnées. Chaque forme élémentaire constituant un objet se voit assigner un certain nombre d'attributs tels que la couleur, la transparence, l'épaisseur du trait, le type de trait, ...C'est ce type de fichier qui est utilsé avec les logiciel de DAO[2]. La figure 1.3 illustre la différence entre les deux représentations.

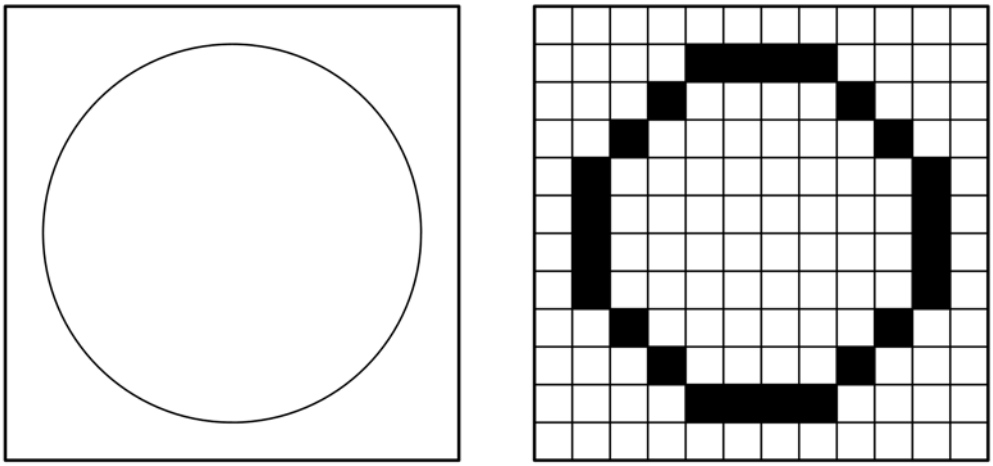


FIG. 1.3 – *Représentation vectorielle Vs représentation matricielle*

Une image vectorielle étant constituée d'entités mathématiques, toutes les modifications spatiales de l'image (réduction, agrandissement, translation, rotation, ...) n'entraînent pas de pertes d'information. Il suffit de changer les coordonnées des points de contrôle qui définissent l'objet. De plus, les images vectorielles permettent de définir une image avec très peu d'information, ce qui rend les fichiers très peu volumineux.

Mais il y a bien entendu un revers à la médaille pour cette technique. La représentation vectorielle ne permet que de représenter les formes simples. Or s'il est vrai que dans certains cas, en ne combinant que des formes élémentaires, on peut obtenir des images impressionnantes, il reste de nombreux cas où cette technique n'est pas assez puissante, notamment pour les photos réalistes.

[2] Dessin Assisté par Ordinateur

### 1.2.4 La vidéo

Les techniques de la numérisation vidéo seront détaillées au cours de la seconde partie de ce document, au travers des standards *MPEG-1*, *MPEG-2* et *MPEG-4*.

## 1.3 Références

Afin de réaliser ce chapitre nous nous sommes inspirés des documents suivants : [Cor98], [Hoo05], [UL00].

CHAPITRE 2

# Description de contenu

## 2.1 Evolutions techniques

L'indexation est une opération permettant de décrire un document dans une forme simple et manipulable afin de rendre ce document exploitable pour un usage précis tel que la recherche. Les premiers systèmes de recherche utilisaient des index textuels. C'est-à-dire qu'un document est décrit manuellement par une séquence structurée ou non de mots clefs ou descripteurs. Les bibliothécaires font cela depuis fort longtemps en notant le titre, le nom de l'auteur, et les thèmes des livres qu'ils conservent et en repérant le rayonnage sur lequel ils sont conservés. Ensuite la recherche est exécutée sur l'information ainsi extraite. Lors d'une recherche, la requête est exprimée sous la forme d'une séquence de descripteurs provenant du vocabulaire utilisé lors de l'indexation.

Cette technique comporte plusieurs problèmes. Premièrement l'indexation manuelle des documents est très coûteuse et incomplète. Ensuite, la relation entre les mots et les concepts que ceux-ci expriment est très complexe à cause des phénomènes de synonymie (différents mots exprimant le même concept) ou d'homonymie (le même mot exprime des concepts différents). On peut aussi relever le fait que certains concepts sont difficilement exprimables par un ou plusieurs mots clés.

A l'origine, l'indexation portait sur le document entier, dont on formalisait globalement le contenu. Si tel était le cas, c'est que pendant longtemps les contenus

que l'on indexait étaient des livres ou des documents papier. De tels contenus ne pouvaient être manipulés qu'en entier.

Depuis longtemps, les archives audiovisuelles sont conservées sur des films ou des cassettes magnétiques. Dans un premier temps, le stockage s'est fait sur étagère, les informations relatives aux contenus étant conservées sur des fiches. On reprenait ainsi le modèle de la bibliothèque. Bon nombre d'entreprises et d'institutions ont accumulé plusieurs dizaines d'années d'archives dont le contenu peut être précieux à une condition : être capable de retrouver la bonne information au bon moment. Il parait évident, au vu des volumes de données à traiter et de la rapidité à laquelle les recherches doivent s'exécuter, que les besoins du secteur de l'audiovisuel dépassent de loin les fonctionnalités offertes par un tel système.

Le développement de l'informatique et la numérisation des documents ont permis d'améliorer le système de façon considérable. Les documents sont stockés sur des cassettes informatiques des très grande capacité (environ 400Go). Ces cassette sont manipulées par des robots d'archives capables de les manipuler rapidement. Les données d'indexation quant à elles sont stockées dans des base de données manipulées par les applications des divers utilisateurs.

Avec la numérisation, il est devenu possible de ne manipuler qu'une partie d'un document (un extrait de film par exemple) indépendamment de l'entièreté du document. Il a donc fallu adapter les méthodes d'indexation de façon à pouvoir indexer des parties de document. L'index devient dès lors un moyen d'accéder directement à une partie pour pouvoir l'exploiter.

La croissance exponentielle des données numériques (textuelles, en images, vidéo, ...) a entraîné la création de bases de données gigantesques dans lesquelles il est devenu impossible de rechercher de manière exhaustive et manuelle une information donnée. Devant ces difficultés de recherche et de réutilisation des informations, il est devenu nécessaire d'organiser les données de façon à pouvoir y accéder rapidement et directement.

Grâce à l'arrivée du numérique, la description de contenu s'est automatisée. L'indexation automatique se base sur des algorithmes qui associent des descripteurs numériques à des parties de documents ou à des caractéristiques du document : couleurs, formes, visages, textures, etc. Dans le cas de documents textuels, chaque mot d'un paragraphe peut indexer celui-ci. De tels outils, après avoir éliminé les mots vides de sens (conjonctions, pronoms, ...), établissent des tables où chaque terme fait référence aux parties du document qui le contiennent. Dans le cas de documents multimédia (sons, images, vidéos, ...), la tâche est plus ardue car le document n'est plus décomposable en éléments simples, facilement repérables comme le sont les mots dans une phrase. Il faut donc disposer d'outils permettant de segmenter le contenu. Si on prend le cas d'une image il faut pouvoir déterminer que la

zone de coordonnées une telle représente un visage et que celui-ci appartient à tel personne.

A l'heure actuelle les données d'archivage peuvent être soit gérées séparément du média dans une base de données, soit, si la vidéo est numérisée (le cas le plus fréquent), être intégrées dans le fichier vidéo comme une composante supplémentaire. Pour le transport et l'échange des informations de contenu, on utilise en général les fichiers XML.

## 2.2 Méthodes

Les données servant à l'indexation, que l'on appelle métadonnées, peuvent être regroupé en trois grandes catégories :

- Métadonnées techniques : elles fournissent des informations techniques sur le document tels que le format d'image, le type de support, format d'enregistrement, etc. En général ces données sont relatives au document entier.
- Métadonnées administratives : elles fournissent les informations liées aux différentes tâches d'exploitation du contenu tel que les droits d'auteur ou les droits de diffusion. Ce type de données est aussi généralement lié à l'entièreté du document.
- Métadonnées descriptives : comme leur nom l'indique, elles décrivent le contenu du document. Elles peuvent être liées au document entier ou à des parties de celui-ci.

La méthode d'indexation classique, surtout pour les métadonnées descriptives, comporte plusieurs étapes :

- La segmentation : durant cette étape on va séparer les différentes parties qui constituent le document. Dans le cas d'une vidéo par exemple, on va séparer les différents plans constituant la vidéo.
- La classification : cette étape consiste à regrouper les parties de même nature. Dans le cas d'un journal télévisé, on va séparer la présentation des sujets qui se fait en plateau des sujets eux-mêmes.
- L'indexation : c'est l'étape où on va ajouter les métadonnées. Encore à l'heure actuelle, la saisie de ces informations se fait beaucoup de façon manuelle, et donc constitue un frein pour une indexation riche et complète. Une description scène par scène de longs métrages avec la précision évoquée précédemment n'est guère réaliste dans la pratique lorsque l'on a des milliers d'heures d'archives à renseigner. Mais de gros efforts de recherche ont déjà été consentis

pour automatiser l'indexation qui continue de susciter encore beaucoup d'intérêt.

Parmi les différentes méthodes d'indexation automatique, on peut citer parmi les plus répandues :

- Speech-to-text : Cette technique permet de retranscrire les paroles sous forme textuelle. L'avantage de cette technique c'est qu'elle permet d'une part de récolter beaucoup d'informations sur le contenu, et d'autre part d'avoir une recherche par mots-clés efficace. Cette technique fonctionne plutôt bien pour l'anglais, mais pour le français il subsiste encore quelques difficultés. En effet le marché francophone étant plus réduit que le marché anglophone, le développement de cette technologie pour le français a pris du retard.
- La production d'imagettes (story board) : Cette technique permet de récupérer des images d'une vidéo soit à intervalles de temps réguliers, ou, plus intéressant, à chaque changement de plan. On peut ensuite appliquer à ces images des méthodes d'indexation d'analyse d'images. L'avantage de cette méthode est qu'elle permet de sélectionner une séquence de la vidéo en parcourant uniquement le story board, sans devoir visionner tout le film. Actuellement, pratiquement tous les tous les logiciels du domaine offrent cette possibilité.
- Analyse des images : Le logiciel va analyser une image afin de repérer des élément graphiques porteurs de sens tels que la reconnaissance de logos, d'objets, de textes, l'analyse de générique, .... Ces techniques sont assez efficaces mais la transparence peut encore poser quelques petits problèmes.
- Reconnaissance automatiques des intervenants : Dans un premier temps le logiciel va repérer les différents intervenants, ensuite il va essayer de reconnaître ceux-ci. Ce type de logiciel est très employé dans le domaine de la sécurité pour l'analyse de photos, et commence à se développer pour l'analyse de séquences animées. Ce genre de logiciel nécessite un long moment d'apprentissage pour chaque individu que le logiciel est susceptible de reconnaître.
- ...

## 2.3 Les principaux travaux

Cette partie est consacrée à un historique des principaux schémas d'encodage des métadonnées utilisés dans les secteurs de l'archivage pour la conservation, des bibliothèques pour la classification et la description, et de l'audiovisuel. Chacun de ces secteurs a développé ses propres standards et normes pour pouvoir effectuer différentes opérations sur leurs ressources. La communauté archivistique a développé les standards ISAD(G) et EAD pour l'administration et la localisation des enre-

gistrements archivistiques. La communauté des bibliothèques utilise la famille de standards MARC pour représenter et échanger des métadonnées bibliographiques et le modèle FRBR pour représenter l'organisation structurelle d'une oeuvre. Enfin le monde audiovisuel où l'apparition des réseaux et du numérique a révélé les questions d'accès et de description comme étant des problématiques majeures, a développé MPEG-7, MXF, TV Anytime, ....

### 2.3.1 Les archivistes et les bibliothèques

#### *MARC*

Le format MARC[1] a été développé par la Bibliothèque du Congrès des États-Unis aux débuts des années 60, c'est-à-dire lorsque les ordinateurs sont venus aider les documentalistes pour cataloguer les monographies et les publications. Le format originel LC MARC a évolué vers le format MARC21 qui s'est imposé rapidement comme standard utilisé dans les bibliothèques en s'élargissant à tous les types de documents que l'on peut retrouver dans une bibliothèque. Encore à l'heure actuelle, la plupart des applications de bibliothéconomie utilisent le format MARC21. Ensuite l'IFLA[2] a développé UNIMARC[3] en 1977 dont le but était de permettre l'échange de l'information bibliographique informatisée et servir d'interface entre les formats MARC nationaux.

#### *FRBR*

L'IFLA a aussi développé un modèle d'organisation logique des données pour les bibliothèques nommé FRBR[4]. Dans ce modèle une ressource dispose de quatre états. Tout d'abord un créateur conçoit une OEUVRE. L'OEUVRE est un concept abstrait car elle doit être réalisée au travers d'une EXPRESSION. Une OEUVRE peut être réalisée par différentes expressions : deux metteurs en scène peuvent réaliser la même pièce de théâtre. Ensuite une EXPRESSION peut être représentée dans une ou plusieurs manifestations : un scénario, un enregistrement vidéo, un disque audio, une interprétation. Quand une MANIFESTATION est produite en masse, chaque copie constitue un ITEM.

[1] MAchine Readable Cataloging
[2] International Federation of Library Associations and Institutions
[3] UNIversal MARC
[4] Functional Requirements for the Bibliographic Record

## *ISAD(G)*

La norme ISAD(G)[5] diffusée en 1994 par le Conseil International des Archives (ICA) structure la description archivistique. Elle propose une description à plusieurs niveaux et non redondante des informations. La norme ISAD comprend des règles générales pour la description archivistique qui peuvent être appliquées indépendamment de la forme ou du support matériel des documents.

La norme ISAD(G) 2ème version (2000) fournit une liste de vingt-six éléments de description répartis en sept zones qui structurent la description :

- Identification : pour identifier l'unité de description.
- Contexte : l'origine et la conservation de l'unité de description.
- Contenu : l'objet de l'unité de description et son classement.
- Conditions d'accès et d'utilisation : les possibilités d'accès.
- Sources complémentaires : les documents ayant un lien significatif avec l'unité de description.
- Notes : les informations particulières qui n'ont pu être données dans aucune des autres zones.
- Contrôle de la description : comment, quand et par qui la description a été effectuée.

## *EAD*

En 1993 l'Université de Berkeley a décidé de remplacer le standard MARC-AMC[6] jugé insuffisant pour les descriptions multi-niveaux. Elle a créé la DTD EAD[7] qui s'appuie sur la norme ISAD(G) mais aussi sur d'autre standards tels que ISAAR(CPF) ou MARC.

## *Dublin Core*

En 1995 à Dublin fut lancé le standard Dublin Core pour l'échange de métadonnées par Internet. C'est un ensemble d'éléments simples mais efficaces pour décrire une grande variété de ressources en réseau. Le Dublin Core comprend 15 éléments dont la sémantique a été établie par un consensus international de professionnels provenant de diverses disciplines telles que la bibliothéconomie, l'informatique, le balisage de textes, la communauté muséologique et d'autres domaines connexes. A

[5] International Standard For Archival Description - General
[6] MARC Archival and Manuscript Control
[7] Encoding Archival Description

l'heure actuelle le format Dublin Core est un format de métadonnées très répandu pour les systèmes fonctionnant en réseau.

A l'origine le Dublin Core permettait de décrire des documents textuels traditionnels, mais son usage pour la description d'autres types de documents dépend de la similitude des métadonnées de ces types de document par rapport aux métadonnées d'un document textuel traditionnel.

## *INDECS*

Avec l'arrivée du numérique et d'Internet qui ont entraîné une disparition du support physique, les distinctions entre les différents secteurs de marché que sont les secteurs du livre, de la musique, du film et de la photographie disparaissaient au profit d'un secteur de marché commun. Dès lors les lois de commerce en vigueur ne pouvaient plus être appliquées de la même façon. C'est pour cela qu'en 1998 un certain nombre d'organismes détenteurs de droits se sont rassemblés et ont fondé l'INDECS[8]. Ce projet a les même ambitions que le Dublin Core, c'est-à-dire établir un certain nombre de descripteurs permettant d'identifier une ressource, en y ajoutant de façon plus précise des métadonnées concernant les personnes physiques et morales ainsi les propriétés intellectuelles.

## *FRBR, INDECS et DUCLIN CORE*

Le Dublin Core fut créé pour échanger des métadonnées via Internet. Le fait qu'il utilise des éléments simples pour décrire une ressource constitue ses principales limites. En effet les descripteurs fournis ne sont pas suffisants pour décrire des ressources complexes notamment en ce qui concerne les descriptions hiérarchiques, l'expression des droits associés aux oeuvres, les descriptions fines des personnes (auteurs, contributeurs, détenteurs de droits, . . .). D'un autre coté les normes et standards utilisés pour la description bibliographique sont très complexes et permettent des descriptions très fines des oeuvres. Cette complexité entraîne cependant des gros inconvénients car il devient très difficile de mettre en place des systèmes d'échange d'informations. Des recherches ont donc été faites pour tenter de concilier la précision bibliothécaire à la simplicité d'Internet. Ce qui a entraîné un rapprochement FRBR, INDECS et DUBLIN CORE s'appuyant sur l'organisation des métadonnées FRBR en y ajoutant le modèle d'organisation des droits et des actions INDECS. Ce travail de rapprochement a été effectué par un travail collaboratif entre les communautés Dublin Core et INDECS.

[8]Interoperability of Data in E-Commerce Systems

Après avoir constaté que les règles de catalogage basés sur MARC et AACR2R[9] comportaient des faiblesse pour cataloguer la musique, l'Université d'Indiana aux Etats-Unis développe depuis 2001 Variation 2. En effet ces formats savent très bien décrire de manière complète un objet physique, mais elles ont du mal à représenter l'organisation structurelle d'une oeuvre, notamment dans le cas de la musique. Le projet Variation 2 a pour but de développer un format d'archivage, des outils de recherche et de navigation, pour les bibliothèques, spécifiques au domaine de la musique.

### 2.3.2 Le monde de l'audiovisuel

Le développement des applications d'indexation se fait en parallèle avec les travaux de normalisation. Dans ce domaine, les principaux travaux en cours sont MXF, *MPEG-7* et TV Anytime. Ces normes et standards ont cependant des domaines d'application tout à fait différents mais complémentaires. Ils couvrent l'ensemble de la chaîne de production et de diffusion des médias.

#### *MXF*

Le MXF[10] est un format de fichier servant à l'échange de contenu audiovisuel et des données et métadonnées associées tout au long de la chaîne de production audiovisuelle. Il a été conçu et implémenté dans le but d'améliorer l'interopérabilité des fichiers entre les serveurs, les stations de travail et autres terminaux de création de contenu. Sa conception est le résultat de la collaboration des principaux fabricants et organisations du domaine tels que Pro-MPEG, EBU[11], l'association AAF[12], afin de s'assurer que le format réponde bien à toutes les attentes. MXF offre une interopérabilité des contenus entre les différentes applications utilisées dans la chaîne de production télévisuelle.

MXF est indépendant de tout système de compression. Il simplifie l'intégration de systèmes utilisant *MPEG* et DV[13] mais aussi des futurs systèmes de compression qui seront développés ultérieurement. Ce qui signifie que le transport de ce type de fichier est indépendant du contenu et des fabricants d'équipement. Tous les processus qui doivent être effectués le sont en invoquant les codecs hardware ou software appropriés de façon transparente pour l'utilisateur.

[9] Anglo-American Cataloging Rule

[10] Material eXchange Format

[11] European Broadcasting Union ou Union européenne de radio-télévision (UER) est la plus importante association professionnelle de radiodiffuseurs nationaux dans le monde

[12] Advanced Authoring Format

[13] (Sony) Digital Video compression format

MXF est indépendant des systèmes d'exploitation, des plates-formes et des infrastructures réseaux. Il permet de se déplacer dans le fichier, de faire des transferts partiels de fichiers, d'accéder au fichier et de l'utiliser avant la fin de son transfert, c'est-à-dire qu'il permet le streaming.

### *MPEG-7*

Le Moving Picture Experts Group dans le cadre de la description de contenu a développé les standards *MPEG-7* et *MPEG-21*.

*MPEG-7* aussi connu sous le nom de "Multimédia Content Description Interface" est un standard *ISO/IEC* basé sur *XML* permettant de décrire le contenu d'un fichier multimédia avec un certain niveau d'interprétation des informations de ce fichier. Le *MPEG-7* n'est pas dédié à un média particulier, mais au contraire permet de standardiser un nouveau moyen de recherche multimédia et ce en visant le plus grand nombre d'applications possibles. On peut évidemment établir une description *MPEG-7* d'un fichier *MPEG-1* ou *MPEG-2*, mais on peut faire de même avec un film analogique *PAL/SECAM*[14] ou encore un journal papier. Il s'agit uniquement d'un standard de représentation du contenu des documents. Cette description sera liée au contenu multimédia lui-même afin de permettre à l'utilisateur, via un équipement spécifique ou un logiciel, une recherche pertinente et rapide d'un média.

*MPEG-7* fournit donc un grand nombre de fonctions descriptives standardisées d'un large éventail d'informations multimédia. Le choix d'un ou plusieurs descripteurs se fera donc en fonction du contexte de l'application visée. Cela implique qu'il est possible que le même matériel soit décrit de manière différentes. En effet *MPEG-7* supporte plusieurs niveaux d'abstraction différents, allant des caractéristiques de bas niveau comme la forme, la texture, la couleur, le mouvement de la caméra ou encore le timbre de la musique à des informations sémantiques de haut niveau comme le genre de contenu ou des événements. Le niveau d'abstraction influence grandement la manière dont l'extraction des informations est faite. En effet, plus le niveau d'abstraction est bas, plus il sera possible d'extraire automatiquement ces informations. Dans le cas contraire, une intervention humaine sera souvent nécessaire.

A coté de la description du contenu, *MPEG-7* permet aussi d'inclure d'autres types d'informations tels que :

- le format : le type de compression utilisé (*JPG*), taille, etc. Ces informations servent à déterminer si le matériau peut être lu par l'utilisateur ;
- les conditions d'accès au support : des informations concernant le copyright et le prix ;

[14]Norme de codage européenne pour la télévision hertzienne au format 4/3

– la classification (appréciation parentale, catégorisation) ;
– des liens vers d'autres matériaux intéressants.

## *MPEG-21*

*MPEG-21* est un standard basé sur *XML* qui a pour vocation de fournir une structure pour une utilisation transparente et interopérable de contenus multimédia de natures très variées (*MPEG-1*, *MPEG-2*, *MPEG-4*, *MPEG-7*, *JPG*, documents Word, MP3, ...). Le *MPEG-21* est composé de Digital Item (DI) avec lesquels les utilisateurs interagissent.

*MPEG-21* est très complémentaire de TV Anytime dans le sens où il ne fournit que très peu de moyens pour décrire la ressource mais permet d'organiser ces ressources hiérarchiquement les unes par rapport aux autres dans une même structure. Le *MPEG-21* fournit des outils indispensables à une chaîne de diffusion de contenus multimédia : Le Right Expression Language (REL) décrit les droits et les conditions d'accès et le Digital Item adaptation (DIA) décrit comment les contenus devront s'adapter aux contraintes réseaux et aux terminaux clients.

Les descriptions de chacune des ressources étant faites, elles sont ensuite encapsulées dans une structure *MPEG-21* grâce à un module graphique. Cette capacité à former des documents complexes offre de grandes possibilités en proposant, par exemple, des options interactives à l'utilisateur : lors de la lecture d'une vidéo, l'utilisateur pourrait se voir suggérer des liens incrustés dans la vidéo, ces liens pouvant pointer vers d'autres ressources telles que d'autres vidéos, des documents textes ou même des programmes.

## *TV Anytime*

Les spécifications de la norme TV Anytime ont été créées par The TV Anytime forum. Il s'agit d'une association d'organisations cherchant à développer des spécifications permettant d'intégrer des services audiovisuels sur les appareils des consommateurs, simplement comme s'il s'agissait d'un stockage local. Les objectifs fondamentaux du forum TV Anytime sont :

– la définition des spécifications qui permettront aux applications d'exploiter le contenu local des appareils des consommateurs ;
– l'indépendance des réseaux ;
– le développement des spécifications afin de fournir un système interopérable et intégré du fournisseur/créateur de contenus jusqu'à l'utilisateur final ;

– la spécification des structures de sécurité nécessaires afin de protéger les intérêts de chacun.

Le format TV Anytime s'appuie sur le format *XML* et reprend certaines parties de *MPEG-7*. Chaque fichier TV Anytime est divisé en plusieurs catégories dont les principales sont :

– ProgramInformationTable qui regroupe les programmes (émissions, films, ...) contenant toutes les informations nécessaires telles que le titre, synopsis, genre, langue, contrôle parental, etc. Chacun des ces programmes est référencé par un crid[15] unique pour un programme donné.

– ProgramLocationTable qui fournit les informations nécessaires à la localisation physique et temporelle de la vidéo. Le crid permet de retrouver le programme correspondant à une localisation.

– SegmentInformationTable qui permet de segmenter plus finement un programme référencé par son crid. (Par exemple un programme correspondant à un bandeau publicitaire segmenté en autant de publicités différentes)

Le standard TVA-1[16] décrit comment un utilisateur d'un équipement d'enregistrement numérique personnel de type PVR (Personal Video Recorder) ou DVR (Digital Video Recorder) peut rechercher, sélectionner et acquérir des contenus multimédia. Il détaille en particulier les moyens qui permettent d'incorporer dans les systèmes de radiodiffusion numérique les spécifications relatives aux métadonnées (données de description et d'exploration du contenu) et au référencement du contenu (données grâce auxquelles il est possible de localiser dans le temps tel ou tel programme).

Une deuxième vague de spécifications TVA-2[17] a pour but de normaliser des procédures de plus en plus évoluées dans le domaine du stockage multimédia personnalisé : partage de fichiers et "superdistribution", synchronisation de sources de contenus multiples, mise en forme de contenus multiples, stockage de contenus interactifs, ciblage du contenu en fonction du profil utilisateur, etc.

## 2.4 Références

Afin de réaliser ce chapitre nous nous sommes inspirés des documents suivants : [Bac01], [Bac00], [Ber04], [Vin00].

[15] Le crid : Content Reference Identifier est une URI particulière de la forme crid ://*/*
[16] TV Anytime Phase 1
[17] TV Anytime Phase 2

# Deuxième partie

# Numérisation

CHAPITRE 3

# MPEG-1

## 3.1 Historique

L'apparition du CD dans les années 80 favorisa l'avènement du multimédia. Ce support permettait de stocker 650 Mo de données, ce qui représentait une capacité équivalente voir supérieure à celles des disques durs de l'époque. Avec un support de cette importance, on essaya rapidement d'y stocker des données informatiques telles que du son, des images, de la vidéo, etc. Ce qu'on appelle aujourd'hui le multimédia venait de naître.

Cependant la vidéo restant une application gourmande en débit, il fallait pouvoir la compresser avant de la stocker. Disposant déjà de la compression *JPEG*[1], et en considérant une vidéo comme une suite d'images fixes, chacune de ces images pouvait donc être compressée en *JPEG*. On vit alors apparaître toute une série de codecs[2] *MJPEG*[3]. Mais comme il n'y avait pas de norme de compression, les codecs étaient souvent incompatibles entre eux, et il devint vite impossible d'échanger des vidéos sans les codecs associés à celles-ci.

[1]JPEG : Joint Photographic Experts Group
[2]CODEC : abréviation de COdeur-DEcodeur
[3]MJPEG : Motion JPEG

En 1988 on créa donc le comité *MPEG*[4] ayant pour but de créer une norme de compression permettent de stocker et de reproduire de la vidéo. La qualité de la vidéo ainsi codée devait avoir la qualité magnétoscope et un débit maximal de 1.5 Mbits/s. Quatre ans plus tard la norme *MPEG-1* était finalisée.

Cependant il fallut encore attendre quelques années pour que *MPEG-1* connaisse son véritable succès. La compression nécessitait une quantité de mémoire (16 Mo de RAM) que la plupart des ordinateurs de l'époque ne possédaient pas (en moyenne 4 Mo RAM). La compression *MPEG-1* imposait aussi de lourds investissements : 3000 à 4000 € pour une carte de compression. Alors qu'aujourd'hui, n'importe quel PC de bas de gamme est capable de compresser et de lire des séquences vidéo en *MPEG-1* avec des logiciels souvent gratuits.

*MPEG-1* fut aussi à l'origine d'une véritable révolution à la fin des années 90, plus précisément la partie audio de la norme. Cette partie est décomposée en *MPEG-1 Audio Layer I, II et III*, cette dernière étant plus connue sous le nom de *MP3*. Celui-ci a fait trembler toute l'industrie de la musique fin des années 90 et continue d'en exaspérer plus d'un.

## 3.2 Techniques de codage de l'image

Lorsqu'une image est affichée sur un écran vidéo, chaque pixel composant cette image est en fait la juxtaposition de trois points lumineux projetés sur l'écran : un rouge, un vert et un bleu. On dit alors que l'image est codée en *RGB*[5]. A chaque image correspondent en fait trois images en teintes de gris, exprimant l'intensité des couleurs rouge, verte et bleue. Plus on s'approche du blanc, plus l'intensité est forte, et vice versa. La figure 3.1 illustre cette technique.

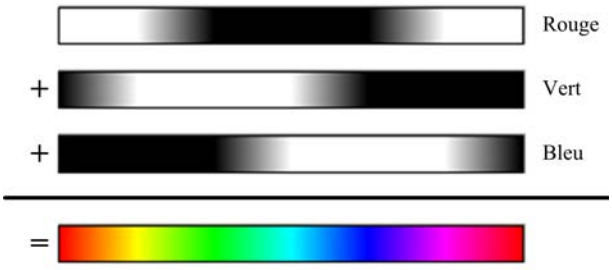


FIG. 3.1 – *Format RGB*

[4]Moving Picture Experts Group
[5] Red ,Green, Blue

Cependant, l'oeil humain étant plus sensible à la luminosité qu'aux couleurs, *MPEG* utilise un autre codage : le *YUV*. Celui-ci favorise la luminance par rapport à la chrominance. Pour chaque image, on dispose d'une matrice pour coder la luminance (Y), d'une autre pour coder la chrominance bleue (U), et d'une dernière pour coder la chrominance rouge (V).

En *RGB* pour une image à afficher on stocke trois images de même taille (une pour chaque couleur). En *YUV* seule la matrice de luminance a la même taille que l'image (une information pour chaque pixel). Les matrices de chrominance sont quant à elles inférieures à la taille de l'image. En effet, la chrominance est moins importante pour l'oeil humain, on ne stocke donc pas l'information pour chaque pixel, ce qui permet de réduire la taille des matrices de chrominance.

Il existe différents types de codage YUV notés sous forme : *"Y : U : V"*

**4 : 2 : 2** Sur une série de 4 pixels, les quatre ont l'information de luminance, mais seulement deux disposent de l'information de chrominance bleue et rouge (d'où la notation 4 : 2 : 2). Les matrices U et V ont une taille équivalente à la moitié de la matrice Y. Ce format est utilisé dans la vidéo professionnelle (norme CCIR 601).

**4 : 1 : 1** Sur une série de 4 pixels, les quatre ont l'information de luminance, mais un seul dispose de l'information de chrominance bleue et rouge. Les matrices U et V ont une taille équivalente au quart de la matrice Y.

**4 : 2 : 0** Dans ce cas-ci, pour chaque ligne de l'image, on ne stocke la chrominance bleue qu'une ligne sur deux, et la chrominance rouge sur les autres. Et sur une série de quatre pixels, les quatre ont l'information de luminance, mais seulement deux disposent de l'information de chrominance bleue ou rouge (suivant la ligne sur laquelle on se trouve). Les matrices U et V ont une taille équivalente à la moitié de la matrice Y. C'est le format utilisé par *MPEG-1*.

Ces différents formats sont repris à la figure 3.2 (Les matrices YUV correspondent à une image de 400*400 pixels).

## 3.3 Méthode de compression vidéo

Lorsque l'on regarde une séquence vidéo, on se rend rapidement compte qu'il y a des éléments redondants. Cette redondance est de deux types :

**spatiale** On parle de redondance spatiale lorsqu'au sein d'une image, on retrouve des fragments d'images où les pixels sont similaires.

**temporelle** On parle de redondance temporelle lorsque des fragments d'images se répètent sur plusieurs images de suite, même si ceux-ci se sont déplacés.

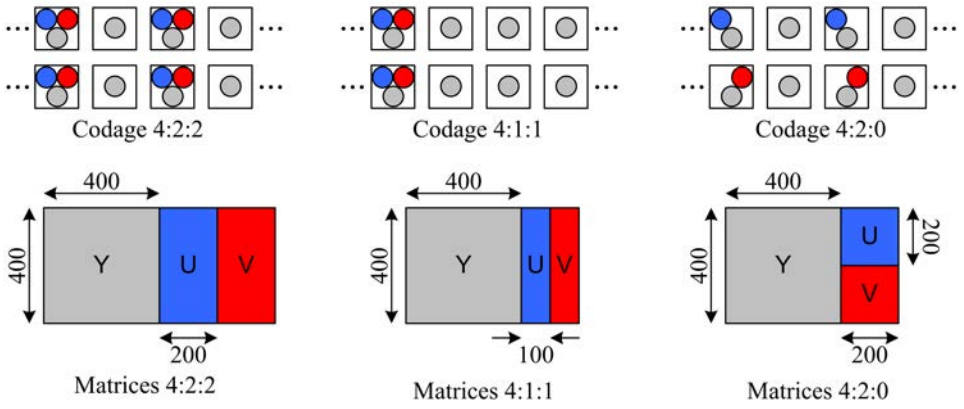


FIG. 3.2 – *Codage YUV*

En compressant chaque image individuellement, comme on le faisait avec les codecs *MJPEG*, on élimine la redondance spatiale, mais on conserve la redondance temporelle laissant un taux de compression qui pourrait être amélioré. C'est ici qu'intervient la norme *MPEG-1*.

En *MPEG-1* une séquence vidéo est divisée en divers éléments ayant une structure hiérarchique décrite à la figure 3.3

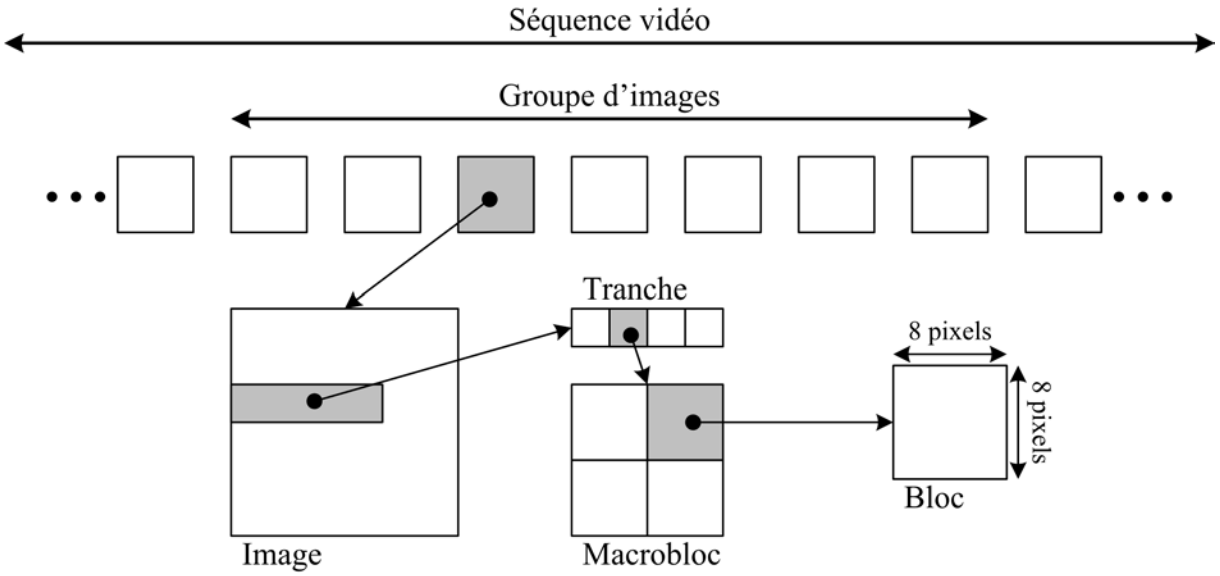


FIG. 3.3 – *Structure hiérarchique d'une séquence vidéo MPEG-1*

On peut définir les différents termes utilisés dans la figure 3.3 comme suit :

**Bloc** Ensemble de valeurs de luminance et de chrominance pour 8*8 pixels. Unité de traitement de la DCT[6].

**Macrobloc** Matrice carrée de dimension deux de blocs. Unité de traitement de la compensation de mouvement[7].

**Tranche** Ensemble de un ou plusieurs macroblocs. Elle est utilisée dans la gestion d'erreurs lorsque le flux de données comporte des erreurs. Si une erreur se produit dans une tranche, le décodeur peut sauter cette tranche, évitant de devoir sauter l'image entière. Plus les tranches sont courtes (contiennent moins de macroblocs), meilleure est la gestion des erreurs, cependant le taux de compression sera moins élevé.

**Image** Elles sont de trois types différents :
- I : Image clé ou Intra Picture
- P : Image prédite ou Predicted Picture
- B : Image bi-directionnelle ou Bidirectional Pictures

**Groupe d'images** Suite d'images débutant par une image I et se terminant avant la prochaine image I.
Exemple : I B B P B B P B B

### 3.3.1 Les images clés : I

Les images clés sont compressées indépendamment des images précédentes et suivantes contrairement aux images P et B, qui, elles, dépendent des images voisines. Les images I sont compressées en *JPEG*. Dans une séquence vidéo, on retrouve une image I toutes les 10 à 15 images. Avec un débit de 25 à 30 images par seconde, il y a 2 à 3 images I par seconde. C'est lors de l'encodage que l'utilisateur choisit la fréquence des images I.

C'est ce type d'image qui permet l'accès aléatoire dans une séquence vidéo. Lorsqu'on veut pouvoir accéder à un endroit quelconque de la vidéo, il faut trouver l'image I la plus proche. D'où l'importance de ce type d'image. Donc pour les applications où l'accès aléatoire est important, on retrouve généralement une image I toutes les ½ secondes.

Les images I sont codées suivant la méthode illustrée à la figure 3.4.

[6]Discrete Cosine Transform, en français : Transformée en Cosinus Discrète. Explicitée ultérieurement

[7]Explicitée ultérieurement

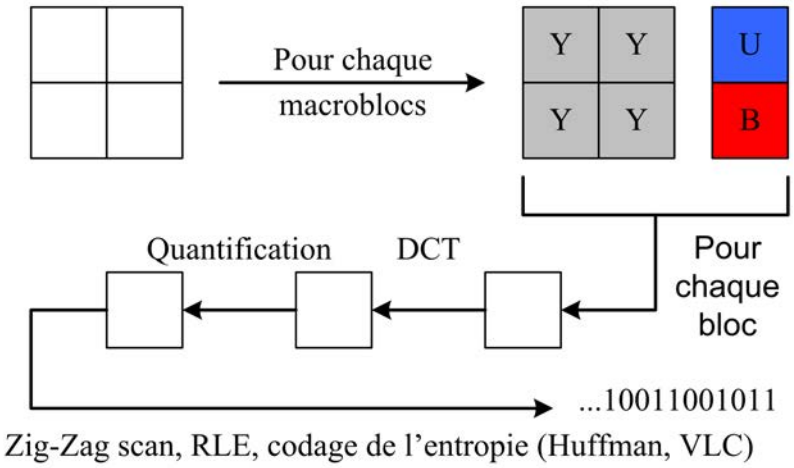


FIG. 3.4 – *Compression d'une image I codée en YUV 4 : 2 : 0*

Les macroblocs sont composés de quatre blocs de 8*8 pixels. Avec l'encodage YUV 4 : 2 : 0, on dispose de quatre blocs de 8*8 pixels pour la luminance, et d'un bloc de 8*8 pixels pour chacune des deux chrominances. Pour le YUV 4 : 2 : 2 on aura quatre blocs de luminance, deux de chrominance bleue et deux de chrominance rouge.

Sur chacun de ces macroblocs va être appliquée la Transformée en Cosinus Discrète (DCT). Elle transforme la représentation spatiale des blocs en une représentation sous forme mathématique différente, plus compacte, requiérant de traiter moins d'information. Cette nouvelle représentation ne se base plus sur une analyse spatiale (positions horizontale, verticale et amplitude) mais sur une analyse fréquentielle savamment calculée. Cette technique est rendue possible grâce à l'utilisation d'une variante des séries de Fourier. Celles-ci permettent de reconstruire une fonction à partir d'une somme de sinusoïdes multipliées chacune par un certain coefficient dit "de Fourier". La DCT s'apparente à cette méthode. La DCT, en elle-même, ne compresse pas l'image et n'entraîne pas non plus la perte de données. Elle représente simplement l'image sous une forme qui se prête beaucoup mieux à la compression. Elle est néanmoins l'étape la plus complexe et la plus importante car elle permet de séparer les hautes fréquences des basses fréquences. Il ne reste alors plus qu'à appliquer un codage intelligent des différents coefficients.

Ensuite on passe à l'étape de quantification. C'est durant cette étape que l'image est compressée, mais aussi qu'il y a une perte de qualité. La quantification consiste à diviser chaque matrice associée aux blocs résultants de la DCT par la matrice de quantification. Cette matrice est une matrice de 8*8 coefficients savamment choisis. Le but de cette méthode est d'atténuer les fréquences les plus élevées (celles auxquelles l'oeil humain est le moins sensible). Plus la quantification est forte, plus les hautes fréquences seront atténuées (ramenées à des valeurs nulles).

Le résultat de la quantification appliquée à un bloc est une matrice 8*8 où les hautes fréquences sont ramenées à des coefficients nuls. Les coefficients non nuls, ceux représentant les basses fréquences (celles auxquelles l'oeil humain est sensible), sont quant à eux situés dans la partie supérieure gauche. (Voir figure 3.5). Ceci offre l'avantage que la longue suite de zéros consécutifs nécessitera peu de place lors de l'encodage du fichier. Cependant si la quantification renvoie trop de zéros (taux de compression trop élevé), il n'y a pas assez d'information significative pour représenter le bloc. Dans ce cas la division en blocs devient visible à l'écran et l'image apparaît pixellisée.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 79 | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| -2 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

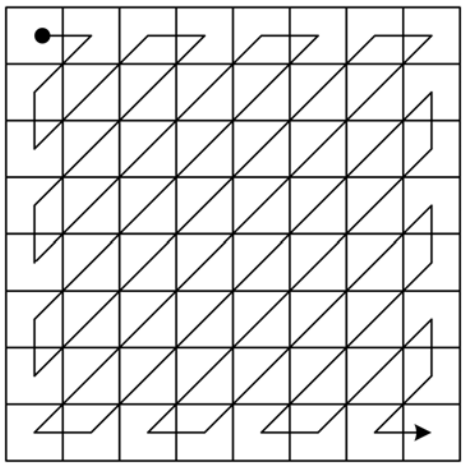

FIG. 3.5 – *Résultat de la quantification et méthode zig zag*

Reste maintenant à transformer les blocs en une suite de bits pour pouvoir encoder le fichier. Dans un premier temps on va transformer la matrice résultant de la quantification en une suite de ses valeurs par la méthode du "zig zag" appelée aussi méthode de diagonalisation (voir figure 3.5).

Résultat : 79, 0, -2, -1, -1, -1, 0, 0, -1, 0, 0, ...,0 EOB[8]

La dernière étape est le codage de l'entropie qui permet de compresser une dernière fois les données sans perte de qualité. Premièrement le codage RLC (Run Length Coding) va être appliqué à la suite des valeurs. Il permet de gérer plus efficacement les sous-suites de zéros consécutifs. Le codage RLC émet simplement le nombre de zéros plutôt que toute la suite de bits nuls.

En étudiant la probabilité de répétition de certaines valeurs particulières de coefficients, certaines valeurs se rencontrent moins fréquemment que d'autre. Sachant

[8]EOB : End Of Block

cela, la suite résultant du codage RLC va encore subir une compression à longueur variable (VLC ou codage dit de “Huffman”). Les valeurs les plus fréquentes sont codées avec des codes courts et les autres en codes plus longs. Cette technique est celle utilisée pour le code morse. La lettre “e” qui est la lettre la plus fréquente n’est codée que par un seul point.

### 3.3.2 Les images prédites : P

Une image P, à la différence d’une image I, est codée à partir de l’image qui la précède grâce à la technique de la compensation de mouvement. Cette technique travaille sur les macroblocs de l’image (matrices carrées de 2*2 blocs), et on ne code que la différence entre les macroblocs, s’il y a une différence.

En prenant comme exemple la figure 3.6, on se rend compte que d’une image à l’autre seuls les macroblocs contenant le disque jaune sont modifiés. Donc ces blocs devront être décalés d’une certaine valeur, tandis que les macroblocs de l’arrière plan restent inchangés.

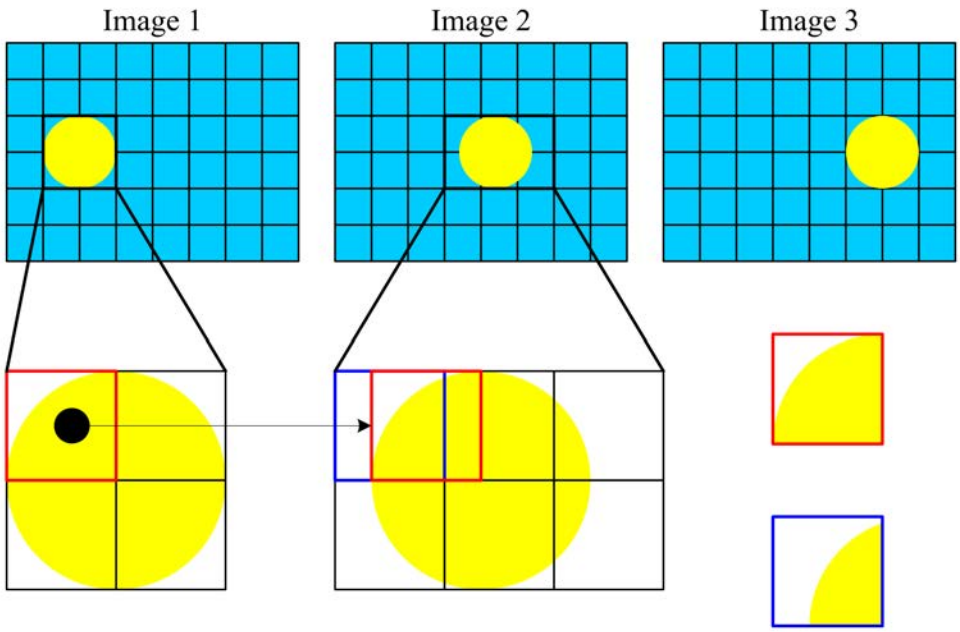


FIG. 3.6 – *Exemple où les images P sont utiles*

Pour chaque macrobloc compressé avec la méthode de compensation de mouvement (ceux qui se sont déplacés), on retient dans le fichier compressé :

– un vecteur de déplacement rendant compte du déplacement de ce macrobloc dans chaque direction par rapport à son emplacement dans l'image précédente ;
– la différence entre le contenu du macrobloc de référence (en rouge à la figure 3.6) et le macrobloc qui va être codé (en bleu à la figure 3.6). Cette différence, appelée le terme d'erreur, est codée de la même manière que les images I.

Pour pouvoir utiliser la méthode de compensation du mouvement, l'encodeur doit aussi disposer d'un décodeur, ceci pour la raison suivante. Quand l'encodeur veut encoder un macrobloc de $l'image_n$, il calcule le vecteur de déplacement du macrobloc à partir de $l'image_{n-1}$. Ensuite il calcule le macrobloc prédit à l'aide de son décodeur grâce à $l'image_{n-1}$ et au vecteur de déplacement. Ce macrobloc prédit est ensuite soustrait au macrobloc de $l'image_n$ pour calculer le terme d'erreur.

Reste à expliquer la manière dont on calcule les vecteurs de déplacement. Dans ce domaine la norme *MPEG-1* ne spécifie en rien la méthode de calcul. Chaque implémentation est libre d'utiliser sa propre méthode. Cependant il existe trois grandes catégories d'algorithmes : la similitude de blocs (block matching), la similitude de gradient (gradient matching) et la corrélation de phase (phase correlation). Mais peu importe le choix, ce sont des techniques gourmandes en temps de calcul car complexes.

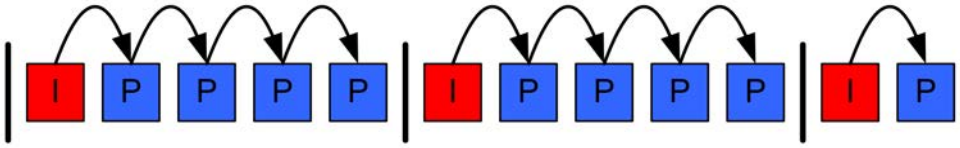


FIG. 3.7 – *Un signal basique MPEG-1*

Pour finir on voit à la figure 3.7 comment sont utilisées les images P dans un signal *MPEG-1*. Une image I est encodée à intervalles de temps réguliers, et entre ces images I, on intercale des images P calculées sur base de l'image précédente. On peut remarquer aisément les deux problèmes majeurs que peuvent poser ces deux techniques. Si une erreur s'immisce dans une image P, toutes les images P suivantes comporteront cette erreur. Deuxième problème, si une image I se perd, il sera impossible de décoder les images P suivantes jusqu'à la prochaine image I.

### 3.3.3 Les images bi-directionnelles : B

La technique des images P est très puissante, mais elle n'apporte pas de solution pour toutes les situations que l'on peut rencontrer dans les films. Prenons l'exemple d'une porte qui s'ouvre, il est impossible de déduire ce qui se trouve derrière la porte à partir de l'image précédente puisque la porte était fermée.

C'est pour palier à ce genre de situation qu'on utilise les images B. Les images B sont un peu comme les images P, si ce n'est qu'elles peuvent se référer à la première image I ou P, précédente ou suivante, mais en aucun cas à une autre image B. Donc normalement ces images ne peuvent transmettre leurs propres erreurs. Les images P, elles non plus, ne se réfèrent pas aux images B. La figure 3.8 montre une suite d'images *MPEG-1*.

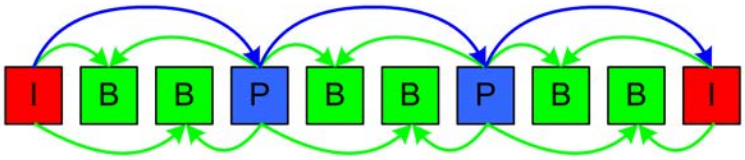


FIG. 3.8 – *Exemple de suite d'images MPEG-1*

Ce type de codage est très puissant. Une séquence d'images IBBP consomme environs 60% de bits en moins qu'une séquence IIII, pour une qualité équivalente. Cependant ce codage nécessite beaucoup plus de temps de traitement, que ce soit à l'encodage ou au décodage.

Si la fréquence des images I n'est pas imposée à l'utilisateur lors de l'encodage, celle des images P et B ne l'est pas non plus. Cela est laissé au choix de l'utilisateur. La fréquence des images I est importante pour l'accès aléatoire dans une vidéo. Et la fréquence des images P et B dépend de la puissance du décodeur et de l'encodeur.

Lorsqu'on veut encoder un flux vidéo en *MPEG-1*, l'ordre des images doit être modifié à cause des images B. En effet, une image B peut se référer à une image qui suit, dans ce cas le décodeur serait dans l'incapacité de décoder l'image. La figure 3.9 illustre bien ce problème. Elle montre aussi que ce sont les images I qui nécessitent le plus d'espace, ensuite ce sont les images P et ce sont les images B qui demandent le moins d'espace. On peut aussi noter que les dernières images B d'un groupe d'images arrivent après la première image I du groupe d'images suivant.

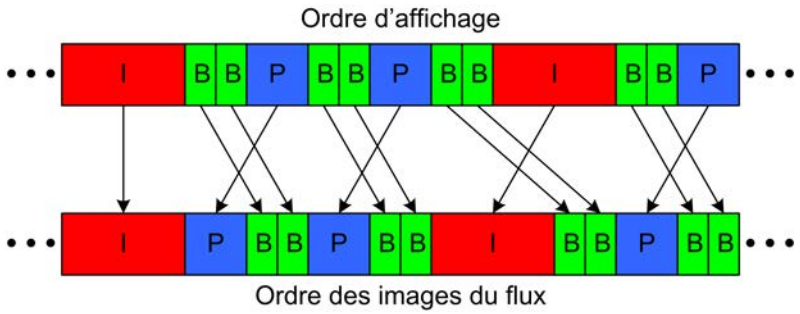


FIG. 3.9 – *Composition d'un flux MPEG-1*

## 3.4 Synchronisation son/vidéo

Nous allons maintenant aborder le problème de la synchronisation du son et de la vidéo. Cependant nous n'aborderons pas les principes de compression du son. Ce sujet, vaste et complexe, fait l'objet de nombre d'articles sur Internet ou dans la littérature.

Une séquence vidéo se composant d'un flux vidéo et d'un flux audio, la norme de compression *MPEG* est basée sur trois parties :

- une partie vidéo : servant à (dé)coder le flux vidéo ;
- une partie audio : servant à (dé)coder le flux audio ;
- une partie système : servant au (dé)multiplexage audio/vidéo.

Les figures 3.10 et 3.11 montrent comment ces trois parties interagissent.

En *MPEG-1*, la synchronisation entre l'audio et la vidéo est assurée par un mécanisme incluant deux paramètres :

- l'horloge de référence du système : SCR[9] ;
- un marquage temporel : PTS[10] et/ou DTS[11].

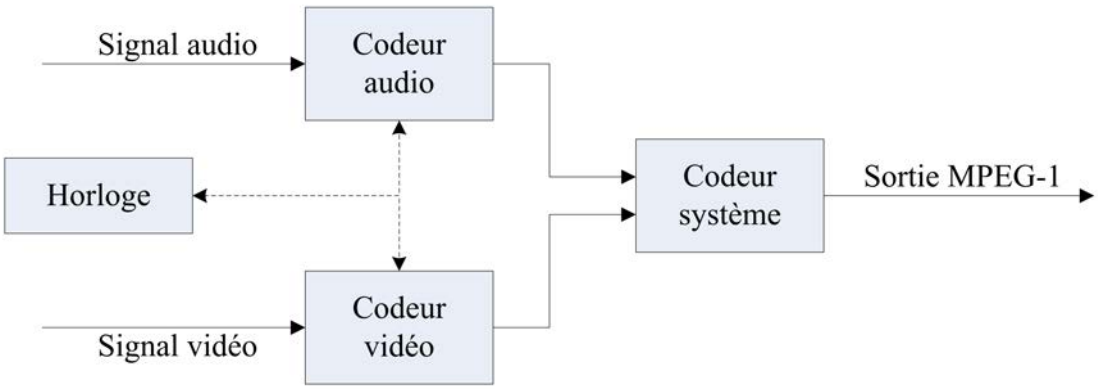


FIG. 3.10 – *Multiplexage d'une séquence vidéo*

[9]SCR : System Clock Reference, en français : référence temporelle système
[10]PTS : Presentation Time Stamp, en français : marqueur temporel de présentation
[11]DTS : Decode Time Stamp, en français : marqueur temporel de décodage

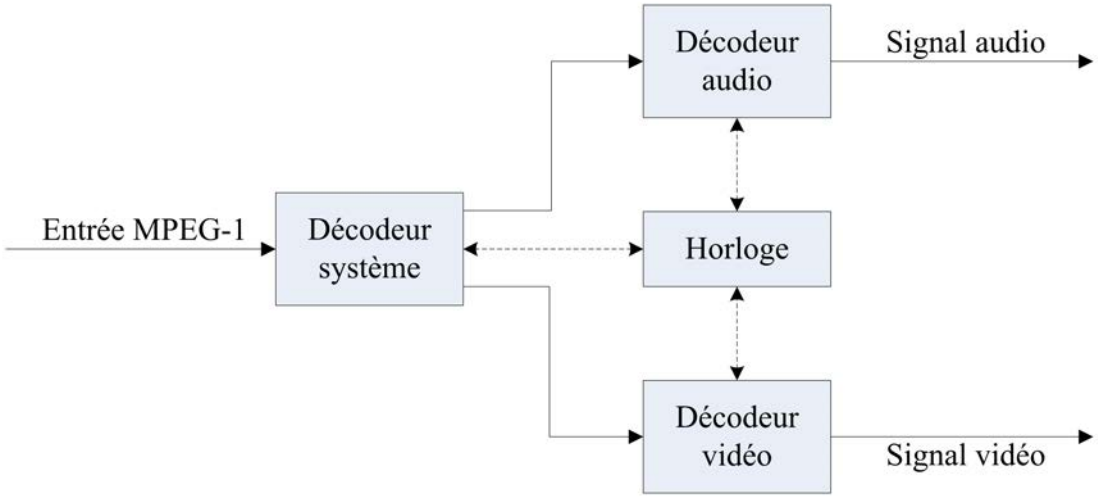


FIG. 3.11 – *Démultiplexage d'une séquence vidéo*

L'horloge du système d'encodage est à la fois stable et précise, ce qui n'est pas le cas de celle du système de décodage. Si celle-ci venait à se dérégler, la vidéo et le son ne seraient plus synchronisés lors de la lecture de la séquence vidéo. Pour remédier à ce problème, lors de l'encodage, on place dans la couche système du flux, à des intervalles de temps réguliers n'excédant pas 700 ms, une référence temporelle (SCR) provenant de l'horloge de l'encodeur. Lorsque l'horloge du décodeur se dérègle, elle peut être réajustée par la partie système du décodeur grâce au SCR.

Comme cela a déjà été mentionné préalablement, à cause des images B, les images ne sont pas codées dans le même ordre que celui dans lequel elles seront affichées. Une suite d'images IBBP sera encodée sous la forme IPBB, les images B ayant besoin de l'image P à laquelle elles se rapportent. Donc, afin que le décodeur puisse savoir quand décoder et quand afficher une image, on utilise le marqueur DTS, indiquant quand une image doit être décodée, et le marqueur PTS, indiquant quand l'image doit être affichée.

Pour que la suite d'images IPBB soit affichée dans le bon ordre, l'image I va être décodée en premier. Ensuite, l'image P sera décodée pendant que l'image I est affichée. Une fois l'image P décodée, elle n'est pas affichée directement car ce sont les deux images B qui doivent être affichées. Celles-ci sont décodées et affichées simultanément. Enfin l'image P est affichée. La figure 3.12 illustre l'utilisation des marqueurs DTS et PTS pour une suite d'images IBBP.

Elle montre aussi qu'aux images B n'est associé que le marqueur PTS, en effet elles sont décodées et affichées simultanément, tandis que les images I et P disposent des deux marqueurs. Lors de la lecture, le décodeur a la possibilité de sauter ou de répéter des images afin de respecter le marquage temporel.

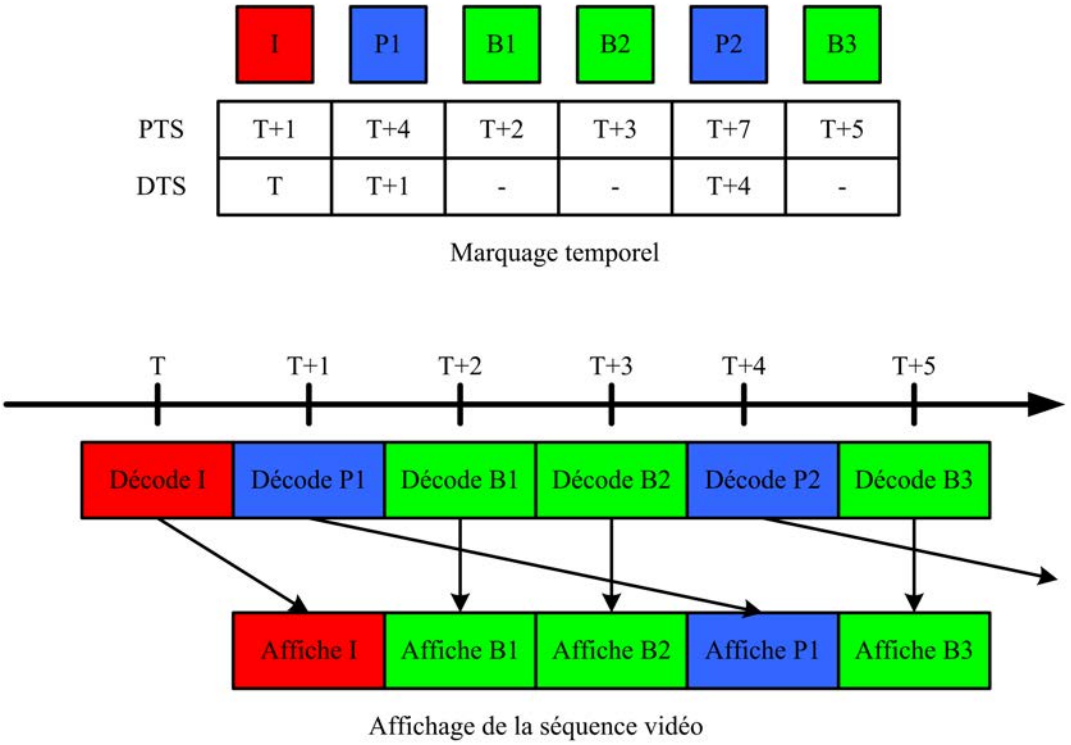


FIG. 3.12 – *Utilisation des marqueurs PTS et DTS pour l'affichage des images*

## 3.5 Conclusion

*MPEG-1* avait pour but de coder de la vidéo (flux audio et vidéo) à un débit n'excédant pas 1,5 Mbits/s. La norme fut certifiée par l'Organisation Internationale de Normalisation (l'ISO) et comprends cinq parties :

**ISO/IEC 11172-1 :** La partie système, validée en 1993 et corrigée en 1996 et 1999.

**ISO/IEC 11172-2 :** La partie vidéo, validée en 1993 et corrigée en 1996 et 1999.

**ISO/IEC 11172-3 :** La partie audio, validée en 1993 et corrigée en 1996.

**ISO/IEC 11172-4 :** Compliance testing, validée en 1995. Elle spécifie comment élaborer des tests vérifiant si les flux et les décodeurs vérifient les spécifications des trois premières parties.

**ISO/IEC 11172-5 :** Software simulation, validée en 1998. Cette partie est en fait un rapport technique contenant l'implémentation complète de programmes pour les trois premières parties. Le code source de ces programme ne fait toutefois pas partie du domaine public.

Depuis 1982, la norme CCIR 601 définit à l'échelon mondial les paramètres de la vidéo numérique. Cette norme est aussi utilisée dans le milieu télévisuel. Les télévisions européennes émettent en 25 images par seconde d'une résolution de 720*576 (PAL/SECAM), tandis que les télévisions américaines émettent en 30 images par seconde d'une résolution de 720*486 (NTSC). Ces valeurs concernent les images de la télévision "standard" avec une image 4/3. Dans les deux cas, chaque image est composée de deux trames. La première trame est composée des lignes impaires de l'image, la deuxième reprend les lignes paires. On dit alors que le signal de télévision est entrelacé (voir figure 3.13). Les télévisions émettent donc 50 trames par seconde en Europe et 60 aux Etats-Unis. La norme CCIR 601 impose aussi à la vidéo numérique d'être codée selon le modèle YUV 4 : 2 : 2, 8 bit par pixels dans chacune des trois matrices Y U V. En calculant le débit du PAL/SECAM et du NTSC on obtient :

$$\underbrace{720 \times 576 \times 25 \times 8}_{Y} + \underbrace{360 \times 576 \times 25 \times 8}_{U} + \underbrace{360 \times 576 \times 25 \times 8}_{V} = 19,78 Mo/s$$

$$\underbrace{720 \times 486 \times 30 \times 8}_{Y} + \underbrace{360 \times 486 \times 30 \times 8}_{U} + \underbrace{360 \times 486 \times 30 \times 8}_{V} = 20,02 Mo/s$$

La norme *MPEG-1* permet d'obtenir des taux de compression élevés avec une qualité d'image relativement correcte. Cette qualité d'image, proche de celle obtenue lors de la lecture d'une cassette VHS, peut convenir pour la plupart des utilisateurs, mais pas pour une utilisation professionnelle.

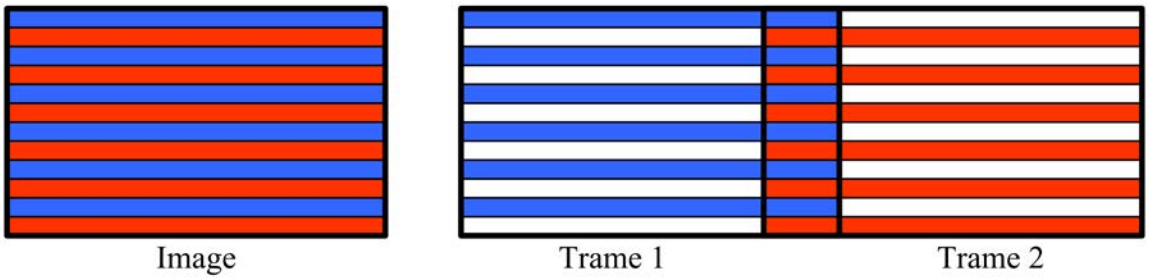


FIG. 3.13 – *Image entrelacée*

Pour obtenir ce résultat, *MPEG-1* réduit la résolution des images. Il passe de 720*576 (720*486) pixels de luminance(Y) et 360*576 (360*486) pixels de chrominance bleue et rouge (U et V), à 360*288 (360*243) pixels en luminance et 180*288 (180*243) pixels en chrominance bleue et rouge. Cette réduction s'effectue en laissant tomber une trame sur deux et sur chaque trame restante, en laissant tomber un pixel sur deux. Vu que *MPEG-1* ne conserve plus qu'une trame sur deux, on obtient au final une vidéo qui n'est plus entrelacée. Cette réduction de résolution permet déjà d'obtenir un facteur de compression de 4.

Le changement de résolution s'accompagne d'un sous-échantillonnage des chrominances bleue et rouge, en passant d'un codage YUV 4 : 2 : 2 à un codage 4 : 2 : 0. Les matrices de chrominance passent ainsi de 180*288 (180*243) pixels à 180*144 (180*122) pixels, réduisant ainsi la quantité de données d'un facteur ⅔ (la figure 3.2 à la page 34 illustre cette réduction).

A ce stade-ci, on obtient des débits de 29,66 Mbits/s pour le PAL/SECAM et de 30,03 Mbits/s pour le NTSC. Le reste du chemin pour atteindre la barre des 1,5 Mbits/s passe par l'utilisation de images I, P et B.

La norme *MPEG-1* a beau être très puissante, elle souffre de défauts liés à la qualité de service, qui empêchèrent celle-ci de s'imposer dans le milieu professionnel, y compris dans le monde télévisuel.

Afin de réaliser ce chapitre nous nous sommes inspirés des documents suivants : [Bel05], [Wik06], [Cou99], [Cre01], [Dan05], [Gas05], [GG96], [Goo01], [HAL03], [MPG].

CHAPITRE 4

# MPEG-2

## 4.1 MPEG-2 Vs MPEG-1

Alors que *MPEG-1* visait le stockage de vidéo sur CD avec un débit n'excédant pas 1.5 Mbits/s, le *MPEG-2* lui s'adapte à un éventail d'applications beaucoup plus large : diffusion télé, DVD, post-production, haute définition, etc. *MPEG-2* offre des débits variables en fonction de l'application visée : de 4 Mbits/s à 100 Mbits/s. Il fut partiellement défini en 1994.

En fait *MPEG-2* n'est pas une norme unique, il peut être comparé à une sorte de boîte à outils offrant différentes normes que l'utilisateur peut choisir en fonction de ses besoins. Cette polyvalence est permise grâce à l'utilisation de profils et de niveaux.

Les profils déterminent le niveau de qualité désiré en fonction de l'utilisation choisie : diffusion, production, montage vidéo, etc. Ils reprennent les outils de codage et les algorithmes spécifiques à cette utilisation. Les niveaux, quant à eux, définissent pour chaque profil, des paramètres clés du flux tel que : la définition des images, le débit maximum, le nombre d'images par seconde, ..., imposant ainsi une limite quantitative.

Si sur le fond, le principe général de fonctionnement de *MPEG-2* est sensiblement identique à celui de *MPEG-1*, il existe de nombreuses différences de forme

entre les deux standards. Tout d'abord, *MPEG-2* permet la compression d'images entrelacées là où *MPEG-1* ne traite que les images en mode progressif (non entrelacé), et ceci bien évidemment pour servir à la télévision numérique. La prédiction de mouvement se base donc aussi sur les trames constituant les images. En *MPEG-2*, on peut faire afficher jusqu'à 120 trames/s, soit 60 images/s.

Ensuite, au lieu de n'avoir qu'un seul niveau de résolution, *MPEG-2* en permet quatre : basse résolution (352 x 288), résolution normale (720 x 576), haute résolution 1440 (1440 x 1152) et haute résolution (1920 x 1152).

En *MPEG-1*, lors de l'étape de quantification (images I et terme d'erreur des images P), la même matrice de quantification était utilisée pour les matrice de luminance (Y) et de chrominance (U, V). Avec *MPEG-2*, lorsqu'on utilise un format d'échantillonnage "4 : 2 : 2", on peut utiliser des matrice de quantification différentes pour la luminance et la chrominance. Les matrices U et V partagent quand même toujours les mêmes matrices.

Voici quelques autres améliorations proposée par *MPEG-2* :

- *MPEG-1* ne permettait que l'échantillonnage "4 : 2 : 0", tandis que *MPEG-2* permet l'échantillonnage "4 : 4 : 4", "4 : 2 : 2", "4 : 1 : 1", "4 : 2 : 0".
- La taille des macroblocs *MPEG-1* était 16*16 pixels, tandis qu'en *MPEG-2* ils font soit 16*16 ou 16*8 pixels, apportant ainsi une plus grande précision.
- En *MPEG-2* la précision des vecteurs de mouvement passe de un à un demi pixel.
- Amélioration du codage VLC grâce à des tables plus performantes.
- Au point de vue audio *MPEG-2* supporte en plus du *MPEG-1 Layer I, II, III*, le son surround et Dolby AC-3. Il peut aussi contenir jusqu'à 8 pistes audio multiplexées.
- ...

MPEG-2 a introduit un nouveau schéma de compression audio appelé MPEG-2 AAC (Advanced Audio Coding).

Ce sont ici quelques améliorations techniques et il y a en bien d'autres. Cependant les principales nouveautés qu'offre le *MPEG-2* sont les notions de profils et niveaux ainsi que de codage hiérarchique (scalable coding). Ces notions sont expliquées plus en détails dans les deux sections suivantes.

## 4.2 Codage hiérarchique

Le *MPEG-1* codait les vidéos sans se soucier des capacités de l'équipement qui servira à décoder cette vidéo. L'utilisateur final n'avait aucun contrôle sur le flux qu'il décodait. *MPEG-2* permet un contrôle du flux en offrant différentes couches de codages. C'est-à-dire que lors du codage, dans un même flux, on va pouvoir encoder différents éléments permettant d'obtenir différents niveaux de qualité lors de la lecture. Dans un même flux peuvent cohabiter une couche de base de qualité inférieure et une ou deux couches supplémentaires d'améliorations. Par exemple la couche de base sera utilisée pour les décodeurs bon marché, ou lorsqu'on accède à une vidéo via une connexion à faible débit, tandis que les couches d'améliorations seront utilisées par les décodeurs plus performants, ou pour les connexions à haut débit. Voici les principaux codages hiérarchiques :

**Codage hiérarchique temporel** (temporal scalability). Il permet d'obtenir différentes fréquences d'images. La couche de base contient des images à une fréquence réduite (30 images/s par exemple) tandis que les couches d'amélioration contiennent des images supplémentaires pour obtenir une fréquence d'image supérieure (60 images/s par exemple).

**Codage hiérarchique spatial** (spatial scalability). Il permet d'obtenir différentes résolutions pour les images encodées. La couche de base offre une résolution minimum, et les couches d'amélioration permettent d'obtenir des résolutions supérieures.

**Partitionnement des données** (data partitionning). La DCT permet de passer d'une représention de blocs d'une image en une représention fréquentielle. La couche de base contient les plus basses fréquences (les plus importantes car l'oeil humain y est plus sensible) pour une qualité d'image restreinte. Les fréquences plus élevées restantes sont codées par paliers dans les couches supérieures afin d'obtenir une qualité d'image supérieure. Un exemple de cette technique est illustré à la figure 4.1.

**SNR Scalability** (signal to noise ratio scalability). Ce processus permet aussi de récupérer une partie de la qualité d'image perdue lors de la quantification. L'encodeur va quantifier, à un certain degré, les coefficients résultant de la DCT et les encoder dans la couche de base. La quantification va amener une "erreur de quantification". Celle-ci est la différence entre le bloc originel à coder, et le bloc calculé en utilisant la fonction de quantification inverse. Ensuite cette erreur va être à son tour quantifiée, puis encodée dans la couche supérieure.

D'autres types de codage hiérarchique ont été expérimenté avec *MPEG-2* mais ont été abandonnés au profit d'autres méthodes plus faciles apportant la même qualité d'image.

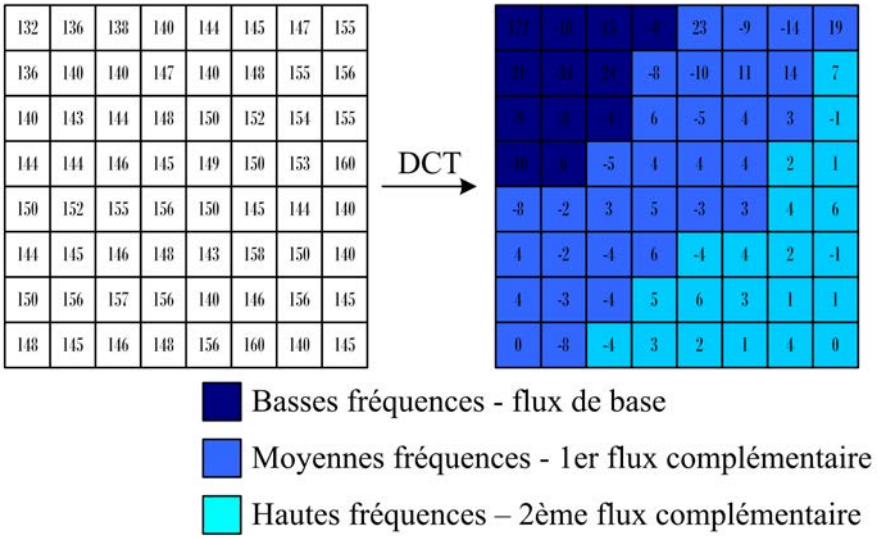


FIG. 4.1 – *Illustration du data partitionning*

## 4.3 Profils et niveaux

En fait *MPEG-2* n'est pas un standard unique, il peut être comparé à une sorte de boîte à outils offrant différents standards que l'utilisateur peut choisir en fonction de ses besoins. Cette polyvalence est permise grâce à l'utilisation de profils et de niveaux.

Les profils, au nombre de six, déterminent le niveau de qualité désiré en fonction du type d'application visé : diffusion, production, montage vidéo, .... Ils reprennent les outils de codage et les algorithmes spécifiques à cette utilisation. Ils définissent ainsi le type des images qui seront utilisées (I, P, B), les procédés de codage employés, etc. Il est à noter que les décodeurs ne sont pas obligés d'implémenter tous les profils du standard *MPEG-2*. Il peuvent se contenter de n'implémenter que les profils destinés à l'usage visé par l'utilisateur.

Voici les différents profils proposé par *MPEG-2* :

**Le profil simple** (simple profile : SP) ne comporte pas d'images B. Cela simplifie le codage et le décodage car sans les image B, il n'y a plus de réorganisation des images. De plus cela permet de réduire les délais de codage et de décodage. Ce profil encode les images en format "4 : 2 : 0". Il convient très bien pour les application de vidéoconférences. Les autres profils concernent principalement la télévision terrestre, satellite, haute définition (TVHD).

**Le profil principal** (main profile : MP) supporte les images B, augmentant ainsi la qualité d'image mais entraînant un temps d'encodage plus élevé dû à la

réorganisation des images. Les décodeurs utilisant ce profil sont aussi capables de décoder des vidéo *MPEG-1*. Comme le profil simple, il encode exclusivement les image en format “4 : 2 : 0”. C’est le profil le plus utilisé.

**Le profil SNR** (SNR scalable profils : SNRP) utilise le SNR scalability. Il utilise une couche de base et une couche d’amélioration.

**Le profil spatial** (spatial scalable profile : SSP) utilise le codage hiérarchique spatial. Il utilise une couche de base et deux couches d’améliorations.

**Le profil haut** (high profile : HP) ajoute la possibilité d’utiliser le codage 4 : 2 : 2 et intègre le SNRP et le SSP. Il utilise une couche de base et deux couches d’améliorations.

**Le profil 4 : 2 : 2** Comme son nom l’indique, il utilise le codage 4 : 2 : 2. Il permet aussi d’utiliser des GOP[1] plus courts, pouvant même ne se réduire qu’a des images I.

Les niveaux, quant à eux, définissent pour chaque profil des paramètres clés du flux tel que : la définition des images, le débit maximum, le nombre d’images par seconde, . . ., imposant ainsi des limites quantitatives. Les différents profils offrent une compatibilité ascendante. C’est-à-dire qu’un profil donné est exploitable et manipulable par les profils qui lui sont supérieurs. Les niveaux sont au nombre de quatre que l’on retrouve dans le tableau 4.2

Avec six profils et quatre niveaux on obtient 24 combinaisons. Cependant toutes ces combinaisons n’ont pas été jugées utiles et/ou réalisables (voir tableau 4.2). Le couple Main Profil@Main Level (MP@ML) est le plus utilisé tant au niveau software qu’au niveau hardware. Il est notamment utilisé pour la télévision standard (SDTV) et pour le DVD. Il reprend les trois types d’images I, P, B avec un GOP de 12[2] images sous la forme IBBPBBPBBPBBPI. . ..

La télévision haute définition (HDTV) utilise quant à elle le MP@H14L pour les images 4/3 et MP@HL pour les images 16/9. Les opérateurs qui veulent utiliser le codage 4 : 2 : 2, reprennent les mêmes niveaux combinés avec le profil haut. Il est à noter qu’avec les évolutions technologiques, la TV haute définition privilégie maintenant le standard *MPEG-4*.

Le profil 4 : 2 : 2P fut créé pour les besoin de la post-production. Ce profil aussi appelé “studio profile” ou “professional profile” fut ajouté pour combler les manques du duo MP@ML. Il permet d’encoder des images en 4 : 2 : 2, sans s’encombrer de la complexité liée au profil haut. En effet, en post-production les SNR ou spatial scalability ne représentent pas un impératif, tandis que le codage 4 : 2 : 0, lui, n’est pas satisfaisant. Ensuite le GOP du MP@ML n’est pas suffisamment précis pour le

[1] GOP : Group Of Pictures, voir compression MPEG-1

[2] 12 images P et/ou B entre deux images I

| NIVEAUX | PROFILES | | | | | |
|---|---|---|---|---|---|---|
| | **Simple**<br>I & P<br>4 : 2 : 0 | **Principal**<br>I, P & B<br>4 : 2 : 0 | **SNRP**<br>I, P & B<br>4 : 2 : 0 | **Spatial**<br>I, P & B<br>4 : 2 : 0 | **Haut**<br>I, P & B<br>4 : 2 : 0 ou<br>4 : 2 : 2 | **4.2.2P**<br>I, P & B<br>4 : 2 : 0 ou<br>4 : 2 : 2 |
| **Haut**<br>1920*1152 | | **MP@HL**<br>60 img/s<br>80 Mb/s | | | **HP@HL**<br>60 img/s<br>100 Mb/s | **4.2.2P@HL**<br>60 img/s<br>3OO Mb/s<br>1920*1088 |
| **Haut 1440**<br>1920*1152 | | **MP@H14L**<br>60 img/s<br>60 Mb/s | | **SSP@H14L**<br>60 img/s<br>60 Mb/s | **HP@H14L**<br>60 img/s<br>80 Mb/s | |
| **Principal**<br>720*576 | **SP@ML**<br>30 img/s<br>15 Mb/s | **MP@ML**<br>30 img/s<br>15 Mb/s | **SNRP@ML**<br>30 img/s<br>15 Mb/s | | **HP@ML**<br>30 img/s<br>20 Mb/s | **4.2.2P@ML**<br>30 img/s<br>5O Mb/s<br>720*608 |
| **Simple**<br>352*288 | | **MP@SL**<br>30 img/s<br>4 Mb/s | **SNRP@SL**<br>30 img/s<br>4 Mb/s | | | |

FIG. 4.2 – *Table de niveaux et profils MPEG-2*

montage vidéo. Le GOP du profile 4 : 2 : 2P peut être limité à 1 (que des images I) ou à deux (alternance d'images I et B).

## 4.4 Débit fixe ou débit variable

*MPEG-2* offre la possibilité d'encoder les séquences vidéos en débit fixe[3] ou en débit variable[4], possibilité que ne proposait pas *MPEG-1*. Lorsqu'on analyse une séquence vidéo image par image, on se rend compte que la complexité des images varie au fil de la vidéo. Par exemple, une scène avec beaucoup de mouvements (la retransmission d'une épreuve sportive) nécessite plus de débit qu'une scène pratiquement immobile (une émission politique en plateau).

Si on encode une séquence vidéo en débit fixe, le débit restera toujours constant

[3] Constant Bit Rate ou CBR
[4] Variable Bit Rate ou VBR

tandis que la qualité des images sera variable. En effet, les images plus complexes seront de moins bonne qualité que les images plus simples, car pour maintenir le débit constant, les images complexes seront compressées avec un taux supérieur à celui des images plus simples.

Par contre en utilisant un débit variable, on assure une qualité d'image constante tout le long de la séquence vidéo. Le débit sera fonction de la complexité de chaque image. Cette technique permet également une meilleure gestion du volume du fichier vidéo. La gain de volume est bien sûr fonction du nombre d'images simples par rapport au nombre d'images complexes. Un DVD encodé à débit variable sera généralement moins volumineux que s'il était encodé à débit fixe.

## 4.5 L'encodage double passe

Beaucoup d'applications *MPEG-2* encodent les images d'une vidéo en temps réel, c'est-à-dire que les images sont encodées sans connaissance des autres images constituant la vidéo à encoder. Néanmoins il existe des applications qui dans un premier temps encodent la vidéo en débit constant, tout en calculant des statistiques concernant la complexité du codage des images : la première passe. Ensuite les informations récoltées sont analysées pour déterminer les paramètres d'encodage de la seconde passe (notamment le débit). Lors de cette dernière étape, la vidéo sera encodée en débit variable. Le codage double passe permet d'optimiser le codage à débit variable, car on dispose d'informations sur l'entièreté des images de la vidéo, assurant ainsi une qualité d'image constante. Cette technique est utilisée pour l'encodage de vidéo sur DVD. [5]

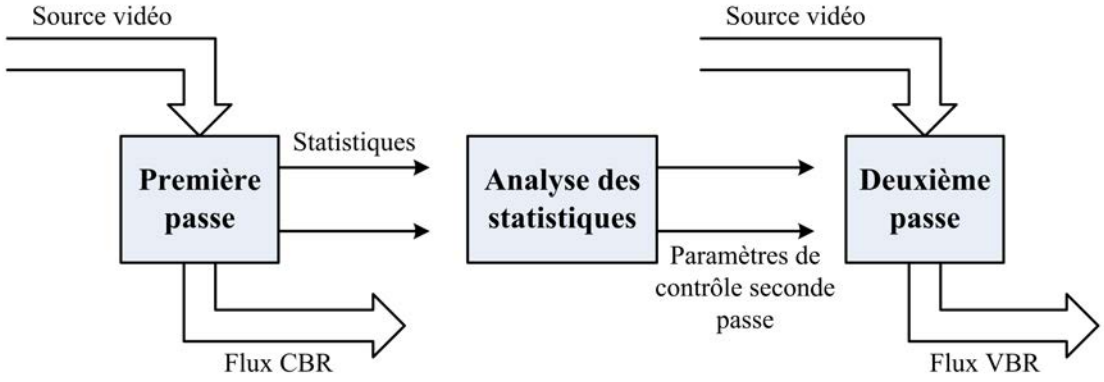


FIG. 4.3 – *Système d'encodage double passe*

[5]Pour de plus amples explications voir : http://www.research.ibm.com/journal/rd/434/westerink.html

## 4.6 Conclusion

Le *MPEG-2* a connu un véritable succès à l'échelle mondiale. De par le monde des millions de décodeurs (les lecteurs DVD par exemple) basés sur le standard *MPEG-2* sont encore en service à l'heure actuelle. Le *MPEG-2* a aussi marqué l'avènement de la télévision numérique que nous connaissons aujourd'hui, qu'elle soit diffusée par satellite, câble, ADSL et même la TNT[6].

La politique qui fut choisie dès le début du développement du standard était de ne pas limiter *MPEG-2* au niveau technologique de l'époque, mais d'en faire un standard évolutif pouvant s'adapter aux progrès technologiques. Ainsi *MPEG-2* n'a cessé de s'améliorer au fil des années, continuant à accroître la qualité des images. En fait, *MPEG-2* , comme *MPEG-4* (on le verra plus loin), ne définit pas les moyens à utiliser pour encoder une séquence vidéo, mais fournit la sémantique du décodage. Pour faire simple, peu importent les moyens utilisés pour encoder les images, le flux binaire produit doit être compatible avec les lecteurs *MPEG-2*. Ceci laisse donc la main libre aux industriels pour développer de nouveaux encodeurs plus performants. Cette politique fut payante car depuis 1994, à qualité d'image égale, le débit nécessaire a été divisé par un facteur de trois à quatre.

A l'heure actuelle on estime que *MPEG-2* a atteint ses limites technologiques et qu'il n'est plus guère possible d'apporter de nouvelles améliorations majeures. Le nouveau standard qu'est *MPEG-4* lui a donc succédé il y a quelques années. Celui-ci est performant, prometteur et lui aussi évolutif. Les premières applications à grande échelle de *MPEG-4* ont vu le jour au cours de l'année 2005, avec notamment la télévision numérique terrestre en France et les nouvelles générations de lecteurs DVD. Ceux ci sont compatible *MPEG-2* et *MPEG-4*.

*MPEG-2* est un standard divisé en 10 parties dont les trois premières parties ont obtenu le statut de standard interational :

**ISO/IEC 13818-1 :** La partie système qui spécifie la façon de combiner un ou plusieurs flux élémentaires vidéo, audio ou données en un ou plusieurs flux convenant au transport et/ou au stockage.

**ISO/IEC 13818-2 :** La partie vidéo spécifiant les profils et les niveaux. Approuvée en 1994 et deux profils supplémentaires ont été rajoutés en 1996 : le profil 4 : 2 : 2 et le Multiview Profile.

**ISO/IEC 13818-3 :** La partie audio qui doit être compatibles avec *MPEG-1*.

**ISO/IEC 13818-4 :** Compliance testing, validée en 1996. Elle spécifie comment élaborer des tests vérifiant si les flux et les décodeurs vérifient les spécifications des trois premières parties.

[6] Télévision Numérique Terrestre

**ISO/IEC TR 13818-5** Software simulation, validée en 1996. Cette partie est en fait un rapport technique contenant l'implémentation complète des programmes pour les trois premières parties. Le code source de ces programmes ne fait toutefois pas partie du domaine public.

**ISO/IEC 13818-6 :** Digital Storage Media Command and Control (DSM-CC) spécifiant un ensemble de protocoles fournissant des fonctions et des opérations de contrôle pour la gestion des flux *MPEG-1* et *MPEG-2* pour des applications Client-Serveur. Approuvée en 1996.

**ISO/IEC 13818-7 :** Advanced Audio Coding (AAC) spécifiant un algorithme de codage audio multi-canaux non compatible *MPEG-1*. Approuvée en 1997.

**Partie 8 :** Cette partie fut abandonnée car elle ne suscitait pas un grand intérêt de la part du monde industriel. Elle concernait le codage vidéo dont la longueur des échantillons d'entrées était de 10 bits

**ISO/IEC 13818-9 :** Extension for Real-time Interface (RTI). Approuvée en 1996.

**ISO/IEC 13818-10 :** Conformance extensions for Digital Storage Media Command and Control (DSM-CC), partie spécifiant les test de conformance de la partie 6 : DSM-CC.

Afin de réaliser ce chapitre nous nous sommes inspirés des documents suivants : [Bel05], [Cou99], [Dan05], [Gas05], [GG96], [HAL03], [MPG], [WRG99].

CHAPITRE 5

# MPEG-4

## 5.1 Introduction

*MPEG-4* est un standard *ISO/IEC* développé par *MPEG*. Englobant *MPEG-1* et *MPEG-2* , *MPEG-4* a vocation à devenir la future norme générique pour applications multimédia, capable de répondre à toute une gamme de besoins émergeant à l'intersection des industries de l'informatique, des télécommunications et de l'audiovisuel grand public. Elle doit en particulier permettre à l'utilisateur d'interagir aisément à plusieurs niveaux avec l'application. Elle a aussi pour prétention de servir aux applications très faible débit et de fournir une plus grande robustesse aux erreurs. Dans cette optique, une philosophie orientée objet a été choisie pour la vidéo comme l'audio. On parlera alors de scène composée d'objets visuels et d'objets sonores. *MPEG-4* est plus qu'une simple amélioration des générations précédentes, il définit une nouvelle conception de la vidéo en dépassant les concepts de codage et de compression. Le *MPEG-4*, en effet, définit à la fois une syntaxe du flux vidéo codé et un ensemble d'outils permettant la diffusion de cette vidéo sur des réseaux hétérogènes.

La version 1 de *MPEG-4* a été finalisée en octobre 1998 et la version 2 est en cours de complétude.

## 5.2 Description et objectifs

### 5.2.1 Notion d'objets média

Le principe adopté par *MPEG-4* est celui d'une description autonome de contenu. L'image n'est plus codée dans son intégralité, en la considérant donc indépendamment de son contenu comme une surface de X sur Y pixels, mais apparaît comme une composition de différents objets média[1]. En plus de définir des objets, *MPEG-4* définit une arborescence structurant ces objets en les décomposant en d'autres objets. Une personne pourra être décomposée en sa voix, son visage, ses mains, et le reste du corps. La scène vidéo sera donc ainsi décomposée selon une hiérarchie de ces objets média et selon leur disposition spatiale.

*MPEG-4* permet également de manipuler les objets média de différentes façons et ainsi de rendre interactif la scène vidéo transmise. On pourra par exemple :

- modifier la place d'un élément ;
- changer le point de vue de la scène ;
- changer les propriétés d'un objet (dimension, couleur, ...).

Les objets média pourront être identifiés et recevoir un code qui leur permettra de ne pas être réutilisés à outrance. On pourra ainsi protéger les droits d'un auteur sur son objet média.

La figure 5.1 donne l'exemple d'une scène décomposée suivant l'idée de *MPEG-4*.

Ainsi, dans l'arborescence de cette hiérarchie, on trouve :

- des images fixes (l'arrière plan,un meuble, ...) ;
- des objets vidéos (objets en mouvement, un personnage, ...) ;
- des objets audios (la voix de la personne, un fond musical, ...) ;
- des objets données (textes, sous-tire, ...).

Ces objets peuvent être naturels[2] ou synthétiques[3], en 2D ou en 3D. Un texte, un graphique, un son synthétique, ..., sont également considérés comme des objets. Chacun de ces objets est codé indépendamment des autres, et isolé de l'environnement.

[1] Objets audiovisuels
[2] Provenant d'une caméra
[3] Provenant d'une ordinateur

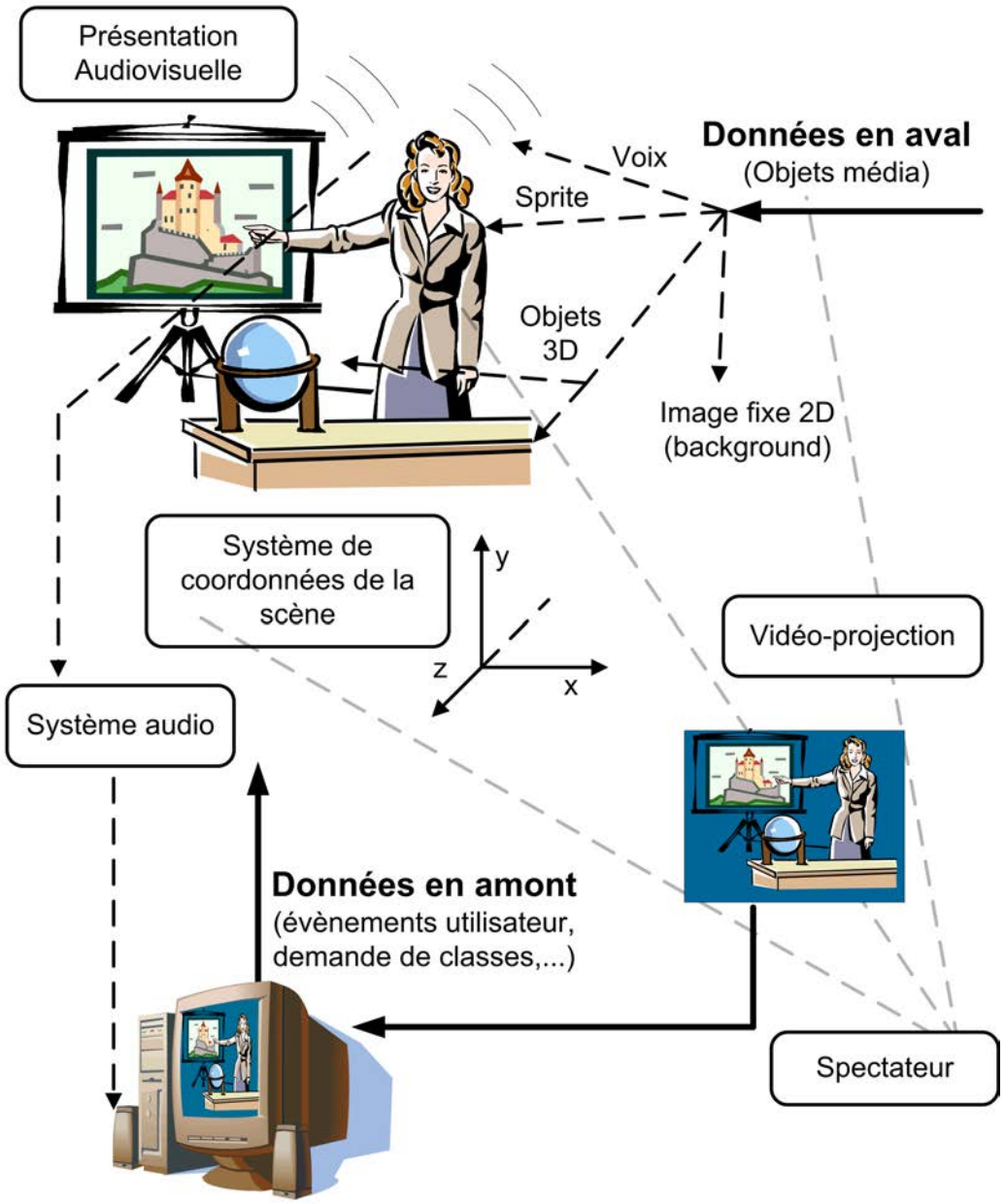


FIG. 5.1 – *Description d 'une scène MPEG-4*

### 5.2.2 Description de scènes : BIFS

*BIFS*[4] est un langage de description qui permet de composer hiérarchiquement une scène avec les composants qui la constituent, aussi différents soient-ils. Il définit donc :

- les objets composant la scène sous forme de noeuds d'un arbre ;
- la structure de la scène sous forme d'une hiérarchie ;
- les comportements des objets ;
- l'interactivité avec ces objets.

Le groupe *MPEG* s'est servi du standard actuel de description de mondes 3D, le *VRML*[5], pour créer *BIFS* en lui ajoutant des particularités comme la gestion de la 2D et des notions de timing.

Un objet dans une scène a un emplacement physique et un emplacement temporel, il est donc codé en 4D. Chaque objet possède son propre système de coordonnées local. Ce repère permet de manipuler cet objet dans la scène par rapport au repère d'un objet parent.

La description d'une scène est codée et encapsulée dans des paquets appelés unités d'accès de la même façon que sont encapsulés les objets vidéos et audios codés. *BIFS* ne contient pas les objets mais juste la description de la scène et les propriétés des nœuds (donc des objets média).

Les noeuds dans l'arbre de description d'une scène et les objets média finaux situés sur les feuilles contiennent des paramètres concernant l'aspect de l'objet. Cela permet de modifier son comportement de plusieurs façons possibles comme de modifier l'échelle d'une figure géométrique, le ton d'un son ou de diminuer la qualité d'un objet si un système de codage hiérarchique est utilisé.

Il faut imaginer que les flux contenant les données média et ceux contenant les descripteurs d'une scène *BIFS* sont des flux différents. Le premier contient l'objet codé et éventuellement des notions de codage hiérarchique, alors que le second contient la façon dont les objets sont placés dans une scène et la façon d'interagir avec eux.

La figure 5.2 illustre une description d'une scène au format *BIFS*.

[4]BIFS : BInary Format for Scene
[5]VRML : Virtual Reality Modeling Language

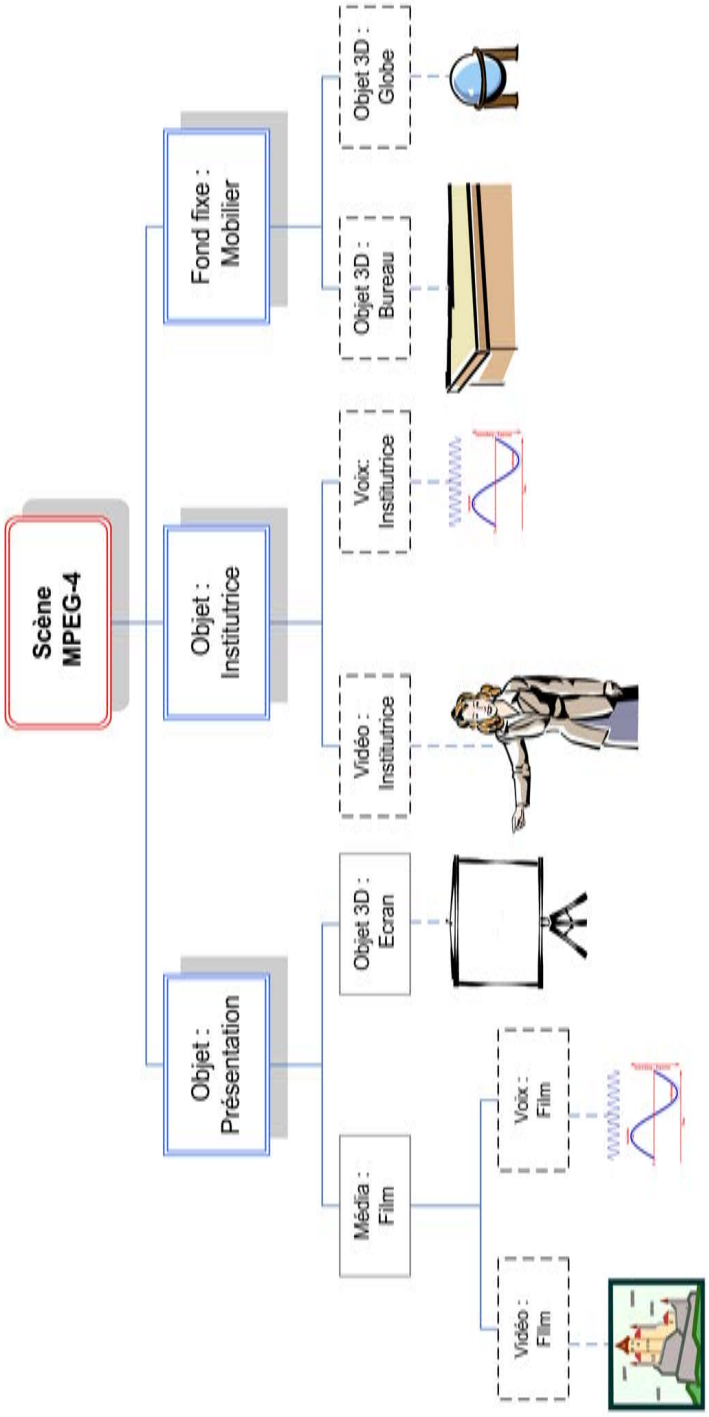


FIG. 5.2 – *Description d'une scène au format* BIFS

*MPEG-4* fournit donc des méthodes de codage pour les objets individuels. L'information nécessaire à la composition d'une scène est contenue dans la description de la scène. Celle-ci est codée indépendamment des objets média et transmise avec ceux-ci. Une grande attention est portée à l'identification des paramètres relatifs à la scène. Ces paramètres sont donnés par différents algorithmes qui codent de façon optimale les objets. *MPEG-4* place ces paramètres dans la partie description de la scène et ceci afin de pouvoir les modifier sans avoir à décoder les objets média.

Plus généralement, *MPEG-4* standardise la façon de décrire une scène en permettant par exemple :

- de placer un objet n'importe où dans un système de coordonnées ;
- d'effectuer des transformations géométriques ou acoustiques sur un objet ;
- de grouper des éléments média simples pour former un plus complexe ;
- de modifier les attributs d'un objet en transformant ses données ;
- de changer, interactivement, la vue et l'écoute d'une scène.

En résumé, les informations nécessaires à la description d'une scène sont :

- la façon dont les objets sont groupés ;
- le positionnement spatial et temporel des objets ;
- les différents paramètres des objets (couleur d'une forme synthétique, ...).

## 5.3 Le codage visuel

Les objets visuels codés peuvent être naturels ou synthétiques, en 2D ou en 3D, fixes ou mobiles. Un objet peut être un personnage se déplaçant, un objet fixe déplacé à un moment donné. Ces objets peuvent être eux-mêmes composés d'autres objets (les membres de la personne, sa tête, ...) et ainsi former une structure hiérarchique en arbre. *MPEG-4* fournit ainsi des outils permettant le codage et la manipulation des objets visuels.

Au niveau de la rétrocompatibilité avec les ancienne normes *MPEG*, le flux vidéo peut être codé (décodé) selon la méthode *MPEG* 1 ou 2 (gestion d'image rectangulaire, compensation de mouvement, ...) ou en utilisant la notion d'objets et de contenu. L'idée d'un codage basé sur le contenu implique que MPEG4 puisse coder séparément les différents objets vidéo d'une scène, afin de permettre une gestion simplifiée de l'interactivité : manipulation et représentation des objets vidéo, ainsi que le mélange entre objets naturels et objets synthétiques (comme par exemple une scène avec un fond virtuel et des personnages réels). Mais les algorithmes supplémentaires nécessaires à la gestion du codage basé sur le contenu ne devront être qu'un ensemble additionnel d'outils à ceux utilisés dans *MPEG* 1 ou 2.

### 5.3.1 La vidéo naturelle

Le principe du codage *MPEG-4* repose sur l'utilisation d'une approche basée sur le contenu. La difficulté est alors de séparer les objets et le fond d'une scène, pour ensuite en tirer des avantages pour la compression et les fonctionnalités supplémentaires que cela entraînera.

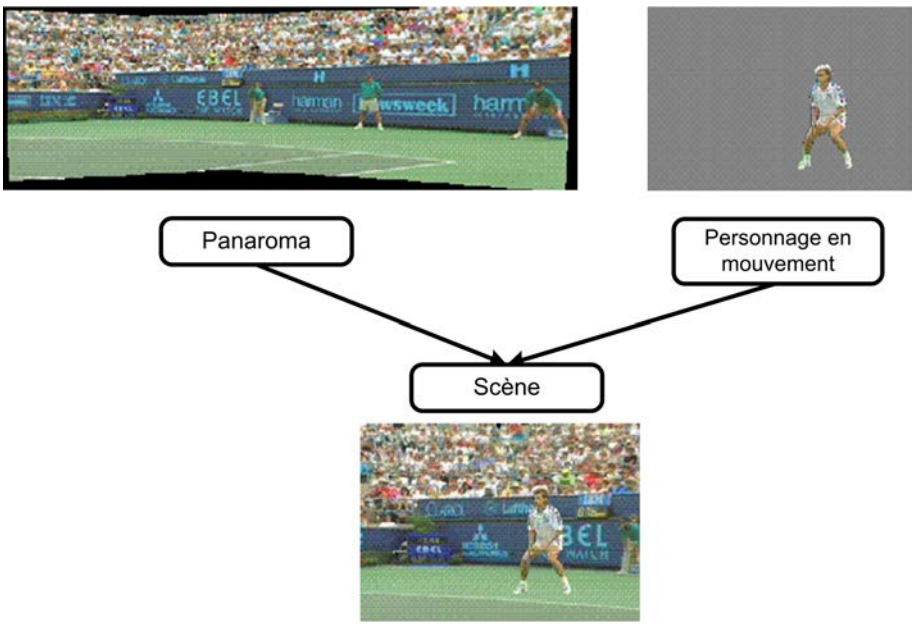


FIG. 5.3 – *Concept de base du codage* MPEG-4

La figure 5.3 issu de [ISO02] montre le concept de base du codage d'une séquence vidéo *MPEG-4*. Le fond de la séquence (image en haut à gauche) est isolé du joueur en mouvement (image en haut à droite) pour recréer un panoramique de fond de la scène complète. Le fond est alors transféré une seule fois afin de décrire l'arrière plan de la séquence. L'objet en mouvement est quant à lui stocké dans un buffer. Pour chaque image consécutive seuls les paramètres de caméra de l'arrière plan sont transmis. Cela permet à l'encodeur pour chaque image de reconstruire une arrière plan cohérent avec la position du joueur.

### 5.3.2 Les images fixes et textures

Le MPEG4 apporte également des améliorations pour la compression des images fixes et des textures. Le codage des images fixes n'est plus effectué en *JPG* (DCT) mais par une compression en ondelettes. Cette technique a donné naissance au format de compression *JPG2000*.

Prenons comme exemple un maçon désirant construire une oeuvre, celui-ci dispose d'un certain nombre de briques. Suivant la façon dont il assemble ces briques, il pourra construire toutes sortes de formes. En effet les briques sont des éléments simples bien adaptés à la construction. Les pixels peuvent être vus comme des briques car se sont des éléments simples permettant de construire n'importe quelle image. Un problème surgit lorsqu'il faut créer une grande oeuvre (image), celle-ci nécessite beaucoup de briques (pixels). Une solution est donc d'utiliser des briques (pixels) plus grosses là où on peut remplacer plusieurs petites briques (pixels) semblables. Si les petites pièces ne sont pas totalement identiques, le remplacement par une plus grosse ne causera qu'une légère modification.

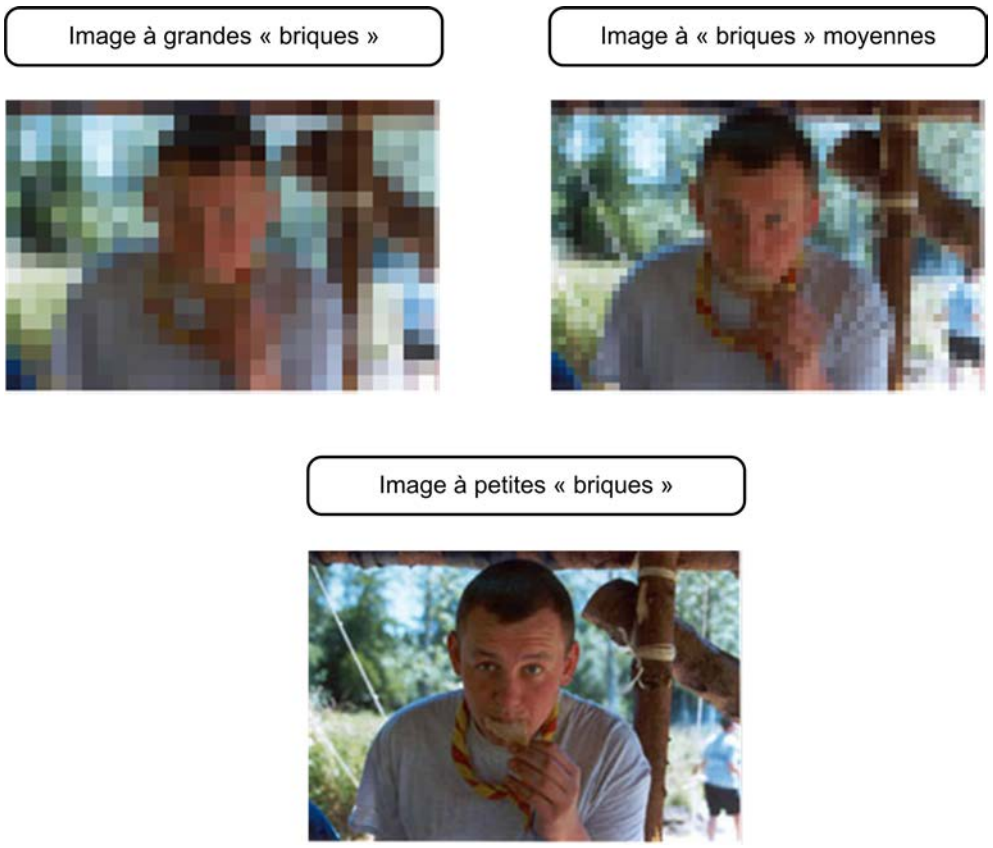


FIG. 5.4 – *Compression par ondelettes*

Pour construire le visage de la figure 5.4, des grosses briques sont placées pour réaliser une vue grossière du visage. Ensuite on utilise des briques de plus en plus petites suivant le niveau de détails des éléments du visage (bras, cous, yeux, ...). Cette approche permet d'utiliser un nombre d'éléments très réduit et est appelée multirésolution. Les ondelettes sont donc des ensembles de pixels de différentes tailles. Pour être précis, les ondelettes ne sont pas des gros blocs rectangulaires mais des paquets plus ou moins gros qui ont la forme d'une vague, ou d'une onde.

### 5.3.3 Les objets synthétiques

Les objets synthétiques représentent une importante partie de l'imagerie par ordinateur. Ces objets sont décrits suivant un modèle que l'on peut diviser en 4 parties :

- la description synthétique du visage et du corps humain ;
- l'animation des champs du visage et du corps ;
- le codage dynamique et statique du maillage avec les textures ;
- le codage des textures suivant les vues.

**A - L'animation du visage**

Un visage est un élément d'une scène vidéo ne se déplaçant pas ou peu mais subissant des déformations (sourire, froncement de sourcil, ...). *MPEG-4* fournit un ensemble d'outils pour pouvoir interpréter un mouvement et générer les modifications du sprite correspondant. Il ne sera ainsi plus nécessaire de renvoyer le visage (en fait les macroblocs le composant) mais de dire quelles modifications vont survenir.

La décomposition du processus de décodage d'un mouvement du visage se fait comme suit : les paramètres de définition du visage *FDP*[6] permettent d'afficher ou de garder en mémoire un visage ayant une forme neutre. Ces *FDP* permettent donc d'afficher tout de suite un visage ou d'y appliquer immédiatement une modification. Les paramètres d'animation du visage *FAP*[7] définissent les mouvements survenant sur un visage. Un *FAP* pourrait dire d'élargir le sourire de 2 centimètres. Les *FAP* sont ensuite interprétés par la table d'animation des visages *FAT*[8] qui traduit chaque requête d'une *FAP* en terme de déplacements de points composant le visage. Dans le cas de *FAP* survenant en même temps et dont les effets doivent tenir compte les uns des autres, des techniques d'interpolation de visage *FIT*[9] permettent de gérer ces déplacements multiples en décodant le *FAP* complexe arrivant en plusieurs *FAP* basiques et en liant leurs effets.

**B - L'animation du corps**

La technologie d'animation du corps proviendra directement de celle du visage, afin de garder l'esprit de standardisation de la norme *MPEG-7*.

[6]FDP : Facial Definition Parametre
[7]FAP : Facial Animation Parametre
[8]FAT : Facial Animation Table
[9]FIT : Face Interpolation Technique

**C - Animation des maillages 2D/3D**

Un maillage est une façon de grillager une surface 2D en polygones (le plus souvent des triangles). Cette technique permet de réduire la complexité d'un objet. Cette décomposition permet d'appliquer des modifications sur les triangles ou sur les points les constituants par :

- déplacement de points ;
- modification de la géométrie ;
- plaquage d'une texture ;
- ...

Comme le montre la figure 5.5 issue de [ISO02b], il est donc possible, dans un but d'économie de bande passante, de représenter des formes animées sous forme de réseaux de triangles 2D ou 3D. La 3D n'est qu'une illusion dans ce cas, car la profondeur est simulée par *MPEG-4*, tout maillage 3D est converti en 2D lors de la création du flux.

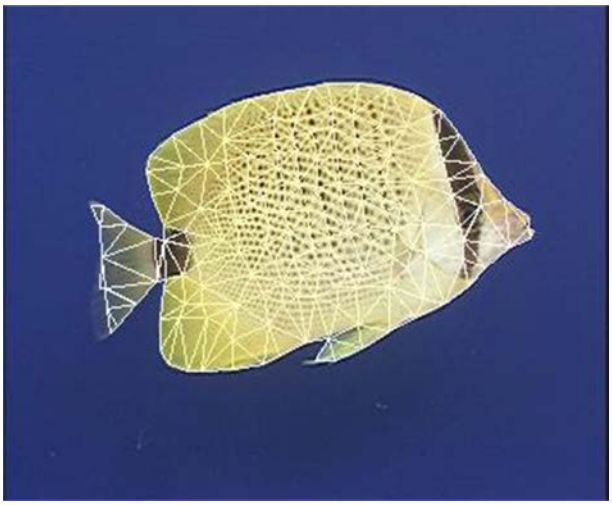

FIG. 5.5 – *Technique de maillage 2D/3D*

## 5.4 Les droits de propriété intellectuelle

*MPEG-4* traite le problème des droits de propriété intellectuelle par insertion dans les objets d'un code d'identification *IPI*[10] donnant des informations sur le contenu, le type du contenu et les droits attenant à l'objet en question. Les données contenues dans l'*IPI* peuvent donc différer pour des objets appartenant à une

[10]IPI : Intellectual Property Identification

même image (par exemple, des droits libres sur un arrière plan d'une séquence mais restreint sur le personnage). L'insertion de l'*IPI* au moment du codage implique également l'insertion de mécanismes de protection équivalant aux droits sur l'image (protection contre les copies, facturation, . . .).

## 5.5 Conclusions

*MPEG-4* est une révolution, ce standard apporte une nouvelle dimension dans la numérisation : la notion d'objet, ce qui apporte comme pour la programmation orientée objet des avantages tels que la réutilisation, la décomposition en sous-tâches plus simples, etc. Tout cela dans le but d'augmenter la rapidité de compression tout en améliorant la qualité de celle-ci. *MPEG-4* n'en est qu'à ses débuts, mais ses applications sont vastes et beaucoup de choses ont déjà été réalisées. Cependant *MPEG-4* ne résoud pas tous les problèmes liés à la gestion des documents multimédia. D'autres standards *MPEG* tels que le *MPEG-7* et le *MPEG-21* permettent respectivement de répondre aux problèmes liés à l'indexation et aux droits intellectuels des documents

*MPEG-4* est composé de 22 standards allant de [ISO/IEC 14496-1] à [ISO/IEC 14496-22].

Afin de réaliser ce chapitre, nous nous sommes inspirés des documents suivants : [ISO02b], [Lut02], [PCD03].

# Troisième partie

# Description de contenu

CHAPITRE 6

# Material eXchange Format

## 6.1 Introduction

Suite à la diffusion du *MPEG* dans le milieu de la production audiovisuelle, amenant une certaine normalisation des formats utilisés, l'EBU[1] et la SMPTE[2] établirent en 1998 un rapport commun indiquant les différents éléments manquants pour pouvoir établir un environnement réseau satisfaisant à toutes les attentes de l'industrie de la télévision.

En 1999, le Pro-MPEG [3] et l'Advanced Authoring Format Association (AAF Association) ont commencé à travailler sur un projet commun afin de palier aux manques relevés par l'EBU et la SMPTE. Ce projet avait pout but de permettre la transmission d'un sous-ensemble de métadonnées définies par l'AAF au travers de toute la chaîne de production de document audiovisuels, depuis la caméra jusqu'à l'archivage du document final. L'objectif de ce projet était que, lors de chaque transfert d'un point de la chaîne de production au suivant, les métadonnées puissent voyager dans le même fichier que celui des données audiovisuelles, chacun des maillons

[1] EBU : European Broadcasting Union ou UER pour Union Européenne de Radiodiffusion, est la plus grande association professionnelle de radiodiffuseurs nationaux (Europe, Afrique du Nord et Moyen-Orient) du monde.

[2] SMPTE : Society of Motion Picture and Television Engineers, société établissant des standards dans le monde de la vidéo

[3] Professional-MPEG Forum

de la chaîne pouvant bien entendu ajouter ses propres métadonnées au fichier. La solution apportée fut le Material eXchange Format (MXF).

Le “Professional-MPEG Forum” est une association de radiodiffuseurs, de fournisseurs, de constructeurs d'équipement et de concepteurs d'émissions ayant pour but de rendre interopérables les équipements télévisuels professionnels, en accord avec les exigences des radiodiffuseurs et autres utilisateurs finaux. Ce forum fut formé en 1998 pour promouvoir des standards ouverts pour les nouvelles applications télévisuelles professionnelles émergentes.

L'AAF Association, une organisation à but non lucratif, fut créée pour promouvoir le développement et l'adoption de la technologie AAF dans l'industrie audiovisuelle. Elle compte parmi ses membres des représentants des plus grandes entreprises du milieu. L'AAF Association a pour vocation d'aider les “créateurs de contenu” à profiter des bénéfices du digital. Parmi les créateurs visés on retrouve les professionnels de la télévision, de l'industrie du film, de la post-production et d'Internet. Parmi ses plus illustres fondateurs on retrouve Avid, the British Broadcasting Corporation (BBC), Cable News Network (CNN), Discreet, Four Media Company, Matrox, Microsoft, Pinnacle, Quantel, Sony, Turner Entertainment Networks, and the United States National Imaging and Mapping Agency.

L'Advanced Authoring Format est un format de fichier multimédia permettant aux créateurs de contenu d'échanger facilement des média digitaux et leurs métadonnées au travers des différents systèmes, plates-formes, et applications. L'AAF simplifie la gestion de projets, permet de gagner du temps et évite la perte de métadonnées précieuses qui sont souvent perdues lors du transferts de contenu audiovisuel entre les applications (soit parce qu'elle ne sont pas inclues dans le fichier, soit parce que l'application qui les reçoit n'est pas conçue pour les manipuler).

L'évolution des technologies dans la diffusion télévisuelle et dans les services offerts aux utilisateurs via la télévision numérique a entraîné des changements majeurs dans la production des contenus audiovisuels au sein même des studios. Ces changements se traduisent par une plus grande utilisation des systèmes informatiques tels que les serveurs, mais aussi par une plus grande dépendance à l'automatisation et à la réutilisation des documents. Le transfert de fichier, en plus de devoir transférer les métadonnées, va devoir s'adapter aux opérations informatisées et permettre le streaming pour les opérations en temps réels

Le format MXF est le fruit d'une remarquable collaboration entre fabricants et grandes organisations telles que Pro-MPEG, UER et AAF, afin de s'assurer que le format réponde bien à toutes les attentes. Il permet d'exploiter le contenu sur les diverses applications de la chaîne de production télévisuelle, il assure une exploitation plus efficace et une plus grande liberté de création dans un environnement réseau unifié.

La première étape fut d'identifier et de documenter les exigences des différents utilisateurs. Le caractère évolutif des technologies rend ce type de tâche assez compliqué. En effet il n'y a pas de meilleure solution sur laquelle on puisse baser son travail. Néanmoins de cette première étape découla un ensemble d'exigences dont on retrouve les principales à la figure 6.1.

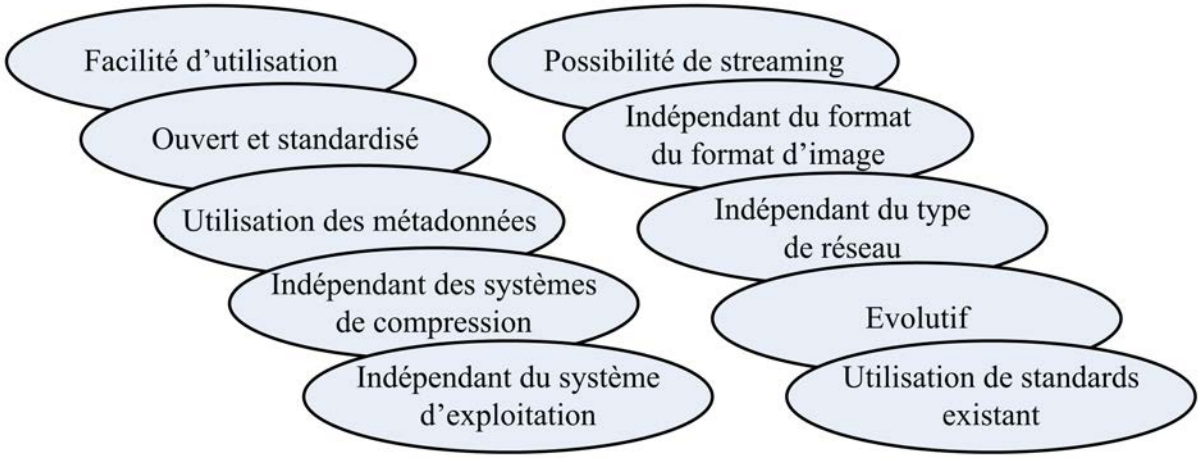


FIG. 6.1 – *Principales exigences de MXF*

Au final, MXF est un format de fichier servant à l'échange de contenu audiovisuel et des données et métadonnées associées, tout au long de la chaîne de production audiovisuelle. Il a été conçu et implémenté dans le but d'améliorer l'interopérabilité des fichiers entre les serveurs, les stations de travail et autres terminaux de création de contenu. MXF offre aussi une interopérabilité des contenus entre les différentes applications utilisées dans la chaîne de production télévisuelle.

En plus d'apporter une meilleure interopérabilité, MXF améliore le transfert des métadonnées. En développant MXF comme un nouveau format de fichier, les créateurs de ce format ont pu intégrer de nouvelles idées quant à l'implémentation et l'utilisation des métadonnées. Cela n'est pas seulement important pour l'utilisation propre des fichiers MXF, mais cela va permettre la création de nouveaux outils puissants pour la gestion des contenus audiovisuels, ainsi que d'améliorer les flux de création de contenu en éliminant la réintroduction de métadonnées répétitives.

MXF est indépendant de tout système de compression. Il simplifie l'intégration de systèmes utilisant *MPEG* et DV[4] mais aussi des futurs systèmes de compression qui seront développés ultérieurement. Ce qui signifie que le transport de ce type de fichier est indépendant du contenu et des fabricants d'équipement. Tous les processus qui doivent être effectués le sont en invoquant les codecs hardware ou software appropriés de façon transparente pour l'utilisateur.

[4]DV : Digital Video compression format, format de compression propriétaire de Sony

MXF est indépendant des systèmes d'exploitation, des plates-formes et des infrastructures réseaux. Il permet de se déplacer dans le fichier, de faire des transferts partiels de fichiers, d'accéder au fichier et de l'utiliser avant la fin de son transfert, c'est-à-dire qu'il permet le streaming.

## 6.2 Les apports du MXF

Jusqu'alors, il n'existait pas de format de fichier prenant en charge les données audiovisuelles et leurs métadonnées tout au long de la chaîne de production, c'est pourquoi l'apparition d'un format de fichier ouvert commun, utilisable par l'ensemble du secteur audiovisuel, traitant les métadonnées, a eu un grand impact sur la manière de traiter les documents audiovisuels.

La figure 6.2 montre les différentes étapes classiques de la création d'un programme de télévision. Lors du passage d'une étape à l'autre, il fallait transférer un mélange de bandes vidéo, de fichiers multimédia propriétaires, de documents Word, Excel, de fax, d'instruction orales, d'information reprises dans différentes bases de données, etc. Ce type de gestion conduisait souvent à la perte de métadonnées et donc dans la plupart des cas il fallait les recréer ou les réintroduire, ce qui est clairement inefficace et source d'erreurs.

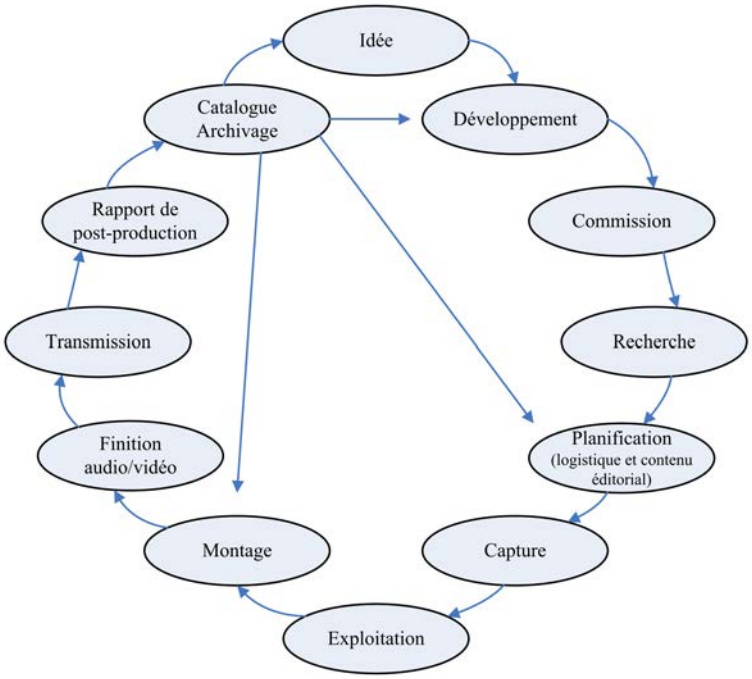


FIG. 6.2 – *Etapes classiques de création d'un programme de TV*

Si un format de fichier commun, comme MXF, se répand parmi toute la chaîne de production, les métadonnées transférées d'une étape de production à l'autre seront plus riches, permettant aux professionnels de se concentrer sur leur sujet plutôt que sur la recherche d'informations nécessaires pour les métadonnées.

Cela peut être démontré facilement avec le simple exemple d'un reportage sur la vie sauvage en Afrique. Lors du tournage des différents plans aux quatre coins de l'Afrique, si les coordonnées GPS de la caméra sont ajoutées comme métadonnées dans les fichiers MXF de chacun des plans, ces métadonnées seront utilisées tout au long de la création du reportage. Un processus automatique pourrait récupérer pour chaque plan le nom de chaque région où il a été tourné à partir des coordonnées GPS (comme illustré à la figure 6.3). Une telle automatisation réduit les tâches ordinaires affectées au personnel et améliore la précision des données stockées.

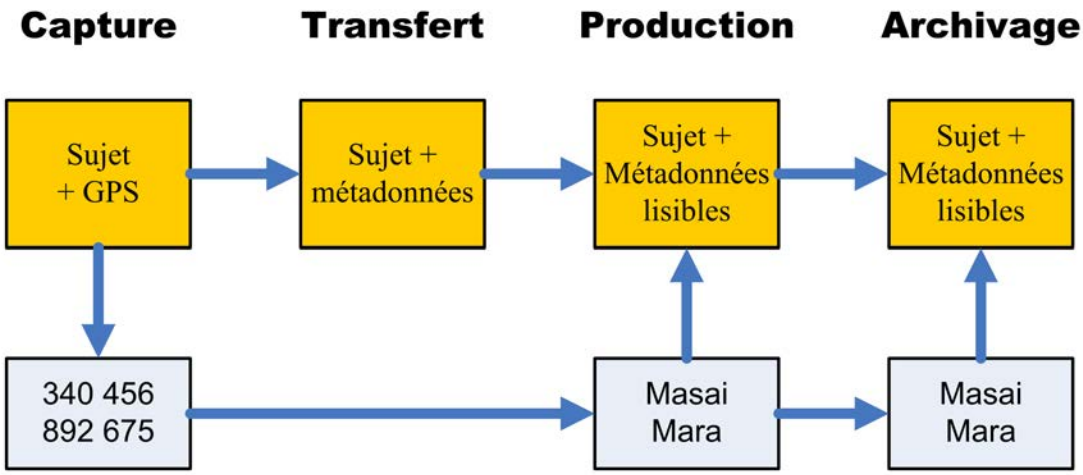


FIG. 6.3 – *Exemple d'une métadonnée GPS*

Les métadonnées de MXF ne sont pas seulement statiques. Alors que les informations à propos du format de l'image, du nombre d'images par secondes, ..., concernent l'ensemble du fichier, MXF peut aussi contenir des métadonnées liées à l'audio et/ou à la vidéo qui évoluent dans les temps. C'est-à-dire qu'on peut avoir des pistes de métadonnées en plus des pistes audio et vidéo. Un exemple simple et évident d'une telle piste serait une piste de sous-titres.

Une autre possibilité offerte par MXF réside dans le fait que les métadonnées ne doivent pas nécessairement être toujours présentes dans le fichier MXF. Ceci est utile si on prend le cas d'une série télévisée. Les informations concernant les acteurs sont toujours les mêmes d'un épisode à l'autre, c'est pourquoi MXF offre la possibilité de lier et d'associer des fichiers MXF avec des bases de données distantes. Ce mécanisme permet de n'avoir qu'une seule copie des informations qui seront identiques pour tous les épisodes de la série.

Un fichier MXF est en fait un container dans lequel on retrouve les éléments audio et vidéo (aussi appelé essence) et les métadonnées. Le format MXF apporte un certain nombre de capacités et caractéristiques avancées permettant de répondre aux exigences fondamentales des utilisateurs. En tant que container il contient des essences mais il ne les crée pas lui-même. C'est-à-dire qu'on introduit des essences (fichier WAV, MP3, MPEG-2, MPEG-4, ...) dans le fichier MXF, ce qui lui permet non seulement de pouvoir contenir une large gamme de types de fichier, mais lui permettra aussi de contenir les futurs types de fichier.

MXF est un format polyvalent permettant d'accomplir un certain nombre de tâches comme :

- stocker de simples travaux terminés avec leurs métadonnées ;
- stocker des fichiers dans un format qui permet le streaming ;
- contenir une playlist de fichiers et stocker des informations de synchronisation ;
- contenir n'importe quel format de compression.

Afin de bien comprendre, il est important de saisir la différence entre la diffusion en continu (streaming) et le transfert de fichier. Depuis leur début, les télévisions diffusent l'audio et la vidéo en continu. Ce qui est logique puisque le téléspectateur s'attend à recevoir ses émissions en temps réel et en continu, tandis que dans le monde informatique les ordinateurs s'échangent des données via un réseau en transférant des fichiers.

La diffusion en continu de média :

- permet la visualisation pendant le transfert, c'est-à-dire avant que toutes les données n'aient étés transférées ;
- offre des délais minimaux pour les actions en direct ;
- est un système de point à point sans goulets d'étranglement, elle permet un fonctionnement fiable et continu.

Tandis que le transfert de média :
- utilise des composants informatiques standard et bon marché :
- permet de stocker les média sur une large variété de supports et d'appareils (disques durs, bandes, CD, DVD, ...) ;
- apporte la flexibilité dans l'échange, le partage et la distribution de données.

Ces deux techniques présentent toutes deux leurs propres avantages et continueront d'être utilisées. Dès lors il est essentiel qu'il y ait un certain degré de compatibilité entre ces deux techniques afin qu'elles puissent cohabiter et s'échanger des documents. C'est dans cette idée qu'a été conçu le MXF, il permet à la fois la diffusion en continu et le transfert de fichier. On va donc pouvoir tirer avantage de

la flexibilité du AAF en postproduction, ensuite par simple conversion "invisible", le fichier MXF pourra être enregistrer sur bande ou sur un serveur de stockage.

Avec MXF, les équipes opérationnelles et créatives pourront se concentrer sur leur travail et ne plus se soucier des problèmes de compression. Cependant, à l'heure actuelle, il n'existe pas de format de compression unique convenant à toutes les applications utilisées, et cette situation risque de continuer encore longtemps. C'est pourquoi MXF est indépendant du format de compression, offrant les mêmes services sans se soucier du format. Ceci permet aux fabricants de proposer des équipements pouvant utiliser plusieurs format de compression, et pouvoir travailler sur plusieurs formats différents tels que MPEG et DV, de façon transparente pour l'utilisateur.

## 6.3 Description du format

Le format MXF soumis à la SMPTE pour standardisation est une suite de documents couvrant les aspects spécifiques du format. Cette approche modulaire permet au format de respecter ses objectifs d'extensibilité. Les différentes parties du format sont illustrées à la figure 6.4. Pour une approche plus intuitive du MXF, le document **Engineering Guideline** est un bon point de départ.

En vue d'une normalisation rapide, le MXF fut développé en suivant les directives KLV de la SMPTE (clé, longueur, valeur, une méthode d'assemblage des données pour les acheminer sur les réseaux) et fait largement usage du dictionnaire et des autres registres de la SMPTE.

Le document **Format** décrit les briques fondamentales du format et la manière de les assembler afin de fournir les solutions spécifiques

Par exemple la structure : Key, Length, Value (KLV) de MXF. La structure KLV est utilisée pour coder chaque élément contenu dans le fichier MXF (essence et métadonnées), voir figure 6.5. Une clé unique de 16 octets identifie le type d'élément, la longueur indique le nombre d'octets du paquet KLV et la valeur est l'élément lui-même . En indiquant la longueur de chaque champ du fichier, les lecteurs peuvent ignorer les bits correspondant aux champs dont ils ne connaissent pas la clé (par exemple des fonctionnalités qu'ils n'offrent pas). Le système de clés permet au format MXF d'être évolutif. Si par exemple un nouveau type d'encodage pour la vidéo est développé, il suffit de lui assigner une nouvelle clé et les nouveaux lecteurs sauront lire ce type d'essence. De plus grâce à la clé et à la longueur des paquets, la recherche d'un élément au sein d'un fichier MXF est grandement facilitée. Si on recherche un piste audio au format MP3, on ne recherche les données que parmi les paquets qui ont la clé de ce type d'élément.

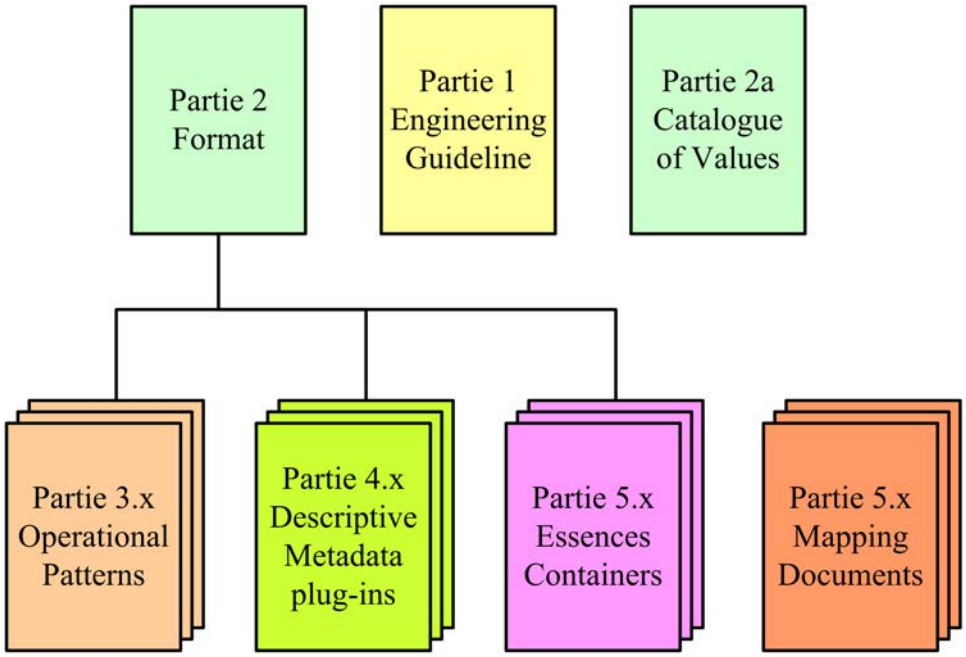


FIG. 6.4 – *Les différentes parties du MXF*


FIG. 6.5 – *Structure KLV d'un fichier MXF*

La structure d'un fichier MXF, illustrée à la figure 6.6, contient :

- des blocs d'en-tête et de fin qui établissent le début et la fin du fichier ;
- des métadonnées d'en-tête regroupant à la fois des métadonnées structurelles (nécessaires pour le traitement du fichier) et des métadonnées descriptives directement compréhensibles par l'homme tels que des notes de production ;
- des tables d'index fournissant des avantages opérationnels avancés qui ne sont toute fois pas obligatoires dans tous les cas. Cependant, lorsqu'on travaille avec des fichiers de grande taille sans tables d'index, la reconstruction des informations concernant les différents points d'accès dans le fichier est très longue et fastidieuse. Ce qui signifie qu'en cas de besoin d'accès aléatoire dans un fichier les tables d'index sont d'une aide très précieuse ;
- un container d'essences qui définit la localisation du contenu réel audio/visuel.

Les métadonnées d'en-tête de fichier de MXF apportent les principaux bénéfices du format. C'est à cet endroit que les métadonnées sont ajoutées et les paramètres

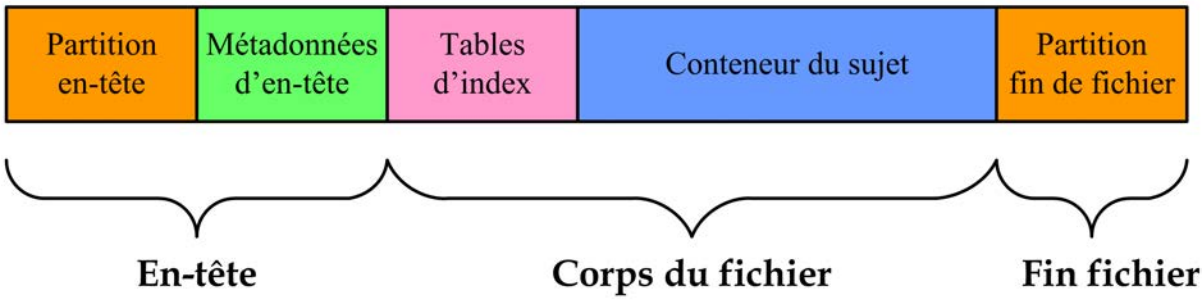


FIG. 6.6 – *Structure d'un fichier MXF*

temporels et de synchronisation sont ajoutés. La synchronisation et la description des essences sont contrôlées par trois packages :

- Material Package : la chronologie du fichier, indique la façon dont doit être lue le File Package.
- File Package : le contenu réel du fichier.
- Source Package : contient les données relatives au sujet : listes de montage précédentes, descriptions du contenu, ...).

On peut voir sur la figure 6.7 que chaque package du fichier peut avoir plusieurs pistes. Chacune d'elles représente les éléments composant une essence. Pour une vidéo on aura par exemple une piste vidéo, une bande son pour chaque canal audio et une piste de métadonnées. Ces pistes se décomposent à leur tour en SourceClips qui définissent comment créer le résultat souhaité avec ce fichier.

Lorsqu'il y a un seul SourceClip dans le Material Package correspondant à un File Package entier, on a un fichier MXF représentant une simple bande. Si le Material Package contient contient plusieurs SourceClip provenant de plusieurs File Package (regroupé dans le même fichier MXF) on a alors un fichier MXF représentant une liste de décision de montage (EDL[5]).

Les **Operational Patterns** ont été créés afin de répondre aux différentes, et parfois même divergentes, exigences d'applications opérationnelles spécifiques. Ces différents Operational Patterns sont repris dans la grille de la figure 6.8, où l'axe vertical représente la difficulté chronologique et où l'axe horizontal représente le nombre de packages à l'intérieur du fichier. Le premier Operational Pattern (1A), le plus simple, contient un seul clip, avec une piste vidéo et audio continue, ainsi que ses métadonnées. Des patterns plus complexes permettent de regrouper différents clips pouvant être traités selon une séquence bien définie.

MXF fournit aussi d'autres outils supplémentaires tels que des tables d'index,

[5] Liste fournissant des indications nécessaires à l'exécution du montage on line

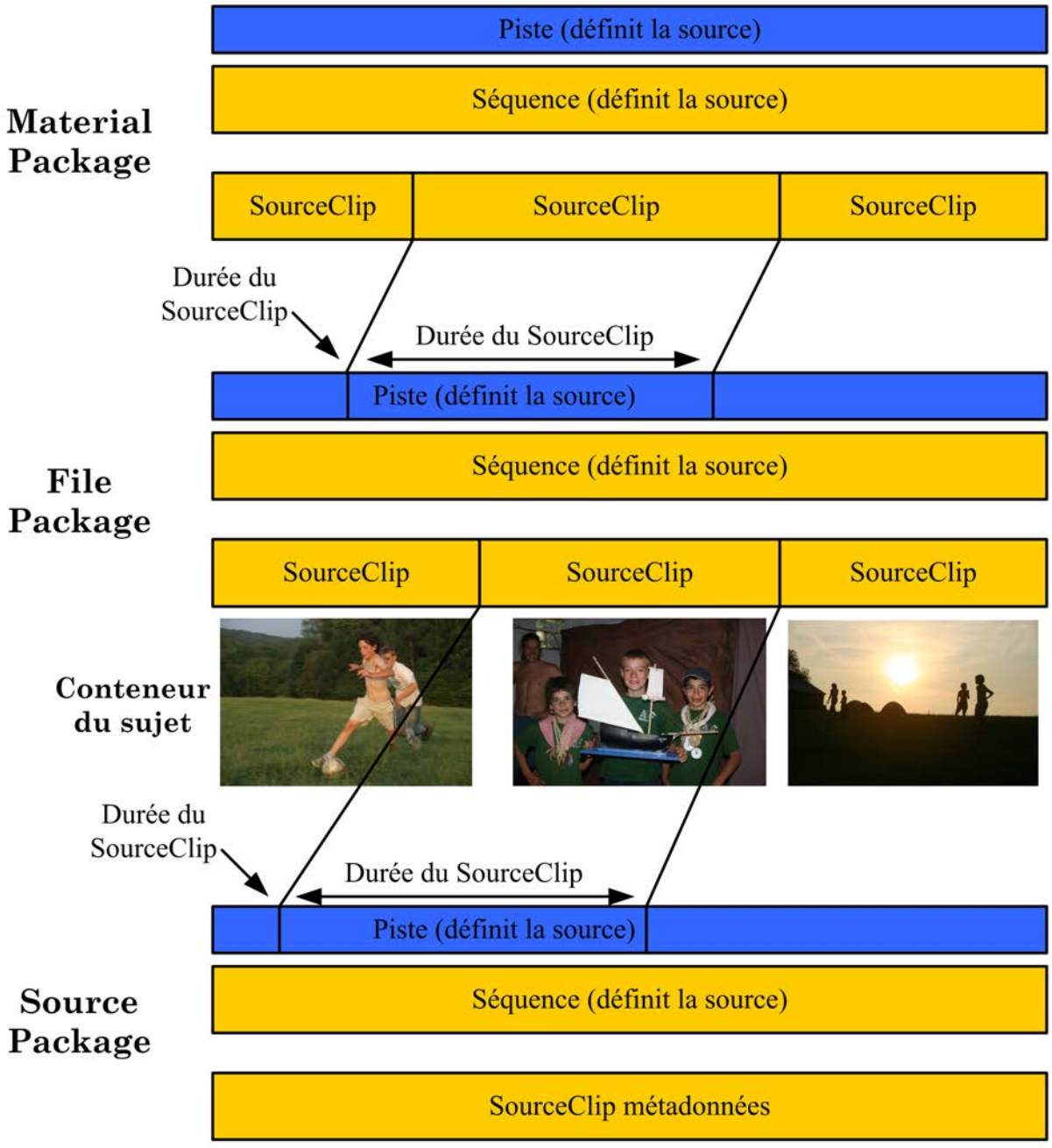


FIG. 6.7 – *Packages et pistes d'un fichier MXF*

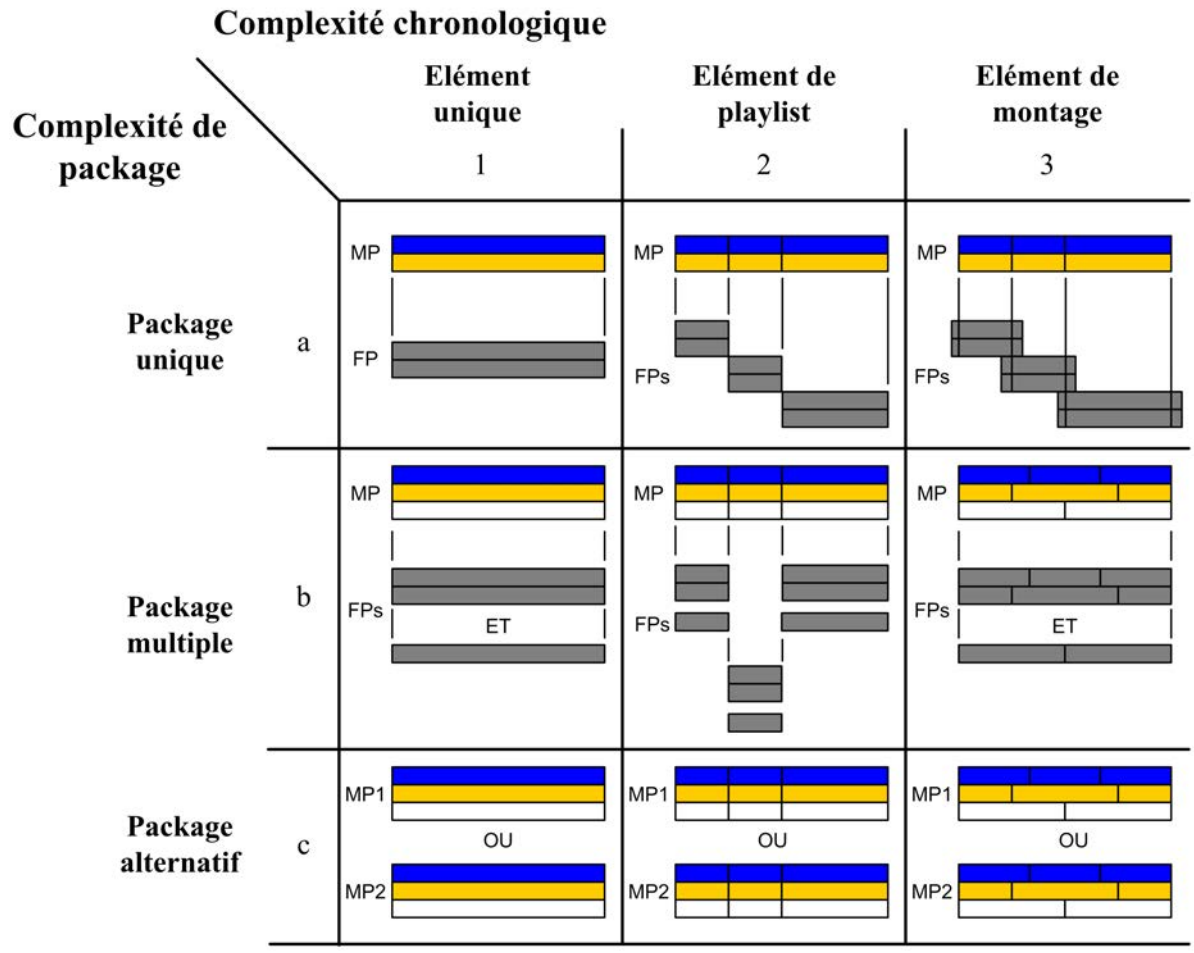


FIG. 6.8 – *Grille des Operational Patterns*

des outils de partitionnement pour le streaming ou le transfert de fichier, la prise en charge du système UMID et bien d'autres encore afin de faire de MXF le format de choix pour les applications multimédia.

La partie **Descriptive Metadata** fut probablement la partie qui a suscité le plus de discussions. C'est en fait la partie pour laquelle les utilisateurs et les constructeurs ont le moins d'expérience. Avant les métadonnées n'étaient rien de plus que des notes manuscrites, un script, des commentaires de tournage, des informations réparties sur différentes bases de données disparates. Il n'y avait pas de méthode cohérente pour préserver les métadonnées tout au long de la chaîne de production. Les métadonnées n'étaient donc pas utilisées de manière efficace.

C'est pourquoi MXF offre un ensemble d'outils pour améliorer la production des métadonnées ainsi que les rendent plus consistantes. Avec comme point de départ le dictionnaire de métadonnées de la SMPTE, MXF a implémenté "DMS-1". C'est en

fait une proposition d'un modèle de métadonnées générales, élaboré pour satisfaire la plupart des aspects de la chaîne de traitement. Ce modèle de métadonnées permet l'élaboration d'autres schémas pour des applications spécifiques. *MPEG-7* et TV-Anytime ont été identifiés comme schémas potentiels.

MXF est capable de contenir une quantité illimitée de métadonnées, mais au point de vue opérationnel ce n'est pas l'idéal. Dans beaucoup de cas, une fois créés, les métadonnées peuvent ensuite être utilisées pour plusieurs documents (comme c'est le cas pour les séries TV). Dans ce cas, la meilleure solution serait de conserver ces métadonnées dans une base de données centrale et de relier les documents à ces informations. De cette façon si on doit modifier les métadonnées, cela se fait pour tous les documents d'un coup, et on ne doit pas modifier les métadonnées dans tous les documents concernés. C'est pourquoi MXF, en combinaison avec UMID (Unique Material Identifier), permet de lier des document à des métadonnées distantes.

## 6.4 Conclusion

Le format MXF, dérivé du modèle de données AAF, est un format d'échange simple dont le but premier est de faciliter le transfert de contenus terminés, de programmes TV entiers ou de parties complètes, entre serveurs et vers des unités de sauvegarde à bande magnétique. MXF aide aussi à la migration des opérations de lecture et de systèmes de production plus simples, dans un environnement réseau standardisé.

Les format MXF et AAF sont particulièrement complémentaires. Alors que AAF s'intègre étroitement aux formats de fichier multimédia existants, et complète ceux-ci, MXF fait de même avec les formats de streaming existants. Chacun de ces formats peut être utilisé seul et ils ont tous deux une conception et des fonctionnalités optimisées pour leur propre sphère d'application. Mais dans le même temps, il ne dépendent pas l'un de l'autre. Par exemple, un système de radiodiffusion peut n'utiliser que MXF et une entreprise de postproduction n'utiliser que AAF. Cependant si un radiodiffuseur s'occupe aussi de postproduction, il devra utiliser les deux.

Alors que AAF et MXF sont complémentaires, ils présentent pas mal de différences. Par exemple, AAF peut transporter des références à des informations extérieures (gardées dans d'autres endroits) qui seront utilisée en postproduction, alors que MXF contient toutes ses métadonnées au sein des fichiers.

MXF a été réalisé sur base des besoins des utilisateurs et a donc un fort potentiel commercial. Pour son développement, des constructeurs, qui sont concurrents en temps normal, ont travaillé ensemble afin de fournir rapidement une solution

industrielle ouverte pour l'échange de fichier. Après cinq ans d'une collaboration intensive, MXF fut standardisé en 2004 par la SMPTE. MXF est maintenant disponible en tant que composant à intégrer dans les produits conçus par les fabricants de matériel audiovisuel et les concepteurs de software. Grâce à ces produits, les utilisateurs peuvent mieux gérer leurs médias et se concentrer sur leurs activités de production et de créativité.

Grâce à son interopérabilité et son caractère de standard ouvert, les radiodiffuseurs et les utilisateurs professionnels considèrent MXF comme une base fondamentale pour les nouvelles installations réseaux.

Afin de réaliser ce chapitre nous nous sommes inspirés des documents suivants : [Dev04], [Dev02], [MPGF98], [MXF] [Var03].

CHAPITRE 7

# MPEG-7

## 7.1 Introduction

Il existe actuellement de nombreux moteurs de recherche textuelle sur l'Internet, dont les taux très élevés d'utilisation attestent de l'intérêt qu'ils suscitent. Cependant, à la différence du texte, qui contient en quelque sorte sa propre description, avec les mots comme unités sémantiques, l'image se compose de pixels sans signification intrinsèque. La recherche doit donc s'effectuer à partir de descriptions externes générées au préalable lors de l'indexation des documents. Certains moteurs proposent aussi des recherches d'images basé sur des mots-clefs. Comme le montre la figure 7.1, la recherche d'une image d'un "dik dik"[1] peut donner des résultats assez surprenants.

De plus, pour tirer pleinement partie des possibilités d'accès direct aux images et aux sons offertes par le numérique, ces descriptions doivent prendre en compte la dimension temporelle des objets audiovisuels (afin par exemple de ne pas avoir à visionner l'intégralité du "Père Noël est une ordure" pour retrouver la scène particulière où "Zézette épouse X").

Le nombre de données multimédia augmentant de manière très importante, il est donc vite devenu nécessaire de créer un outil permettant d'accéder au contenu

[1] Une petite gazelle

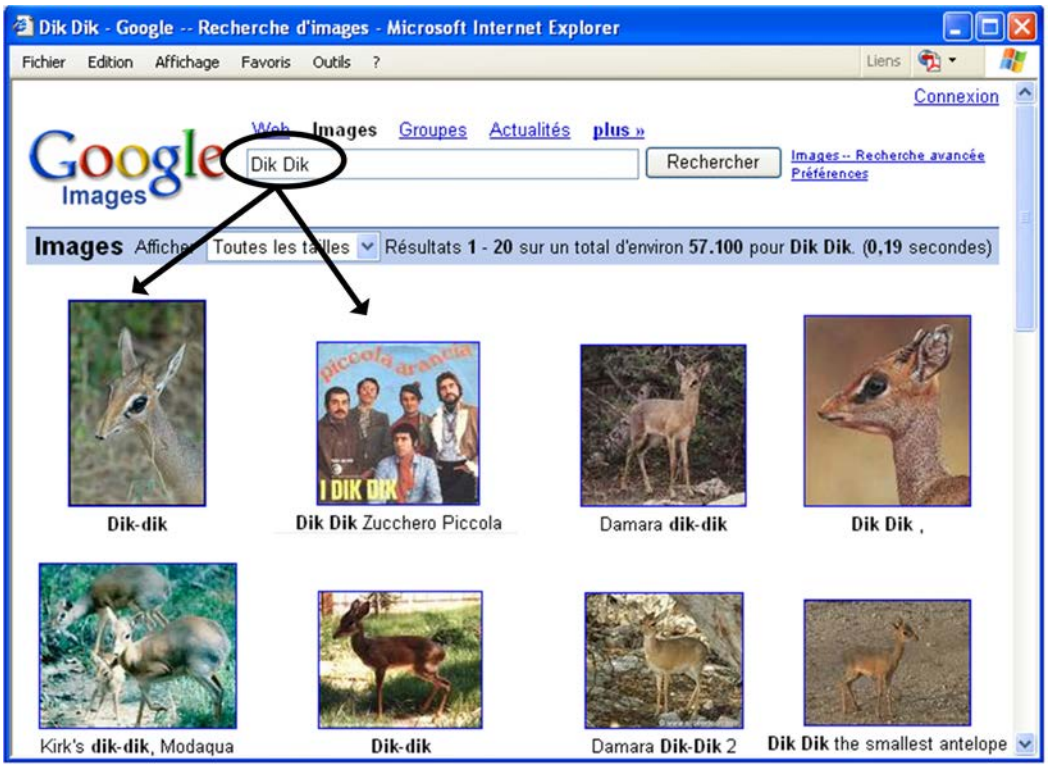


FIG. 7.1 – *Recherche d'image avec le moteur Google*

d'images fixes, de vidéos, graphiques, d'objets de synthèse, etc. Le *MPEG-7* aussi connu sous le nom de « Multimédia Content Description Interface » est un standard offrant la possibilité de décrire le contenu d'un fichier multimédia avec un certain niveau d'interprétation des informations de ce fichier.

Le *MPEG-7* n'est pas dédié à un média particulier, mais au contraire permet de standardiser un nouveau moyen de recherche multimédia et ce en visant le plus grand nombre d'applications possibles. On peut évidemment établir une description *MPEG-7* d'un fichier *MPEG-1* ou *MPEG-2*, mais on peut faire de même avec un film analogique *PAL/SECAM*[2] ou encore un journal papier. Il s'agit uniquement d'un standard de représentation du contenu des documents. Cette description sera liée au contenu multimédia lui-même afin de permettre à l'utilisateur, via un équipement spécifique ou un logiciel, une recherche pertinente et rapide d'un média.

*MPEG-7* fournit donc un grand nombre de fonctions descriptives standardisées d'un large éventail d'informations multimédia. Le choix d'un ou plusieurs descripteurs se fera donc en fonction du contexte de l'application visée. Cela implique qu'il est possible que le même matériel soit décrit de manières différentes. En effet *MPEG-7* supporte plusieurs niveaux d'abstraction différents, allant des carac-

[2]Norme de codage européenne pour la télévision hertzienne au format 4/3

téristiques de bas niveau comme la forme, la texture, la couleur, le mouvement de la caméra ou encore le timbre de la musique à des informations sémantiques de haut niveau comme le genre de contenu ou des événements. Le niveau d'abstraction influence grandement la manière dont l'extraction des informations est faite. En effet, plus le niveau d'abstraction est bas, plus il sera possible d'extraire automatiquement ces informations. Dans le cas contraire, une intervention humaine sera souvent nécessaire.

A coté de la description du contenu, *MPEG-7* permet également d'inclure d'autres informations tels que :

- le format (la compression utilisée (*JPG*), la taille, ...). Ces informations peuvent aider à déterminer si le média peut être lu par l'utilisateur ;
- les conditions d'accès au support. Cela pourrait inclure des informations concernant le copyright et le prix ;
- la classification (appréciation parentale, catégorisation, ...) ;
- des liens vers d'autres matériaux intéressants.

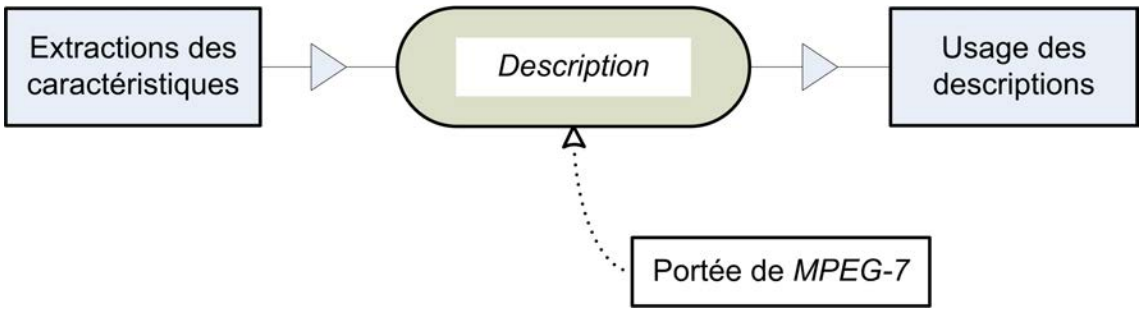


FIG. 7.2 – *Portée de MPEG-7*

La figure 7.2 montre le traitement de la chaîne *MPEG-7*. Cette chaîne comporte l'analyse du contenu pour l'extraction des caractéristiques, la description elle-même et les moteurs de recherches pour la consommation. Pour exploiter pleinement les possibilités de *MPEG-7* , il est nécessaire d'utiliser des logiciels d'extraction automatique des caractéristiques. Cependant comme dit plus haut, cela n'est pas toujours possible. On constate aussi que les algorithmes d'extraction ne sont pas standardisés par *MPEG-7*. La raison principale de cette non-prise en charge est que leur standardisation n'est pas nécessaire pour assurer l'interopérabilité de *MPEG-7*. De plus ce choix permet de stimuler la concurrence et donc l'innovation dans ce domaine. Il en va de même pour le bout de la chaîne, les applications (moteurs de recherche, filtres, ...) utilisant les descriptions ne sont pas spécifiées par la norme. La standardisation s'applique donc uniquement à la *syntaxe* des descriptions.

## 7.2 Principe de MPEG-7

*MPEG-7* est donc une interface de description des données de contenu multimédia. Ces données appelées aussi descripteurs (D[3]) sont structurées en schémas de description (DS[4]) selon un langage de définition de description (DDL[5]).

**D** Les descripteurs sont une représentation primitive du média. Ils définissent la syntaxe et la sémantique de chaque caractéristique. Il s'agit d'une description bas niveau tenant compte de la couleur, de la texture, du mouvement, de la localisation et du temps, etc. Ces informations sont généralement extraites automatiquement par les applications.

**DS** Les schémas de description spécifient la structure et la sémantique des relations entre les composants. Ces schémas peuvent être aussi bien des descripteurs que d'autre schémas de description.

**DDL** Le langage de définition de description fondé sur *XML* Schema permet la création de nouveaux schémas de description, de nouveaux descripteurs et également de modifier les existants.

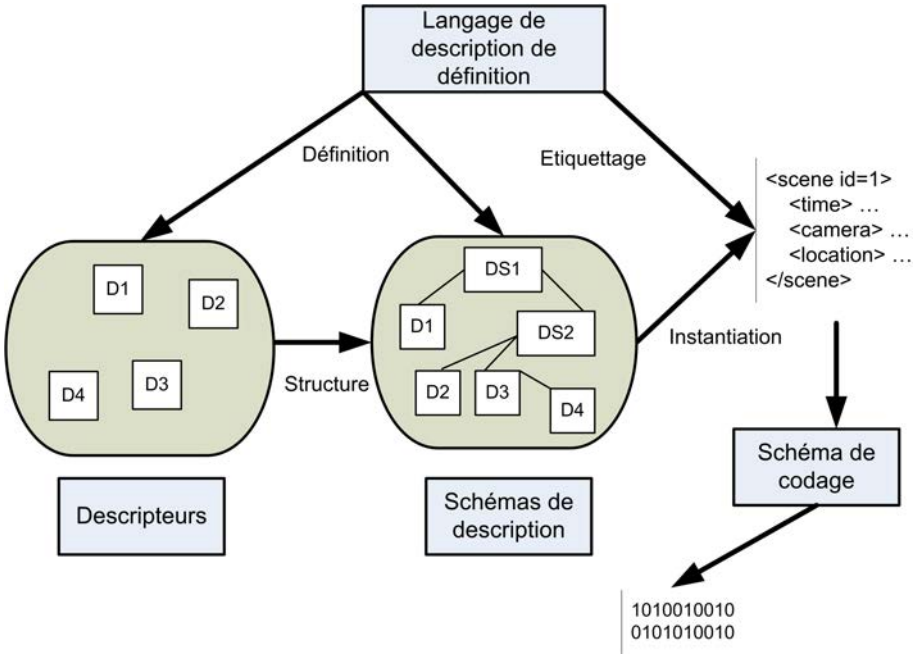


FIG. 7.3 – *Eléments principaux de MPEG-7*

[3]D : Descriptor
[4]DS : Description Scheme
[5]DDL : Description Definition Language

De plus *MPEG-7* dispose d'un ensemble de schémas de codage permettant de réaliser des fonctions comme la compression efficace, la correction d'erreur, l'accès direct, le stockage, etc. A l'heure actuelle le codage retenu est appelé *BiM*[6].

La figure 7.3 montre les différentes relations entre les différents éléments principaux de *MPEG-7*. La DDL permet de définir les outils de description de *MPEG-7* (D et DS) et fournit les moyens de structurer les descripteurs en schémas de description. Les outils de description sont quant à eux instantiés en tant que description au format *XML* grâce à la DDL (basé lui-même sur *XML* Schema). Le format binaire de description est obtenu au moyen du *BiM*.

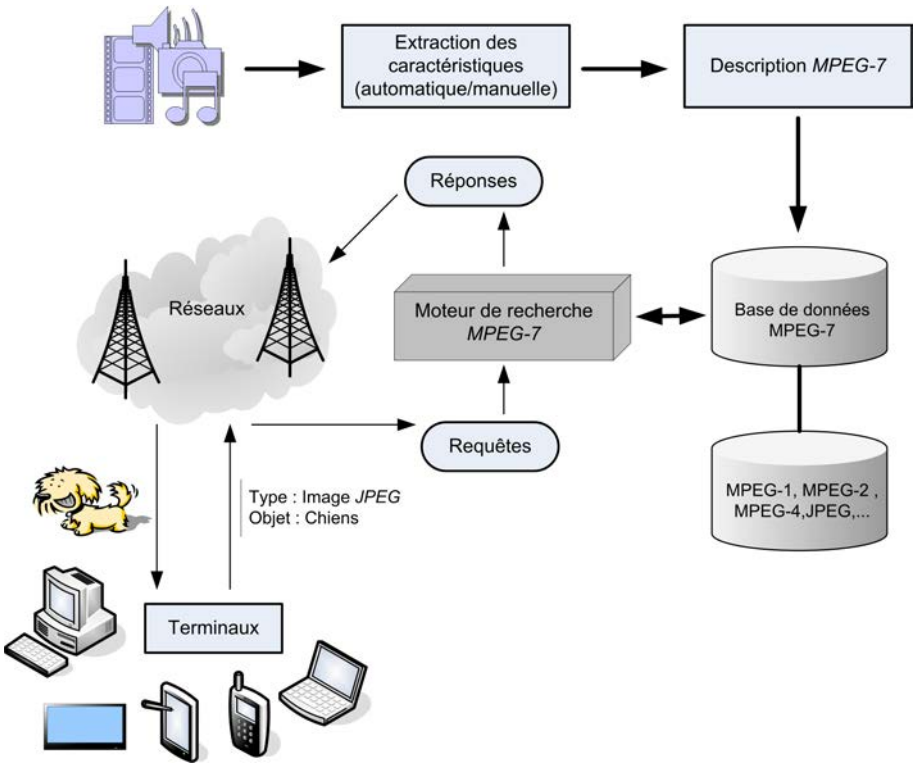


FIG. 7.4 – *Exemple d'application utilisant MPEG-7*

La figure 7.4 montre une application possible client/serveur utilisant la technologie *MPEG-7*. Comme nous l'avons déjà dit plus haut, les caractéristiques des contenus multimédia sont extraites soit manuellement soit via des logiciels automatisés. Les caractéristiques extraites sont codées en format *MPEG-7* et stockées dans une base de données. Pour chaque description, on retrouve une référence physique vers le contenu multimédia. Lorsqu'un utilisateur envoie une requête (par exemple, des images contenant des chiens), le moteur de recherche va retrouver toutes les descriptions représentant une image avec chien. Une fois les descriptions filtrées,

[6]BIM : Binary Format for Metadata, défini dans [ISO04a]

l'application transmet via les adresses physiques contenues dans les descriptions valables les images contenant des chiens.

## 7.3 Principales fonctionnalités

*MPEG-7* fournit les fonctionnalités principales[7] suivante :

**DDL** Comme nous l'avons déjà vu, le langage de définition de description fondé sur *XML* Schema permet la création de nouveaux schémas de description, de nouveaux descripteurs et également de modifier les existants. Cependant, vu que le langage *XML* Schema n'est pas adapté pour la description de documents audiovisuels, certaines extensions ont dû être rajoutées.

***MPEG-7* Visuel** Ce nom regroupe l'ensemble des structures basiques et descripteurs qui couvrent les caractéristiques visuelles de bases comme la couleur, la texture, la forme, le mouvement, etc.

***MPEG-7* Audio** *MPEG-7* Audio définit conjointement avec les schémas de description multimédia (voir point suivant) un standard de description de document audio. Il regroupe un ensemble des descripteurs de bas niveau couvrant un large champ d'application (par exemple les caractéristiques spectrales, paramétriques ou encore temporelles d'un signal). A coté des descripteurs de bas niveau, *MPEG-7* Audio fournit aussi un ensemble de descripteurs de haut niveau utilisés pour des applications plus spécifiques comme la reconnaissance vocale, la description du contenu sonore, la description du timbre, etc.

**MDS** Les schémas de description multimédia regroupent un ensemble d'outils de description (aussi bien des descripteurs que des schémas de description) permettant une description orientée multimédia ou plus générique à chaque type de média (tel que des caractéristiques de temps, des commentaires, ...).

### 7.3.1 Le DDL

Le *DDL* représente le coeur du standard *MPEG-7*. Il définit les règles sémantiques pour formuler et combiner les descripteurs et schémas de description. Le *DDL* n'est pas un langage de modélisation unifié comme l'*UML*[8] mais bien un langage de schéma permettant de représenter les résultats de la "modélisation" d'une

[7] La norme *MPEG-7* contient d'autre fonctionnalités comme *MPEG-7* Conformance, Profilers, Schema Definition,...que nous ne détaillerons pas dans ce mémoire. De plus amples informations sont disponible dans [ISO04a]

[8] UML : Unified Modeling Language

donnée audiovisuelle. Conformément à la norme *MPEG-7*, *DDL* permet d'exprimer les relations spatiales, temporelles, structurelles, ou encore conceptuelles entre les différents éléments d'un DS, et entre DSs.

Le *DDL* était à l'origine basé sur le langage *XML*[9]. Le groupe de travail *MPEG-7* s'est ensuite tourné vers le langage *XML* Schema qui lui-même est un sous-ensemble du *XML*. En effet ce dernier répondait à la majorité des exigences de *MPEG-7*. Cependant vu que le *XML* Schema n'était pas spécifiquement destiné au contenu multimédia, il a été nécessaire d'ajouter certaines extensions spécifiques au *MPEG-7*. Le *DDL* comprend donc 3 composantes :

- les composants structurels de *XML* Schema[10] ;
- les composants dits de type de données de *XML* Schema[11] ;
- les extensions *MPEG-7* au *XML* Schema.

**A - Le *XML***

Le *XML* est un langage de méta-description permettant de créer selon ses propres besoins des balises. L'ensemble de ces balises forme la structure du document *XML* et permet de donner sens à l'information contenue dans le document. Le *XML* est donc un format de représentation des données par balises défini par le *W3C*[12]. Sous-ensemble du *SGML*[13], il possède la majorité de ses caractéristiques tout en étant plus facile à manipuler. Il est souvent qualifié de métalangage, c'est-à-dire un format de modélisation de documents grâce à des balises qui décrivent la structure et le contenu du document et non sa présentation.

Les balises XML ne sont pas prédéfinies, il est libre à chacun de créer ses propres balises mais celles-ci doivent toutefois respecter certaines règles de construction et de structuration.

```
<objet attribut = "1">
Texte, Balise XML
</objet>
```

FIG. 7.5 – *Jeu de balises* XML *ouvrante - fermante*

Une balise non vide XML comporte en réalité un jeu de balises ouvrante et fermante. La balise fermante délimitée par le caractère "/" doit porter le même nom

[9]XML : eXtensible Markup Language
[10]Spécification disponible à l'url suivante : http://www.w3.org/TR/xmlschema11-1/
[11]Spécification disponible à l'url suivante : http://www.w3.org/TR/xmlschema11-2/
[12]W3C : World Wide Web Consortium
[13]Standard Generalized Markup Language : Langage international de documentation

que l'ouvrante. Toute donnée est ainsi encapsulée entre une balise ouvrante et fermante. Une donnée peut être aussi bien du texte qu'un autre jeu de balises XML. L'encapsulation de balises forme l'arbre des éléments c'est à dire une hiérarchie de balises comportant éventuellement des attributs.


FIG. 7.6 – *Balise vide*

Lorsqu'un élément de la structure n'a pas besoin de contenu, il est possible d'utiliser une balise vide remplaçant le couple de balises ouvrante/fermante. Une balise vide peut également comporter des attributs.

Un document XML bien construit est un document suivant les règles de syntaxes et de structure du langage XML dont voici les plus importantes :

**Structuration des balises**

Chaque balise ouvrante doit posséder sa balise de fin à l'exception des balises vides qui sont à la fois ouvrantes et fermantes. Cette règle garantit la cohérence de la structuration de l'information qui est primordiale lors des traitements des documents XML.

**Imbrication des balises**

Une balise encapsulée dans une autre doit être obligatoirement fermée avant que la balise englobante ne se ferme. Cela permet d'assurer la hiérarchisation de la structure du document.

**Élément racine**

Un document XML comporte un seul élément racine représentant le sommet de l'arborescence du document XML.

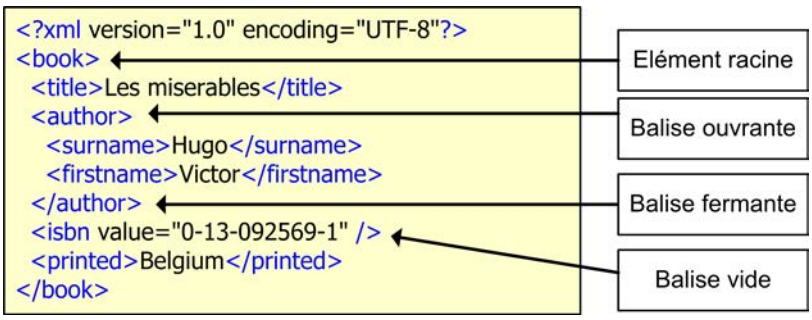


FIG. 7.7 – *Exemple de document XML*

**B - les composants structurels de *XML* Schema**

Les composants structurels de *XML* Schema permettent de décrire la structure et le contenu des documents *XML*. Un schéma *XML* est construit par assemblage de différents composants rassemblés en différentes catégories.

Les composants primaires qui regroupent :

- les définitions de type simple ;
- les définitions de type complexe ;
- les déclarations d'attributs ;
- les déclarations d'éléments.

Dans *XML* Schema, il y a un une différence de base entre les éléments de type complexe qui peuvent contenir des sous-éléments et être qualifiés par des attributs et ceux de type simple qui ne peuvent ni contenir des sous-éléments ni être qualifiés par des attributs. Il y a également une distinction majeure entre les définitions qui servent à créer de nouveaux types (qu'ils soient de type simple ou complexe) et les déclarations qui servent à spécifier les noms et les types des éléments et attributs qui pourront être utilisés dans les instances de documents

Les composants secondaires qui se composent :

- des définitions de groupes d'attributs ;
- des définitions de contraintes d'identifiants ;
- des déclarations de notation ;
- des définitions de modèles de groupes d'éléments.

Les composants d'aides regroupant :

- les annotations ;
- les modèles de groupes ;
- les Wildcards ;
- les particules.

Les composants d'aides "helpers" définissent des petites parties d'autres composants qui dépendent du contexte.

A titre d'illustration, les figures 7.9 et 7.8 montre un exemple de schéma *XML* pour la gestion de bon de commande ainsi qu'un document *XML* instantiant ce schéma.

```
<?xml version="1.0" encoding="UTF-8"?>

 <bonCommande dateCommande="20-12-1980">
  <client>
   <nom> Michel Feron </nom>
   <rue> Rue Félix Wodon 10 </rue>
   <ville> Namur </ville>
   <cp> 5000 </cp>
   <telephone> 04 77 38 69 50 </telephone>
  </client>
  <vendeur>
   <nom> Paul Delcorde </nom>
   <rue> Rue Henri Lemaitre 15 </rue>
   <ville> Namur </ville>
   <cp> 5000 </cp>
   <telephone> 04 76 32 45 89 </telephone>
  </vendeur>
  <commande>
   <produit>
    <nomProduit> Nakio 3211 </nomProduit>
    <quantite> 1 </quantite>
    <prixEuro> 50,10 </prixEuro>
   </produit>
   <produit>
    <nomProduit> Chargeur Nakio 3211 </nomProduit>
    <quantite> 2 </quantite>
    <prixEuro> 20,00 </prixEuro>
   </produit>
   <produit>
    <nomProduit> Batterie Nakio 3211 </nomProduit>
    <quantite> 1 </quantite>
    <prixEuro> 40,30 </prixEuro>
   </produit>
  </commande>
 </bonCommande>
```

FIG. 7.8 – *Document XML*

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
 targetNamespace="http://www.w3.org/2000/10/XMLSchema" >

<xsd:element name = "bonCommande" type="bonCommandeType"/>
<xsd:element name = "commentaire" type="xsd:string "/>

<xsd:simpleType name="numeroTelephone">
 <xsd:list itemType="xsd:unsignedByte" />
</xsd:simpleType>

<xsd:complexType name="BEAdresse" >
 <xsd:sequence>
  <xsd:element name="nom"          type="xsd:string"/>
  <xsd:element name="rue"          type="xsd:string"/>
  <xsd:element name="ville"        type="xsd:string"/>
  <xsd:element name="cp"           type="xsd:decimal"/>
  <xsd:element name="telephone"    type="xsd:numeroTelephone"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="bonCommandeType" >
  <xsd:sequence>
    <xsd:element name="client"       type="BEAdresse"/>
    <xsd:element name="vendeur"      type="BEAdresse"/>
    <xsd:element name="commande"     type="listeProduit"/>
    <xsd:element name="comBon"       type="commentaire"/>
  </xsd:sequence>
  <xsd:attribute name="dateCommande" type="xsd:date"/>
</xsd:complexType>

<xsd:complexType name="listeProduit">
 <xsd:sequence>
  <xsd:element name="produit" minOccurs="0" maxOccurs="100">
   <xsd:complexType>
    <xsd:sequence>
     <xsd:element name="nomProduit" type="xsd:string"/>
     <xsd:element name="quantite">
      <xsd:simpleType>
       <xsd:restriction base="xsd:positiveInteger">
        <xsd:maxExclusive value="100"/>
       </xsd:restriction>
      </xsd:simpleType>
     </xsd:element>
     <xsd:element name="prixEuro"  type="xsd:decimal"/>
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
 </xsd:sequence>
</xsd:complexType>

</xsd>
```

FIG. 7.9 – *Document XML Schema pour la gestion de bon de commande*

Comme tout document *XML*, un schéma *XML* commence par un prologue, et a un élément racine. L'élément racine est l'élément <xsd : schema>.

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
 targetNamespace="http://www.w3.org/2000/10/XMLSchema" >

<!-- déclarations d'éléments, d'attributs et de types ici -->

</xsd>
```

FIG. 7.10 – *Prologue d'un schéma* XML

La déclaration de l'attribut <xmlns> permet d'associer le fichier *XML* à un espace de nommage. Les namespaces, ou espaces de nommage, sont un mécanisme destiné à lever les ambiguïtés éventuelles des intitulés de balise. Par exemple <titre> peut aussi bien désigner le titre d'un ouvrage que celui d'une personne. Pour lever ces ambiguïtés, le mécanisme des namespaces consiste à utiliser des préfixes, par ex. <perso : titre> et <biblio : titre>. Ces préfixes doivent être déclarés comme associés à une URL qui peut être fictive, mais qui le plus souvent fera référence à l'organisme garant du vocabulaire en question.

L'attribut <xmlns> fait référence ici à l'espace de noms par défaut défini par *W3C XML* Schema. Pour être conforme, un élément appartenant au vocabulaire *XML* Schema devra être préfixé par <xsd :>. L'attribut <targetNameSpace> indique quant à lui un espace de noms cibles pour tout élément étranger au vocabulaire de *XML* Schema.

Un élément, dans un schéma, se déclare avec la balise <xsd : element>.

```
<xsd:element name = "bonCommande" type="bonCommandeType"/>
<xsd:element name = "commentaire" type="xsd:string "/>
```

FIG. 7.11 – *Déclaration d'éléments*

Ce schéma déclare deux éléments : un élément <bonCommande> et un élément <commentaire>. Chaque élément est typé c'est-à-dire qu'il doit respecter un certain format de données. L'élément contact est ainsi du type <bonCommandeType>, qui est un type complexe défini par l'utilisateur, qui peut contenir des sous-éléments et des attributs. L'élément <commentaire> quant à lui est du type <xsd : string> qui est un type simple prédéfini de *XML* Schema.

La définition suivante est celle du type complexe <BEAdresse> :

```
<xsd:complexType name="BEAdresse" >
  <xsd:sequence>
    <xsd:element name="nom"          type="xsd:string"/>
    <xsd:element name="rue"          type="xsd:string"/>
    <xsd:element name="ville"        type="xsd:string"/>
    <xsd:element name="cp"           type="xsd:decimal"/>
    <xsd:element name="telephone"    type="xsd:numeroTelephone"/>
  </xsd:sequence>
</xsd:complexType>
```

FIG. 7.12 – *Déclaration d'un type complexe avec séquence*

Les types complexes sont créés en utilisant la balise <complexType> composée dans la plupart des cas d'une série de déclarations d'éléments et d'attributs et de références d'éléments. Les déclarations ne sont pas elles-mêmes des types mais plutôt des associations faites entre un nom et des contraintes qui régissent les règles d'apparition de l'élément dans une instance de document correspondant au schéma. Les éléments sont déclarés en utilisant la balise <element>. Par exemple, l'élément <BEAdresse> est défini comme étant de type complexe et, à l'intérieur de sa définition, nous remarquons la présence de cinq déclarations d'éléments. La balise <sequence> sert à définir un type complexe contenant des suites d'éléments. Ces éléments doivent être appelés <nom>, <rue>, <ville>, <cp> et <telephone> conformément à ce qui est spécifié par les différentes valeurs de l'attribut name des déclarations de la figure 7.12. Les instances de <bEAdresse> devront impérativement contenir ces 5 éléments en suivant le même ordre que la déclaration.

```
<xsd:complexType name="bonCommandeType" >
  <xsd:sequence>
    <xsd:element name="client"     type="BEAdresse"/>
    <xsd:element name="vendeur"    type="BEAdresse"/>
          ....
  </xsd:sequence>
  <xsd:attribute name="dateCommande" type="xsd:date"/>
</xsd:complexType>
```

FIG. 7.13 – *Définition d'un type complexe avec séquence*

Dans la définition de <bonCommandeType> à la figure 7.13, les déclarations des deux éléments <vendeur> et <client> s'appuient sur le même type complexe <BEAdresse>. La conséquence de cette définition est que tout élément d'une instance de document dont le type déclaré est <bonCommandeType> doit contenir les

éléments <client> et <vendeur>, chacun contenant à leur tour les 5 sous-éléments (nom, rue, ville, cp et telephone) déclarés dans la définition de <BEAdresse>. Cette définition contient aussi la déclaration de l'attribut <dateCommande> qui utilise un type simple. En fait, toutes les déclarations d'attributs ne peuvent que référencer des types simples car, contrairement aux éléments, les attributs ne peuvent pas contenir eux-mêmes d'autres attributs ou éléments. Ces attributs sont déclarés en utilisant la balise <attribute>.

Prenons maitenant la définition du type simple <numeroTelephone> :

```
<xsd:simpleType name="numeroTelephone" >
 <xsd:restriction base="xsd:string">
    <xsd:maxLength value="10"/>
  </xsd:restriction>
</xsd:simpleType>
```

FIG. 7.14 – *Définition d'un type simple*

Les types simples sont créés en utilisant la balise <simpleType>. L'élément <restriction> et son attribut <base> indiquent que ce type de données est dérivé par restriction (c'est-à-dire en donnant une contrainte sur les données acceptées) du type de données <string> appartenant à l'espace de noms *W3C XML* Schema. L'élément <maxLength> est appelé une *facette* et définit cette restriction comme étant le fait que la taille maximale est de 10, l'unité étant fonction du type de données est ici le caractère. L'exemple pris ici n'est évidemment pas la bonne solution pour encoder un numéro de téléphone car il permettrait l'insertion de : <telephone> AZ12DE14 </telephone>.

La figure 7.15 montre une manière plus correcte de représenter le type simple <numeroTelephone> :

```
<xsd:simpleType name="numeroTelephone">
  <xsd:list itemType="xsd:unsignedByte" />
</xsd:simpleType>
```

FIG. 7.15 – *Le type liste*

Les types listes sont des suites de types simples (ou atomiques). *XML* Schema possède trois types de listes intégrés : NMTOKENS, ENTITIES et IDREFS. Il est également possible de créer une liste personnalisée, par dérivation de types existants comme le montre l'exemple ci-dessus. Un numéro de téléphone sera repréenté

ici comme une liste d'octets non signés. Un élément conforme à cette déclaration serait <telephone>02 20 21 17 43</telephone>.

Les types atomiques et listes permettent à un élément ou une valeur d'attribut d'être composé d'une ou plusieurs instances d'un même type atomique. Par contraste, un type <union> permet à un élément ou une valeur d'attribut d'être composé d'une ou plusieurs instances d'un type résultant de l'union de plusieurs types atomiques ou listes.

```
<xs:simpleType name="contact">
  <xs:union memberTypes="xsd:string numeroTelephone" />
</xs:simpleType>
```

FIG. 7.16 – *Le type union*

Ici, le type simple <contact> est soit un string, soit le type <telephone>. Des instances valides de cet élément pourraient être : <contact>Michel Feron</contact> ou <contact>04 77 38 69 50</contact>.

```
<xsd:complexType name="listeProduit">
 <xsd:sequence>
  <xsd:element name="produit" minOccurs="0" maxOccurs="100">
   <xsd:complexType>
    <xsd:sequence>
     <xsd:element name="nomProduit" type="xsd:string"/>
     <xsd:element name="quantite">
      <xsd:simpleType>
       <xsd:restriction base="xsd:positiveInteger">
        <xsd:maxExclusive value="100"/>
       </xsd:restriction>
      </xsd:simpleType>
     </xsd:element>
     <xsd:element name="prixEuro"  type="xsd:decimal"/>
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
 </xsd:sequence>
</xsd:complexType>
```

FIG. 7.17 – *Les facettes XML Schema*

La figure 7.17 représente la définition du type complexe <listeProduit>. Cette liste correspond à une séquence d'éléments <produit>. Les attributs <minOccurs> et <maxOccurs> liés à l'élément <produit> correspondent, respectivement, au nombre minimum et maximum d'occurrences de l'élément. <produit> est lui même composé

de 3 éléments (nomProduit, quantite, prixEuro). La *facette* restrictive <maxExclusive> dans la définiton de l'élément <quantite> permet la commande de minimum 1 et maximum 99 mêmes produits.

Il existe d'autres *facettes* qui permettent de :

- fixer la longueur d'un type simple, restreindre sa longueur maximale et minimale ;
- énumérer toutes les valeurs possibles d'un type ;
- gérer des expressions régulières ;
- fixer la valeur minimale ou maximale d'un type ;
- fixer la précision du type ;
- ...

### C - Extensions *MPEG-7* de *XML* Schema

Comme nous l'avons déjà dit plus haut, le *XML* Schema n'est pas spécifiquement adapté pour décrire la structure de documents multimédia. Certaines extensions ont donc du être rajoutées pour mieux répondre aux caractéristiques de ce type de media telles que la prise en charge des types de données tableaux et matrices ainsi que des types de données temporelles[14]. L'exemple ci dessous montre la défintion d'une matrice en *DDL* ainsi qu'une représentation *XML* de cette matrice.

```
<simpleType name="MatriceEntier3x4Type" base="integer" derivedBy="list">
        <mpeg7:dimension value="3 4" />
</simpleType>

<element name='MatriceEntier3x4' type='MatriceEntier3x4Type'/>
```

```
<IntegerMatrix3x4>
      5 8 9 4
      6 7 8 2
      7 1 3 5
</IntegerMatrix3x4>
```

FIG. 7.18 – *Extensions MPEG7 XML Schema : les matrices*

[14]Pour plus de plus amples détails voir ISO/IEC 15938-2

### 7.3.2 Le MPEG-7 Visual

Les outils de description visuelles de *MPEG-7* sont des structures basiques et des descripteurs couvrant les caractéristiques visuelles de base telles que la couleur, la texture, la forme, le mouvement et la reconnaissance de visage. Chaque catégorie regroupe un ensemble de descripteurs simples ou plus sophistiqués.

**A - Les structures de bases**

*MPEG-7* Visual comporte 5 structures de bases : les grilles, les structures temporelles, les points de vue multiples, les coordonnées 2D et les interpolations temporelles

**Les Grilles (Grid layout)**

Cette structure de base permet d'éclater une image en un ensemble de zones rectangulaires de même taille. Sur chaque rectangle on peut ainsi appliquer des descripteurs communs ou individuels. Ces descripteurs peuvent être par exemple la couleur ou la texture.

```
<element name="GridLayout">
  <complexType content="empty">
    <attribute name="PartNumberH" datatype="positiveInteger"/>
    <attribute name="PartNumberV" datatype="positiveInteger"/>
  </complexType>
</element
```

FIG. 7.19 – *Syntaxes DDL d'un Grid Layout*

La figure 7.19 montre la définition *DDL* d'un élément <GridLayout>. Les attributs PartNumberH et PartNumberV permettent de définir en combien de régions l'image doit être délimitée.

**Les descripteurs de structures temporelles (Time Series)**

Ces descripteurs permettent de définir des séries de descripteurs temporels dans une séquence vidéo et offrent des fonctionnalités permettant de faire correspondre une image à une image tirée d'une séquence vidéo ou des images tirés de séquences vidéo entre elles. Deux types de séries temporelles sont disponibles : les *régulières* et les *irrégulières*. Pour les régulières, les descripteurs localisent à intervalles réguliers pendant une période de temps définie. Quant aux irrégulières, les descripteurs localisent avec des intervalles de temps irréguliers.

**Les points de vue multiples (2D-3D Multiple View)**

Le descripteur 2D/3D permet de représenter une caractéristique visuelle d'un objet en trois dimensions sous différents angles par la combinaison de différents descripteurs 2D. Tous les descripteurs visuels 2D peuvent être utilisés, qu'ils soient de forme, de couleur ou encore de texture.

**Les coordonnées 2D (Spatial 2D Coordinates)**

Cette structure de base permet de définir un système de coordonnées spatiales en deux dimensions pouvant être utilisé dans d'autres Ds et DSs si cela est nécessaire. Le système de coordonnées est obtenu par une correspondance entre une image et le système de coordonnées. L'avantage de cette technique réside dans le fait qu'il n'est pas nécessaire de refaire une description *MPEG-7* d'une image même si elle vient d'être redimensionnée. Dans ce cas, seule la correspondance entre l'image originale et celle modifiée est nécessaire.

**L'interpolation temporelle (Temporal Interpolation)**

L'interpolation temporelle propose une approximation de variables multidimensionnelles qui varient dans le temps. Cette interpolation est souvent utilisée pour le calcul de trajectoire dans une vidéo.

**B - Les descripteurs de couleur**

La couleur est, parmi tous les descripteurs, la plus représentative des caractéristiques visuelles. L'histogramme couleur est le plus fréquemment utilisé pour caractériser la couleur des images. Cependant un espace couleur doit être choisi pour la représenter. Souvent il s'agit de l'espace HSV[15] ou YCbCr[16] (YUV). En effet, il est connu que les différentes couleurs de l'espace HSV coïncident approximativement avec celle de la perception humaine des différentes couleurs. L'espace YCbCr est l'espace le plus utilisé dans les formats d'images et se comporte mieux que l'espace RGB : c'est donc souvent un espace de compromis. La comparaison entre deux histogrammes de deux images est réalisée par un calcul de distance. Les différentes méthodes sont issues du choix de la distance et de la façon de représenter un histogramme.

Des travaux considérables ont été effectués par le groupe *MPEG* pour concevoir des descripteurs efficaces et compacts permettant la recherche de similarité dans une base de vidéos. Après de nombreuses expérimentations, le groupe a défini les descripteurs de couleur suivants :

[15]HSV pour Hue Saturation Value est un modèle de représentation de couleur basé sur la *Teinte*, la *Saturation*, et la quantité de lumière que répand une couleur (*Valeur*)

[16]Voir Le *MPEG-1* : 3.2 Techniques de codage de l'image

**Le descripteur de l'espace de couleur (Color Space)**

Les différents espaces couleur utilisés sont la luminance, RGB, HSV, YCrCb et le nouveau HMMD. Ce nouvel espace supporté par *MPEG-7* possède quatre composantes : la *teinte* (Hue) qui est la même que dans l'espace HSV, le *Max* et le *Min* qui sont le maximum et le minimum parmi les composantes R,G,B et la *Dif* qui est définie comme la différence entre le Min et le Max. Lors de la recherche d'images, le groupe *MPEG* a observé que l'espace HMMD est très efficace. De plus, la conversion de l'espace RGB vers les autres espaces est possible. Le HSV est développé pour fournir une représentation intuitive de la couleur et pour approximer la façon dont les humains perçoivent les couleurs.

**Le descripteur de couleurs dominantes (Dominant Colors)**

Il permet d'obtenir les couleurs dominantes dans une région d'intérêt. Les couleurs dans l'image ou dans une région donnée sont d'abord regroupées en un petit nombre de couleurs représentatives. Le pourcentage de chacune des couleurs est ensuite calculé ainsi qu'une mesure de cohérence spatiale qui différencie les couleurs étendues sur toute l'image des couleurs localisées. La cohérence spatiale est un simple nombre qui représente l'homogénéité spatiale globale des couleurs dominantes dans l'image. Le nombre de couleurs par région est limité à 8 couleurs dominantes.

**Le descripteur de disposition des couleurs (Color Layout)**

Ce descripteur renseigne sur la distribution spatiale de la couleur dans l'image. Il constitue un descripteur efficace pour la recherche basée sur le croquis. Ce descripteur utilise une grille de 8*8 pixels qui permet d'extraire les couleurs dominantes selon la disposition spatiale. Le procédé d'extraction des caractéristiques s'effectue en 2 étapes : la première est de sélectionner la couleur représentative de chaque cas de la grille 8*8 et ensuite d'effectuer la transformée DCT sur la grille résultante. Il est d'ailleurs recommandé de prendre la couleur moyenne pour chaque bloc. L'espace de couleur choisi est le YCbCr.

**Le descripteur de structure des couleurs (Color Structure)**

Ce descripteur exprime la structure locale de la couleur dans une image en utilisant un élément structurant de 8*8 pixels. On compte le nombre de blocs possédant une couleur particulière. L'histogramme h(m) représente le nombre d'éléments structurants de 8*8 contenant un ou plusieurs pixels avec la couleur cm. Dans ce cas-ci, on ne tient donc pas compte uniquement de la couleur du pixel mais aussi de sa position.

**Le descripteur SCD (Scalable Color)**

Le SCD est un histogramme de couleur de l'espace HSV sur lequel on applique une transformation en ondelettes de Haar. Ce descripteur est utile pour la comparaison d'images et la recherche basée sur la couleur.

**Le descripteur de quantification des couleurs (Color Quantization)**

Ce descripteur définit une quantification uniforme des axes des différents espaces de couleur.

**C - Les descripteurs de textures**

La texture, comme la couleur, est un descripteur de bas niveau très puissant pour la recherche d'image. Une image peut en effet être considérée comme une mosaïque d'images de texture. *MPEG-7* définit trois descripteurs de texture.

**Le descripteur de parcours rapide de texture (Texture Browsing)**

Le descripteur de parcours rapide de texture sert à classifier grossièrement et rapidement les textures. Pour cela il prend en compte différents paramètres comme la direction, la régularité, l'orientation et la grossièreté des textures. Ce descripteur ne sert que pour les applications de recherche et classification rapide des textures. Premièrement, on utilise les filtres de Gabor[17] afin de déterminer et coder deux orientations de textures dominantes. Ensuite un filtre analyse à travers les orientations de textures dominantes la régularité et la grossièreté des textures.

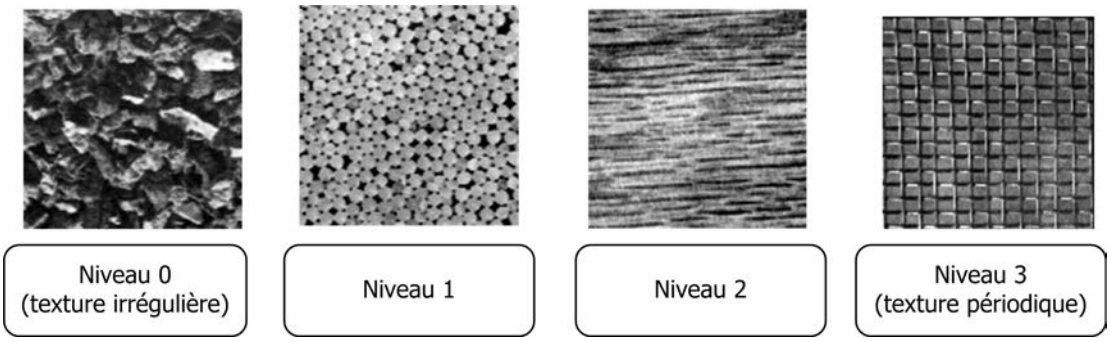


FIG. 7.20 – *Classification de la régularité d'une texture*

**L'histogramme d'orientation des contours (Edge Histogram)**

L'histogramme d'orientation des contours est le deuxième descripteur. Il représente la distribution spatiale des cinq types de bords. Ce descripteur est toujours utilisé pour faire des requêtes de similarité entre images. Son efficacité est grandement

[17]Un filtre de Gabor (ou Gabor filter) est un filtre linéaire. Il est défini par le produit entre un filtre Gaussien et une sinusoïdale orientée.

améliorée quand il est associé avec d'autres descripteurs comme l'histogramme de couleur.

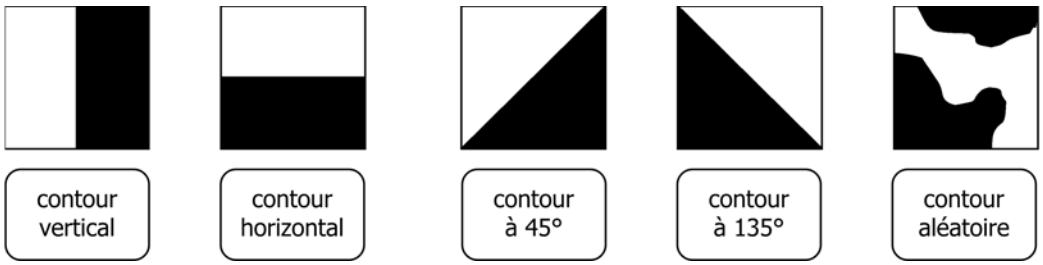


FIG. 7.21 – *Types de contours de l'histogramme des contours*

L'image est divisée en blocs de 4*4 pixels. De chaque bloc, on extrait les 5 caractéristiques de contour présentée à la figure 7.21. Il en résulte un histogramme d'orientation des contours.

**Le descripteur de texture homogène (Homogenous Texture Descriptors)**

Lorsqu'il s'agit de textures homogènes bien spécifiques l'histogramme d'orientation des contours se révèle trop élémentaire. On utilise alors le descripteur de texture homogène. Il génère une représentation quantitative qui est très pratique pour comparer des images. Pour cela on utilise les filtres de Gabor dans le domaine fréquentiel.

**D - Les descripteurs de formes**

La forme est un des éléments principaux utilisés par l'homme pour distinguer les données visuelles. En comparant avec la couleur et la texture, on se rend compte aisément que la forme est plus facile à décrire lors d'une recherche d'un document. Dans la littérature, on retrouve un grand nombre de descripteurs de forme, cependant, la plupart de ces descripteurs ne sont pas capables de définir l'ensemble des variétés de variation de forme. En effet, une forme peut être agrandie, diminuée, pivotée, allongée ou encore déformée. Pour couvrir l'ensemble de ces fonctions, *MPEG-7* a défini trois descripteurs de formes : le descripteur de zones, le descripteur de contour, le descripteur par spectre de forme 3D.

**Le descripteur de zones (Region Shape)**

Comme le montre la figure 7.22, la forme d'un objet peut être constituée d'une unique zone, d'un ensemble de zones disjointes ou encore de trous à l'intérieur d'un objet. Vu que le descripteur utilise tous les pixels constituant la forme d'une image, celui-ci peut décrire n'importe quelle forme, non seulement les formes simples constituées d'une seule zone [7.22 (a) et (b)] mais aussi des formes plus complexes comportant des trous [7.22 (c)] ou constitués de plusieurs zones [7.22 (d) et (e)]


FIG. 7.22 – *Formes constituées d'une zone unique, de trous ou des zones disjointes*

La figure 7.23 représente des formes très similaires de tasses. Malgré cette similarité, le descripteur peut considérer ces formes comme différentes. En effet les poignées de la figure [7.23 (a) et (b)] sont creuses tandis que la poignée de [7.23 (c)] est pleine. Le descripteur de zones va considérer les deux premières comme similaires et la troisième comme différentes des deux autres.

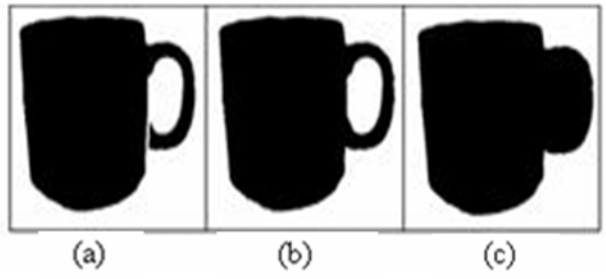


FIG. 7.23 – *Représentation de formes similaires*

**Le descripteur de coutour (Contour Shape)**

Ce descripteur, basé sur l'étude des comportements des courbures, repère les caractéristiques d'une forme d'un objet en se basant sur son contour fermé. La représentation faite par ce descripteur reflète bien la perception visuelle de l'homme et capture de très bonne manière les caractéristique d'une forme. Il est donc tout à fait adapté à la recherche par similarité.

Les trois résultats présentés à la figure 7.25 montre les caractéristiques intéressantes qui se dégagent du descripteur de contour. Premièrement ce descripteur est invariant par rapport à la taille et à la position des objets [7.25 (a)]. Ensuite il est invariant au mouvement des objet [7.25 (b)]. L'homme qui court est toujours considéré comme similaire malgré ces mouvements. Enfin le cheval met en évidence que le descripteur de contour est toujours opérationnel même s'il y a des parties de l'objet qui sont cachées [7.25 (c)].

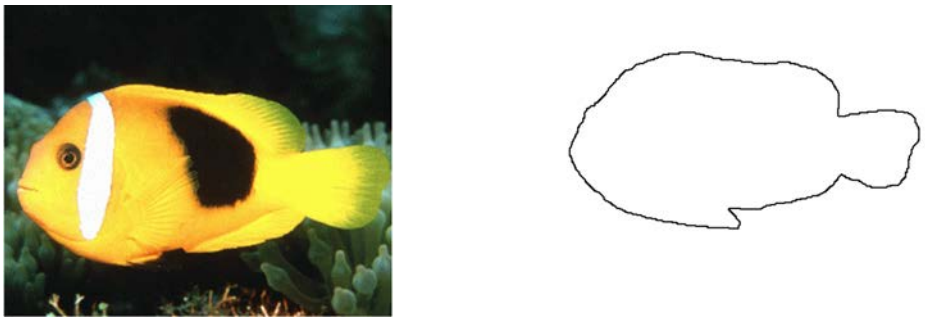

FIG. 7.24 – *Extraction du contour d'une forme d'une image*

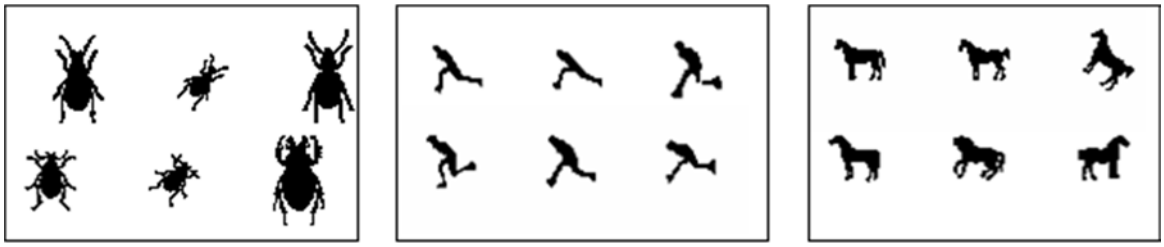


FIG. 7.25 – *Caractéristiques du descripteur de contour*

**Le descripteur par spectre de forme 3D (Shape 3D)**

Ce descripteur permet d'analyser les objets réels en les représentant comme des maillages 3D[18].

**E - Les descripteurs de mouvements**

Le but de l'indexation sur le mouvement est de capturer les caractéristiques du flot optique afin d'avoir des descripteurs concis et efficaces. L'obtention de ces caractéristiques dans le domaine de la compression de *MPEG* 1/2/4 est populaire vu la facilité d'extraction des vecteurs de mouvement contenus dans le flot binaire. Néanmoins, les vecteurs de mouvement sont constitués d'un champ peu dense. Ils ne peuvent pas être utilisés pour des descriptions qui demandent des champs de mouvement précis. Cependant même si le mouvement estimé est grossier, ils donnent déjà une information sur l'activité contenue dans l'image.

Dans la plupart des travaux actuels, l'obtention du flot optique entre deux images successives n'est pas souvent utilisée pour l'indexation vidéo. En effet, pour caractériser un mouvement de scène dans une séquence, les mesures de mouvement sont

[18]Voir Le *MPEG-4* : 5.3.3 Le codage visuel, Animation des maillages 2D/3D

calculées après compensation du mouvement dominant supposé provenir de la caméra. En général, les méthodes proposées sont complexes et nécessitent un temps de calcul trop important pour être mises en place. Souvent une mesure d'activité de la scène est préférée pour la représenter. Ces techniques se reposent généralement sur une différence d'images.

La norme *MPEG-7* a instauré les descripteurs de mouvement suivants :

**Le descripteur de mouvement de caméra (Camera Motion)**

Ce descripteur caractérise les paramètres de mouvement en trois dimensions d'une caméra. Ces paramètres peuvent être automatiquement extraits grâce à des outils de capture.

Le descripteur de mouvement de caméra supporte les opérations basiques de caméra suivantes :

- les plans fixes ;
- les rotations horizontales (*Panning*) ;
- les rotations verticales (*Tilting*) ;
- les déplacements le long de l'axe optique (*Dollying*) ;
- les changements de focale (*Zoom*) ;
- les mouvements tranversaux horizontaux (*Tracking, Travelling*) ;
- les mouvements tranversaux verticaux (*Booming*) ;
- les rotations autour de l'axe optique (*Rolling*).

La figure 7.26 issue de [ISO04a] illustre ces différentes opérations de caméra :

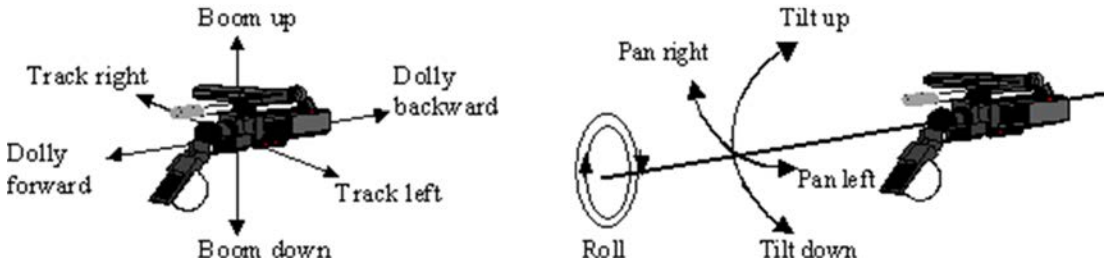


FIG. 7.26 – *Mouvements basiques de caméra*

**Le Descripteur de trajectoire de mouvement (Motion Trajectory)**

La trajectoire d'un objet est définie par la localisation successive dans l'espace et dans le temps d'un de ses points représentatifs (souvent le centre de gravité). Ce descripteur est essentiellement composé d'une liste de points-clés (x,y,z) pour la

position de l'objet dans l'espace et (t) pour le temps] et de fonctions qui décrivent le chemin de l'objet entre ces différents points-clés.

**Le descripteur d'activité (Motion activity)**

Le descripteur d'activité capture la notion intuitive d'intensité de l'action ou de rythme de l'action dans un segment de vidéo. Un exemple d'une haute activité dans une scène pourrait être un but marqué lors d'un match de football ou encore une course poursuite en voiture. Une scène de faible activité peut être représenter par une scène de dialogue entre deux acteurs.

**F - Les descripteurs de localisation**

*MPEG-7* définit deux descripteurs de localisation :

**Le délimiteur de zones (Region locator)**

Ce descripteur permet la localisation des zones d'une image en les représentant par une boite ou un polygone.

**Le délimiteur spatio-temporel(Spatio-temporal locator)**

Ce descripteur décrit des régions spatio-temporelles dans une séquence vidéo et fournit des fonctions de localisation. Cela peut s'avérer très utile pour une recherche d'objet vérifiant si l'objet est bien passé par un point particulier.

**G - La reconnaissance de visages**

Le descripteur de reconnaissance de visages peut être utilisé pour rechercher des images d'un visage qui correspondent à l'image d'un visage lors d'une requête.

### 7.3.3 Le MPEG-7 Audio

*MPEG-7* Audio définit conjointement avec les schémas de description multimédia un standard de description de document audio. Il regroupe un ensemble des descripteurs de bas niveau couvrant un large champ d'application (les caractéristiques spectrales , paramétriques ou encore temporelles d'un signal, par exemple). A côté des descripteurs de bas niveau, *MPEG-7* Audio fournit aussi un ensemble de descripteurs de haut niveau utilisés pour des applications plus spécifiques comme la reconnaissance vocale, la description du contenu sonore, la description du timbre, etc.

On peut retrouver :

**A - Les descripteurs de bas niveau (*MPEG-7* Audio Framework)**

Ces descripteurs sont au nombre de 18, illustré à la figure 7.27. On peut citer notamment le descripteur de silence qui permet d'identifier une absence de son signifiant dans une séquence audio. Les autres descripteurs sont décrits en détail dans [ISO04a].

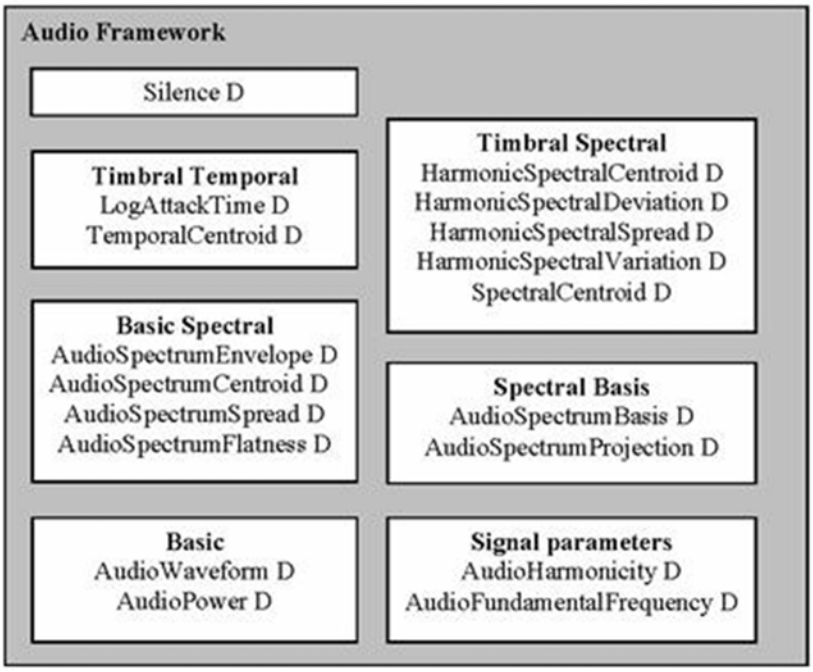


FIG. 7.27 – *Descripteurs audio* MPEG-7 *de bas niveau*

**B - Les descripteurs de haut niveau**

Les descripteurs audio de haut niveau concernent le contenu parlé, le timbre instrumental, la mélodie, la signature audio.

**Les outils de description du contenu parlé**

Ces descripteurs permettent une description détaillés des mots prononcés dans un flux audio. Ils peuvent permettre la recherche de flux audio ou encore d'objet multimédia contenant du son. Un exemple d'utilisation de ces descripteurs pourrait être une séquence vidéo où un personnage cite un mot particulier ou une combinaison de mots. Il serait possible en précisant certains mots dans une requête de retrouver le position de ces mots dans la séquence vidéo.

**Les outils de description de timbre instrumental**

Les descripteurs de timbre visent à décrire les éléments sonores perceptibles des instruments. Le timbre est couramment défini dans la littérature comme les éléments perceptibles qui rendent différents deux sons ayant mêmes tonalité et intensité. Ces descripteurs ont donc pour but de décrire ces éléments perceptibles et font référence à des notions tels que l'attaque, la clarté ou encore la richesse du son.

**Les outils de description de mélodies**

Le schéma de description de mélodies est une représentation compacte de l'information mélodique qui permet la recherche efficace de similarités entre mélodies.

**Le schéma de description de signature audio**

Le schéma de description de signature audio définit une représentation condensée d'un signal audio permettant de fournir un identifiant unique d'un contenu sonore dans le but d'une identification automatique et robuste des signaux audio.

**Les outils d'indexation et de reconnaissance de sons généraux**

Les outils de description d'indexation et de reconnaissance de sons généraux consistent en une collection d'outils dont le but est l'indexation et la catégorisation des sons généraux.

### 7.3.4 Les schémas de description multimédia

Les schémas de description multimédia (MDSs) sont des structures de métadonnées permettant de décrire et d'annoter du contenu audiovisuel. Ces structures fournissent un outil standard basé sur le *XML* permettant de décrire les concepts relatifs à la description de contenu audiovisuelle dans le but de faciliter les recherches, l'indexation, le filtrage et l'accès à l'information. Ces schémas de description sont définis à l'aide du langage de définition de description (DDL) qui lui-même est basé sur le *XML* Schema. Ces descriptions peuvent être stockées sous un forme textuelle (sous format *XML* compréhensible par l'homme) ou compressées sous forme binaire (pour la transmission par exemple).

Comme nous l'avons déjà dit précédemment, les descripteurs (Ds) sont utilisés principalement pour décrire des caractéristiques audio et visuelles de bas niveau telles que la couleur, la texture, le timing, etc. La majorité de ces informations de bas niveau peut être extraite automatiquement par des applications spécialisées. Les schémas de description (DSs) quant à eux permettent de décrire des caractéristiques de plus haut niveau telles que des zones, des segments, ou encore des formes. Ils

permettent de construire des descriptions très complexes en combinant plusieurs Ds ou DSs et en déclarant des relations entre différents descriptions.

Les schémas de description ont trait aux domaines du multimédia, de l'audio et du visuel. Alors que les schéma de description audio et visuel ne s'attachent qu'aux caractéristique de leur domaine respectif, les MDSs, eux, décrivent une combinaison de données textuelles, audio et visuelles. Dans certains cas, des outils d'extraction automatique de caractéristiques pourront être utilisés mais dans la majorités des autres l'extraction devra encore se faire par l'homme.

La figure 7.28 issu de[ISO04a] illustre une vue d'ensemble de l'organisation des MDSs *MPEG-7*. On y retrouve les éléments de base, la description de contenu, la gestion du contenu, l'organisation du contenu, la navigation et l'accès et l'interaction avec l'utilisateur.

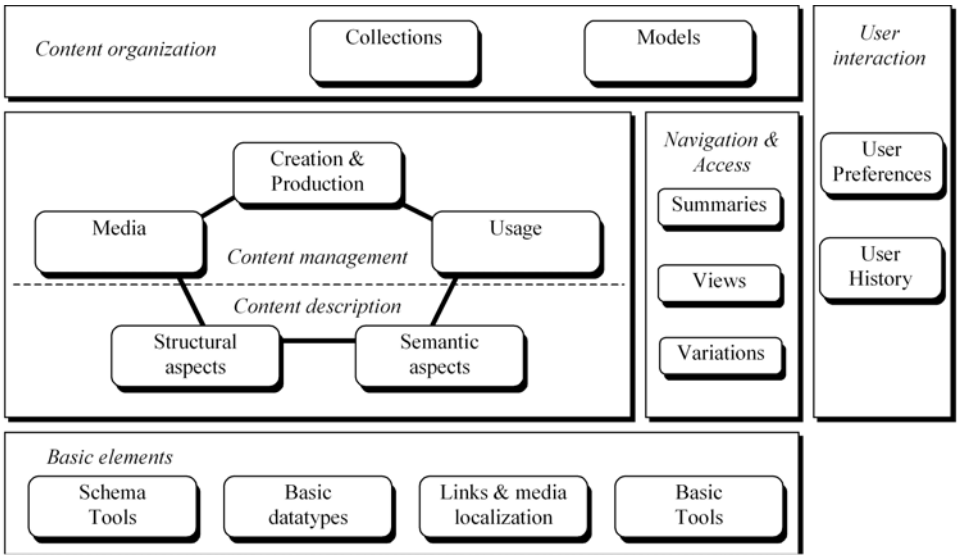


FIG. 7.28 – *Vue d'ensemble de l'organisation des MDS* MPEG-7

**A - Les éléments de base**

Les éléments de base permettent de construire les fondements d'un schéma de description. Les types de données de base fournissent un ensemble de types de données étendus et de structures mathématiques (telles que les vecteurs ou les matrices) nécessaires à la description d'un contenu audiovisuel par un schéma de description. Ces éléments de base comprennent aussi des structures permettant de lier différents fichiers média, de localiser des éléments d'un contenu et de décrire des caractéristiques comme le temps, les lieux, les groupes ou encore des annotations textuelles.

### B - La description du contenu

*MPEG-7* fournit des schéma de description caractérisant la structure et la sémantique du contenu audiovisuel.

#### Les aspects structurels

L'élément principal de cette partie de la description est le segment. Il adresse la description des aspects physiques et logiques d'un contenu audio-visuel. Le segment décrit le résultat d'un découpage spatial, temporel, ou spatio-temporel du contenu audiovisuel. Le segment peut décrire une décomposition récursive ou hiérarchique du contenu audiovisuel en segments qui forment un arbre de segments. Par exemple, un programme vidéo peut être segmenté en plusieurs niveaux de scènes ; une table des matières peut donc être basée sur cette structure.

Un segment temporel peut être un ensemble d'échantillons dans une séquence sonore, représenté par un segment audio, un ensemble d'images dans une séquence de la vidéo, représenté par un segment vidéo ou une combinaison d'informations sonores et visuelles décrites par un segment audiovisuel. Un segment spatial peut être une région dans une image ou une image dans une séquence vidéo, représentée par une région immobile pour de la 2D et une région immobile 3D pour de la 3D. Un segment spatio-temporel peut correspondre à une région en mouvement dans une séquence vidéo.

Tout segment peut être décrit par les informations de création, d'usage et d'annotation textuelle. Par ailleurs, les caractéristiques spécifiques selon le type du segment sont aussi possibles. Ces caractéristiques spécifiques sont rapportées dans le Tableau de la figure 7.29.

| Caractéristiques | Segment vidéo | Région immobile | Région en mouvement | Segment audio |
|---|---|---|---|---|
| Temps | ✓ | | ✓ | ✓ |
| Forme | | ✓ | ✓ | |
| Couleur | ✓ | ✓ | ✓ | |
| Texture | | ✓ | | |
| Mouvement | ✓ | | ✓ | |
| Mouvement de caméra | ✓ | | | |
| audio | | | ✓ | ✓ |

FIG. 7.29 – *Caractéristiques spécifiques pour la description d'un segment*

Un exemple de description d'image est illustré dans la Figure 7.30. L'image originale est décrite comme une Région Immobile, SR1[19], qui est décrite par création (titre, créateur), information d'usage (copyright), information médiatique (format de fichier) aussi bien que d'une annotation textuelle (résumé du contenu de l'image), un histogramme de couleur et un descripteur de la texture. Cette région initiale peut être décomposée en régions individuelles. Pour chaque étape de la décomposition, nous indiquons si les Intervalles de temps (Gap) et Chevauchements (Overlap) sont permis. L'arbre du segment est encore composé de 8 régions (à noter que SR8 est un segment unique fait de deux composants connectés). Pour chaque région, la figure 7.30 montre le type de caractéristique qui est répertorié. On note qu'il n'est pas nécessaire de répéter dans la hiérarchie de l'arbre la création, l'information d'usage, et l'information médiatique, puisque le segment enfant est supposé hériter des valeurs du segment parent.

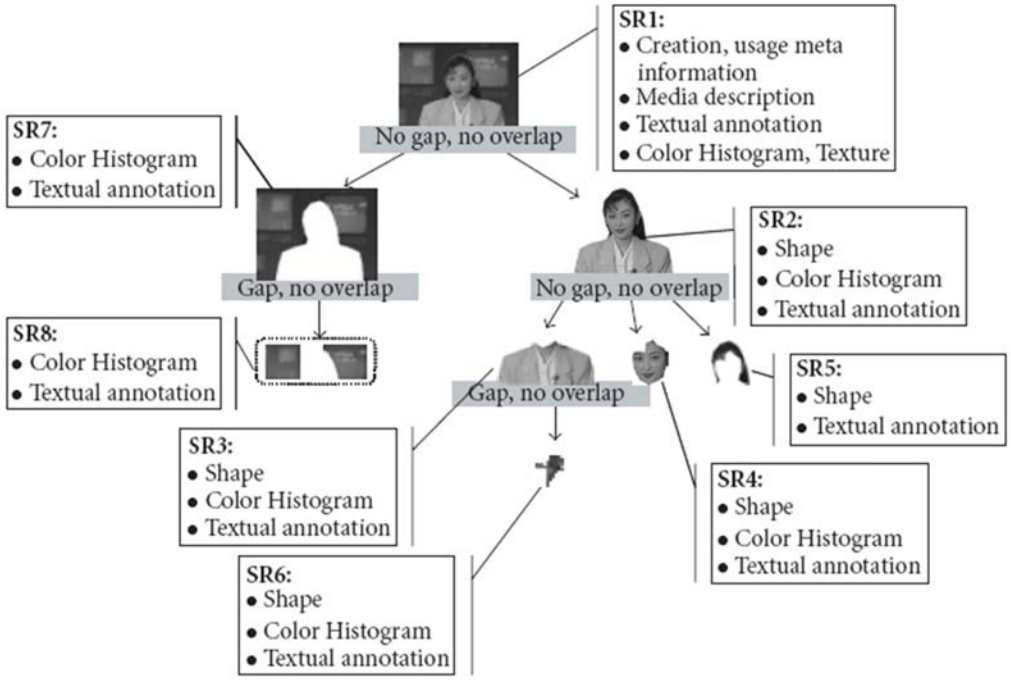


FIG. 7.30 – *Description d'une image par des segments de Régions Immobiles*

**Les aspects sémantiques**

Il existe une autre approche basée sur les schémas de description sémantiques. Dans cette approche, l'emphase n'est plus mise sur les segments mais sur les événements, objets et concepts du contenu à décrire dans le monde narratif. Le niveau d'abstraction est ici bien plus élevé que pour l'aspect structurel. On retrouve des

[19]SR pour Still Region (Région Immobile)

entités sémantiques décrivant les objets, les événements, les interactions entre les objets ou encore le lieu de l'action.

A titre d'exemple, la figure 7.31 montre un exemple de description d'une séquence vidéo. Cette description est représentée par deux structures hiérarchiques représentées sous forme d'arbre. Le premier, basé sur les segments, concerne les aspects structurels ; le deuxième décrit ce qui se passe dans la séquence, c'est l'aspect sémantique. Les liens entre l'arbre des segments et l'arbre sémantique retracent les différents événements de la séquence vidéo. Un même événement peut évidemment se rattacher à un ou plusieurs segments.

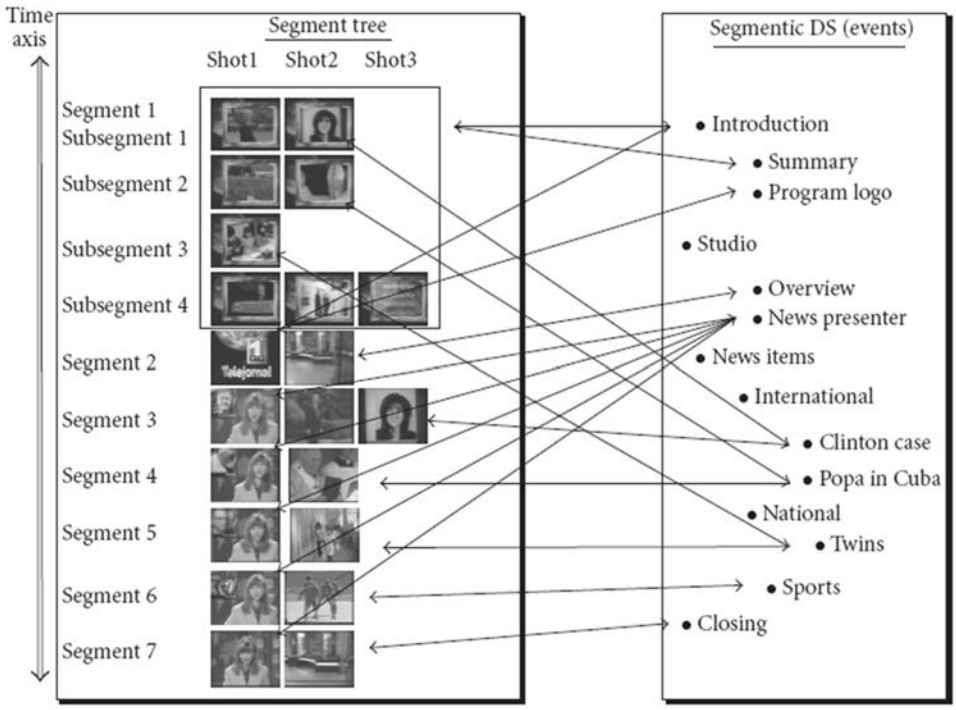


FIG. 7.31 – *Exemple de description combinant l'aspect structurel et sémantique*

**C - La gestion du contenu**

*MPEG-7* fournit des schémas de description pour la gestion du contenu audiovisuel. Ils décrivent des caractéristiques qui ne concernent pas généralement le contenu lui-même mais pouvant être d'une importance vitale pour beaucoup d'applications. Ces outils décrivent les informations suivantes : la création et la production, le type de média, les conditions d'accès au support.

Les informations de création et production fournissent des éléments de description tels que le titre, les créateurs, l'endroit de réalisation, et les dates de diffusion/création du média. Les schémas de description de gestion fournissent aussi des informations de classification sur les média (genre, sujet, langage, accord parental, ...).

Les informations sur le type de média concernent essentiellement le format du stockage et l'encodage du contenu audiovisuel (taux et format de compression) ou encore l'identification du média original à partir duquel différentes séquences sont produites.

Les informations sur les conditions d'accès englobent des notions telles que les détenteurs de droit, les informations financières, etc.

**D - L'organisation du contenu**

L'organisation du contenu est construite autour de schémas de description : les schémas *Collection* et *Model*. Le schéma *Collection* structure des collections de contenus audiovisuels, de concepts sémantiques, de segments ou encore d'objets. De cette façon, il est possible d'établir des propriétés communes à un ensemble d'éléments. Le schéma *model* fournit des outils permettant de modéliser les attributs et les caractéristiques du contenu audiovisuel.

**E - La navigation et l'accès**

*MPEG-7* facilite la navigation et l'accès des contenus audiovisuels en fournissant un schéma de description décrivant des récapitulatifs, des vues ou encore des variations du matériel audiovisuel. Les récapitulatifs fournissent des résumés de contenu audiovisuel permettant notamment de parcourir et visualiser le contenu audiovisuel. Les vues décrivent différentes décompositions de signaux audiovisuels dans le temps, l'espace et la fréquence. Les variations, quant à elles, décrivent les relations entre les différentes versions d'un même programme audiovisuel (les qualités de compression, les langages, ...). La principale fonctionnalité du schéma de description variation est de permettre à un serveur de choisir la version d'un programme audiovisuel par rapport au préférence d'un utilisateur.

**F - L'interaction avec l'utilisateur**

Le schéma de description relatif à l'interaction avec l'utilisateur permet de décrire les préférences et l'historique des utilisateurs quant à leur consommation de documents multimédia. De cette façon, il est possible de faire correspondre les descriptions de contenu *MPEG-7* avec les préférences des utilisateurs afin de faciliter l'accès, la présentation et la consommation de document multimédia.

## 7.4 Conclusion

*MPEG-1* et *MPEG-2* sont des standards à succès et il est fort probable que *MPEG-4* suivent la même voie. Avec le *MPEG-7*, le groupe *MPEG* tente de résoudre les problèmes de l'identification, de la recherche et de la gestion des différents types de documents multimédia. En comparant avec les autres standards de description, on peut caractériser *MPEG-7* de la sorte.

1. **Générique**. Il offre la possibilité de décrire la plupart des document multimédia sur la plupart des environnements (car il se base sur le *XML*).
2. **Complet**. Il intègre plusieurs niveaux d'abstraction de description et offre la possibilité de les combiner.
3. **Extensible**. Grâce au DDL, il est possible de créer de nouveaux descripteurs ou schémas de description ou encore de modifier des descripteurs et schémas existants.
4. **Prometteur**. Encore confidentiel en 1997, *MPEG-7* commence à faire l'objet d'une intense coopération internationale, impliquant de nombreux laboratoires de recherche universitaires et des opérateurs de télécommunications.

Au vu des ses caractéristiques, *MPEG-7*semble être un candidat idéal pour s'intégrer dans une application d'indexation et de recherche de document multimédia. Si nous prenons comme exemple une application de recherche d'image, les descripteurs et schémas de description fournis par *MPEG-7* permettrait non seulement de décrire des caractéristiques sémantiques mais aussi des caractéristiques visuelles. De plus pour ces dernières, il serait possible d'utiliser des outils d'extraction automatique.

Nous présenterons dans la quatrième partie deux architectures permettant l'indexation et la recherche d'image. Pour des raisons de licence, ces deux architecture utiliseront des descripteurs *XML* équivalent au standard *MPEG-7*. L'indexation portera aussi bien sur les caractéristiques sémantiques encodées manuellement par un utilisateur que sur les caractéristiques visuelles extraites automatiquement par des outils d'analyse d'image.

Afin de réaliser ce chapitre, nous nous sommes inspirés des documents suivants : [Tor06], [Cha05], [XML06b], [XML06a], [EVV00] pour le *XML* et le *XML* Schema et [GL04], [All98], [ISO04a], [EPM05] pour le *MPEG-7*.

CHAPITRE 8

# MPEG-21

## 8.1 Introduction

Durant ces dernières années le nombre des documents électroniques (images, textes, sons, vidéos,...) a littéralement explosé. Parallèlement à la production sans cesse croissante de ces documents multimédia, les moyens d'accès à ces différents contenus ont eux aussi fortement évolué. Cependant il n'y a pas encore vraiment d'outils à la disposition des utilisateurs permettant des gérer les difficultés liées à ces nouveaux documents multimédia (les droits intellectuels notamment), mais le besoin de nouvelles solutions dans ce domaine est omniprésent.

Les modèles existants de commerce et d'échange de biens physiques sont inappropriés dans le monde électronique. Il a donc fallu développer de nouveaux modèles pour la distribution et l'échange de contenus électroniques, notamment pour le respect des droits de propriété intellectuelle, respect de la vie privée, protection contre l'accès ou la modification par des personnes non-autorisées,...

Le manque de solutions technologiques dans ce domaine a motivé la création de *MPEG-21*. Son but est de fournir des solutions pour la diffusion, la gestion et la protection des contenus électroniques.

*MPEG-21* est basé sur deux concepts essentiels : le Digital Item qui est l'unité fondamentale pour la distribution et les transactions, communément appelé contenu

(collection vidéo, album musical,...), et le concept de User qui est l'utilisateur qui interagit avec les Digital Item.

Le but de *MPEG-21* était de définir la technologie nécessaire permettant au User (utilisateur) d'accéder, d'échanger, de commercialiser et manipuler les Digital Item de manière efficiente, transparente et interopérable. *MPEG-21* identifie et définit les mécanismes et les éléments nécessaires à toute la chaîne de production et de consommation de documents multimédia. Il définit aussi les relations entre les différentes parties de la chaîne.

## 8.2 Le modèle "User"

Par User *MPEG-21* entend toute personne interagissant dans l'environnement *MPEG-21*. On retrouve parmi les utilisateurs : les consommateurs, les communautés, les gouvernements, les organisations, les consortiums, .... *MPEG-21* ne fait aucune distinction entre les producteurs et les consommateurs de contenu. Les utilisateurs sont identifiés par leur relation avec un autre utilisateur pour une certaine interaction.

*MPEG-21* fournit un framework dans lequel un User interagit avec un autre User et l'objet de leur interaction est un Digital Item. Par interaction *MPEG-21* reprend la création de contenu, la fourniture de contenu, l'archivage de contenu, la modification de contenu, le regroupement de différents contenus, la vente de contenu, la consommation de contenu, la régulation de contenu,....

## 8.3 Le Digital Item

*MPEG-21* se propose de traiter une très large palette d'actions manipulant des Digital Item. Le grand défi est donc d'arriver à pouvoir décrire le contenu de chaque DI[1] de façon précise. Considérant qu'il y a énormément de types de contenu différents et autant de façons de décrire le contexte d'utilisation, la difficulté réside dans le fait de pouvoir établir un modèle flexible et puissant à la fois qui puisse être applicable à toutes les sortes de contenu. De plus il est évident que les descriptions ne doivent pas être ambiguës.

Si on prend le cas d'une simple page web constituée de texte, de quelques liens et d'images, il est assez simple de parcourir le document HTML et de déduire tout

[1] Digital Item

ce qui constitue la page web. Maintenant si on prend la même page web et qu'on lui ajoute un script permettant l'affichage de la page dans différentes langues selon le choix de l'utilisateur, se pose alors le problème de savoir si chaque traduction de la page constitue un nouveau DI, ou si l'ensemble des pages dans les différentes langues ne constitue qu'un seul DI.

## 8.4 Contenu de MPEG-21

### 8.4.1 MPEG-21 partie 1 : Vision, Technologies et Stratégie

Cette partie est en fait un rapport technique décrivant le framework multimédia ainsi que tous ses éléments architecturaux et les exigences fonctionnelles pour leur spécification.

Elle définit premièrement la vision d'un framework multimédia permettant l'utilisation transparente de ressources multimédia. Cette utilisation se fait au travers de nombreux réseaux et les ressources sont manipulées par un grand nombre d'appareils différents. Il faut donc essayer de combiner les attentes de tous les utilisateurs.

Ensuite afin de faciliter l'harmonisation des technologies, la première partie de *MPEG-21* réalise l'intégration des composants et des standards pour la création, la gestion, le transport, la manipulation, la distribution et la consommation de DI.

Finalement elle définit une stratégie pour réaliser un framework multimédia via le développement de spécifications et de standards basés sur des exigences fonctionnelles recueillies auprès des utilisateurs potentiels de *MPEG-21*.

### 8.4.2 MPEG-21 partie 2 : Digital Item Declaration

La DID[2] est décrite en trois parties :

- Model : le modèle du DID fournit un ensemble de termes abstraits ainsi que des concepts permettant de décrire les DI de manière flexible et générale.
- Representation : une description normalisée de la syntaxe et de la sémantique de chaque élément de la DID, représenté en XML. Cette section contient aussi des exemples illustratifs non normalisés.

[2] Digital Item Declaration

– Schema : un schéma XML normalisé comprenant la grammaire entière de la représentation XML du DID.

Le DIDL[3] est le langage de déclaration des DI. Il définit une structure hiérarchique des divers éléments du langage qui sont repris ci-dessous. Les règles grammaticales sont exprimées en EBNF[4]. Il y a d'abord les entités de structure : container, item, component, ressource, anchor et fragment, ensuite les entités de description : descriptor, statement et annotation et pour terminer il y a les entités de choix : choice, condition, selection, assertion et predicate.

**Container** Une *container* est une structure permettant de regrouper des *items* et/ou des *containers* formant ainsi des groupement physiques pour le transport et l'échange ou des groupes logiques pour structurer un contenu. Les *descriptors* permettent de décrire ces groupements (notamment sur la nature ou l'utilité du groupe). Il faut noter qu'un *container* ne constitue pas un *item* mais seulement un groupement d'*items* et/ou de *containers*.

$$container ::= descriptor^* \ container^* \ item^*$$

**Item** Un *item* est un groupement de *sous-items* et/ou de *components* qui sont reliés à des *descriptors* appropriés. Les *descriptors* donnent de l'information sémantique sur les *items* en tant que représentation d'un travail. Les *items* peuvent aussi être conditionnels lorsqu'il comprennent des *predicates* basés sur des *selections* définis dans des *choices*. Un *item* qui ne contient aucun *sous-item* est considéré comme une entité, un *item* indivisible alors qu'un *item* contenant des *sous-items* est considéré comme une collection logique d'*items* potentiellement indépendants. Un *item* est en fait la représentation déclarative d'un DI.

$$item ::= condition^* \ descriptor^* \ choice^* \ (item|component)^* \ annotation^*$$

**Component** Un *component* est un élément qui lie une *resource* à tous ses *descriptors*. Dans ce cas-ci les *descriptors* contiennent des informations de contrôle ou de structure (débit binaire, points de départ, information de cryptage, table de caractères,...), mais aucune information décrivant le contenu. Il est à noter aussi qu'un *component* ne constitue pas un *item* mais que les *components* permettent la construction des *items*.

$$component ::= condition^* \ descriptor^* \ resource \ anchor^*$$

**Resource** Une *resource* est un objet média identifiable individuellement tel qu'un clip vidéo ou audio, une image ou du texte. La *resource* n'est généralement pas contenue dans le document mais on utilise une référence à cette *resource*. Donc toutes les *resources* doivent pouvoir être localisées de manière non-ambiguë.

[3] Digital Item Declaration Language

[4] ISO/IEC 14977 :1996, Information technology — Syntactic metalanguage — Extended BNF

**Anchor** L'*anchor* permet de lier des *descriptors* à un *fragment*. Un *fragment* sert à pointer sur une partie à l'intérieur d'une *resource*.

$$\boxed{anchor ::= condition^* \ descriptor^* \ fragment}$$

**Fragment** Un *fragment* sert à pointer de manière non-ambiguë sur une partie à l'intérieur d'une *resource*.

**Descriptor** Un *descriptor* est une enveloppe de métadonnées qui est associé à l'élément qui l'englobe. L'élément englobant peut être un *item*, un *component* ou une *resource*. Les information contenues dans un *descriptor* peuvent être soit un *component* soit un *statement*. A ce stade, le modèle n'est pas très directif sur la façon d'utiliser ces enveloppes de métadonnées. On sait juste qu'il vaut mieux mettre un descriptif au niveau de l'*item* et une description technique au niveau des *resources*.

$$\boxed{descriptor ::= condition^* \ descriptor^* \ (component|statement)}$$

**Statement** Un *statement* contient une description textuelle de l'élément parent et non pas un objet à décrire. On y retrouve des information de description, d'identification, de contrôle ou de version.

**Annotation** Une *annotation* fournit un ensemble d'informations sur un autre élément du modèle sans altérer l'élément décrit. L'information peut se présenter sous forme d'*assertion*,d'*anchor* et de *descriptor*.

$$\boxed{annotation ::= assertion^* descriptor^* anchor^*}$$

**Choice** Un *choice* regroupe un ensemble de *selection* permettant de configurer un *item* lors d'une utilisation. Les *selection* d'un *choice* sont inclusives (en choisir un certain nombre, toutes ou aucune) ou exclusives (en choisir une seule).

$$\boxed{choice ::= condition^* descriptor^* selection^+}$$

**Condition** Une *condition* rend l'élément qui l'englobe optionnel. Elle lie l'élément au(x) *selection(s)* qui affecte(nt) son inclusion. Lorsqu'il y a plusieurs *predicates* dans une *condition*, elles sont liées par une relation de conjonction (ET). Les *prédicates* d'une *condition* peuvent être niés. Lorsqu'il y a plusieurs *conditions* elle sont liées par une relation de disjonction (OU).

$$\boxed{condition ::= predicate^+}$$

**Selection** Une *selection* décrit une décision spécifique prise par l'entité qui manipule un *item* (utilisateur, player,...) et qui affecte une ou plusieurs *conditions* de cet *item*. Si la *selection* est choisie son *predicate* prend la valeur vraie, si elle n'est pas choisie son *predicate* prend la valeur faux et si elle est laissée non-résolue, le *predicate* a la valeur indeterminé.

$$selection ::= condition^* descriptor^* predicate$$

**Assertion** Une *assertion* définit un état totalement ou partiellement configuré d'un *choice* en assignant les valeurs vrai, faux, indéterminé à certain *predicates* associés aux *selections* de ce *choice*

$$assertion ::= predicate^*$$

**Predicate** Un *predicate* est une déclaration identifiable de manière non-ambiguë qui peut être vraie, fausse ou indéterminée.

La figure 8.1 montre les éléments les plus importants du modèle, comment ils sont liés et leur place dans la hiérarchie des éléments du modèle. Pour de plus amples explication voir le rapport technique ISO/IEC TR 21000-2 :2005.

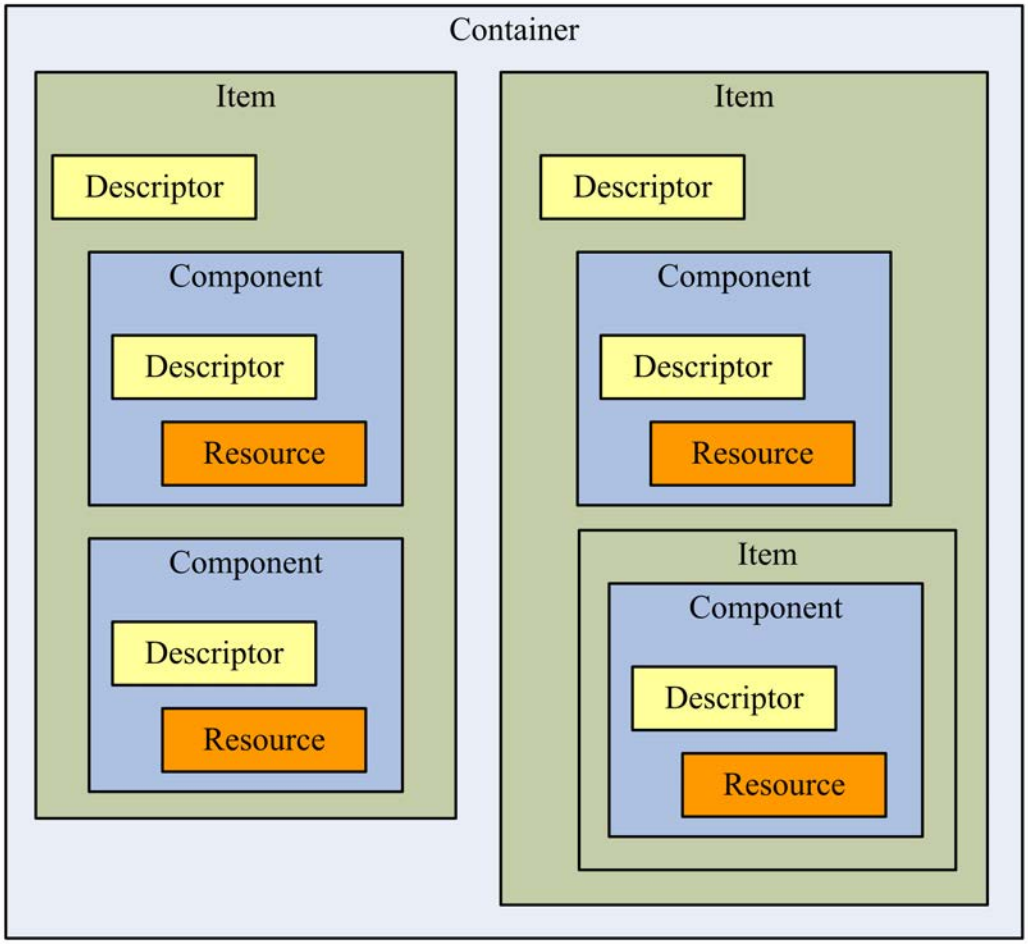


FIG. 8.1 – *Relations entre les principaux éléments du modèle DID*

### 8.4.3 MPEG-21 partie 3 : Digital Item Identification

Le but de cette partie, la DII[5], est de spécifier le moyen d'identifier de façon unique :

- un DI et les différents éléments qui composent ce DI.
- les Description Schemes en utilisant des namespaces XML.
- les différent types de DI.

La DII explique aussi comment utiliser des identifiants pour lier les DI aux informations qui les concernent telles que les métadonnées descriptives.

La spécification du DII n'introduit pas de nouveaux systèmes d'identification des éléments pour lesquels des schémas de description et d'identification existent déjà et sont utilisés. Par exemple, elle ne remplace pas l'ISRC[6] des enregistrements audio, mais permet leur utilisation au sein d'*MPEG-21*.

Pour utiliser des identifiants déjà existants, il suffit de les inclure dans un endroit spécifique de la Digital Item Declaration : le statement. Celui-ci inclut généralement des informations descriptives, de contrôle et/ou d'identification.

La figure 8.2 montre le lien entre la DID[7] et la DII. Les éléments de la DID (en bleu dans la figure) peuvent contenir zéro, un ou plusieurs descriptors. Chaque descriptor peut contenir un identifiant relatif à l'élément parent du statement (en vert dans la figure). Dans la figure 8.2, deux statements servent à identifier un component (celui de gauche) et un item (celui de droite).

*Identifier un Digital Item*

Pour identifier un DI (et les éléments qui le composent), la DII indique que l'on peut utiliser n'importe quel identifiant sous forme d'URI[8]. Une URI est un string compact de caractères servant à l'identification d'une ressource abstraite ou physique. La figure 8.3 à la page 127 montre comment un album de musique pourrait être identifié grâce à la DII.

*Identifier différents Description Schemes*

[5] Digital Item Identification

[6] Le code ISRC (International Standard Recording Code) est l'instrument d'identification unique des enregistrements sonores et audiovisuels. Ce code permet d'identifier individuellement des enregistrements dans le monde entier. Grâce à cette identification unique, l'enregistrement utilisé (et l'ayant droit) peuvent facilement être identifiés.

[7] Digital Item Declaration

[8] Uniform Resource Indentifier

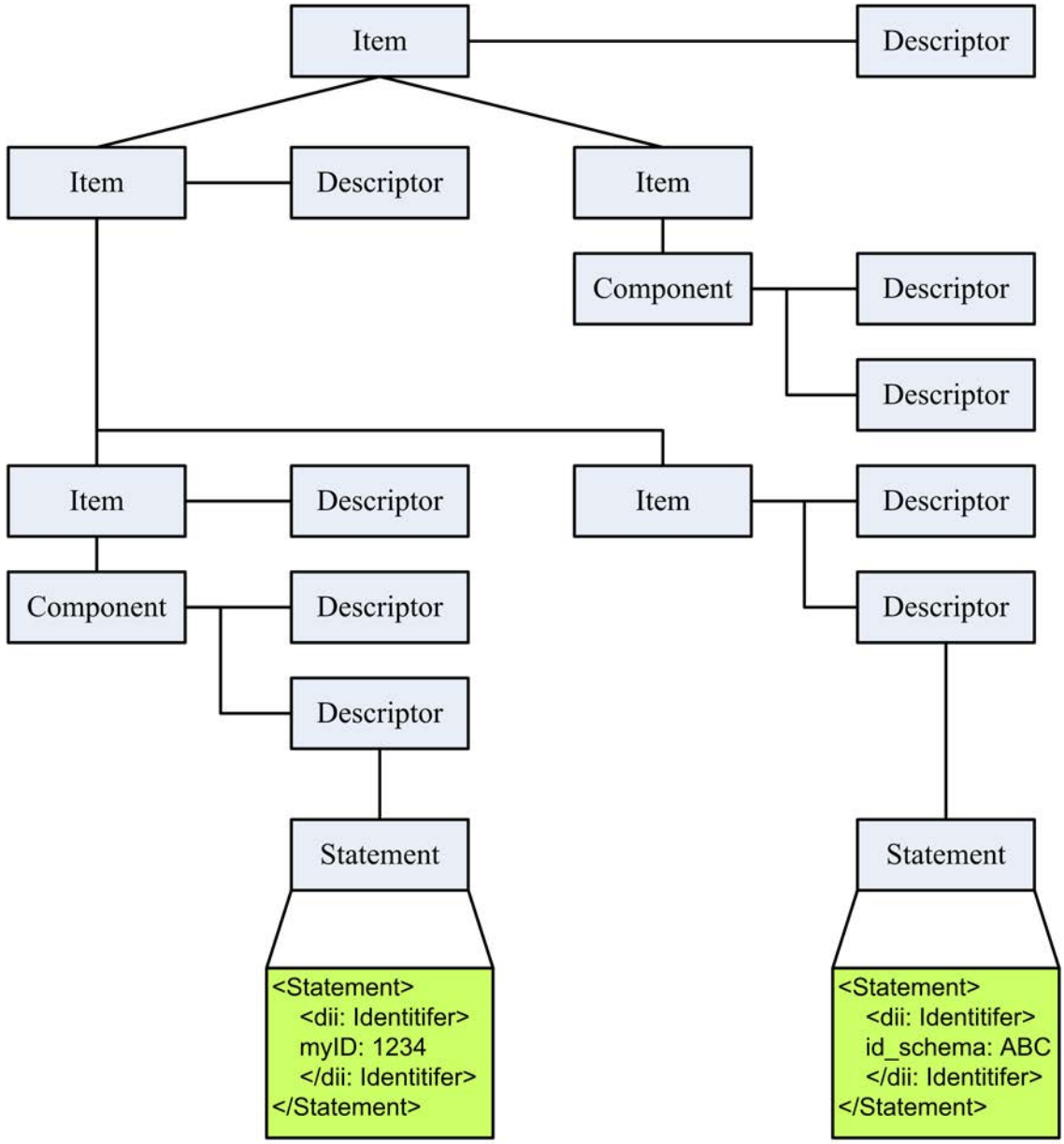


FIG. 8.2 – *Relation entre DI Declaration et DI Identification*

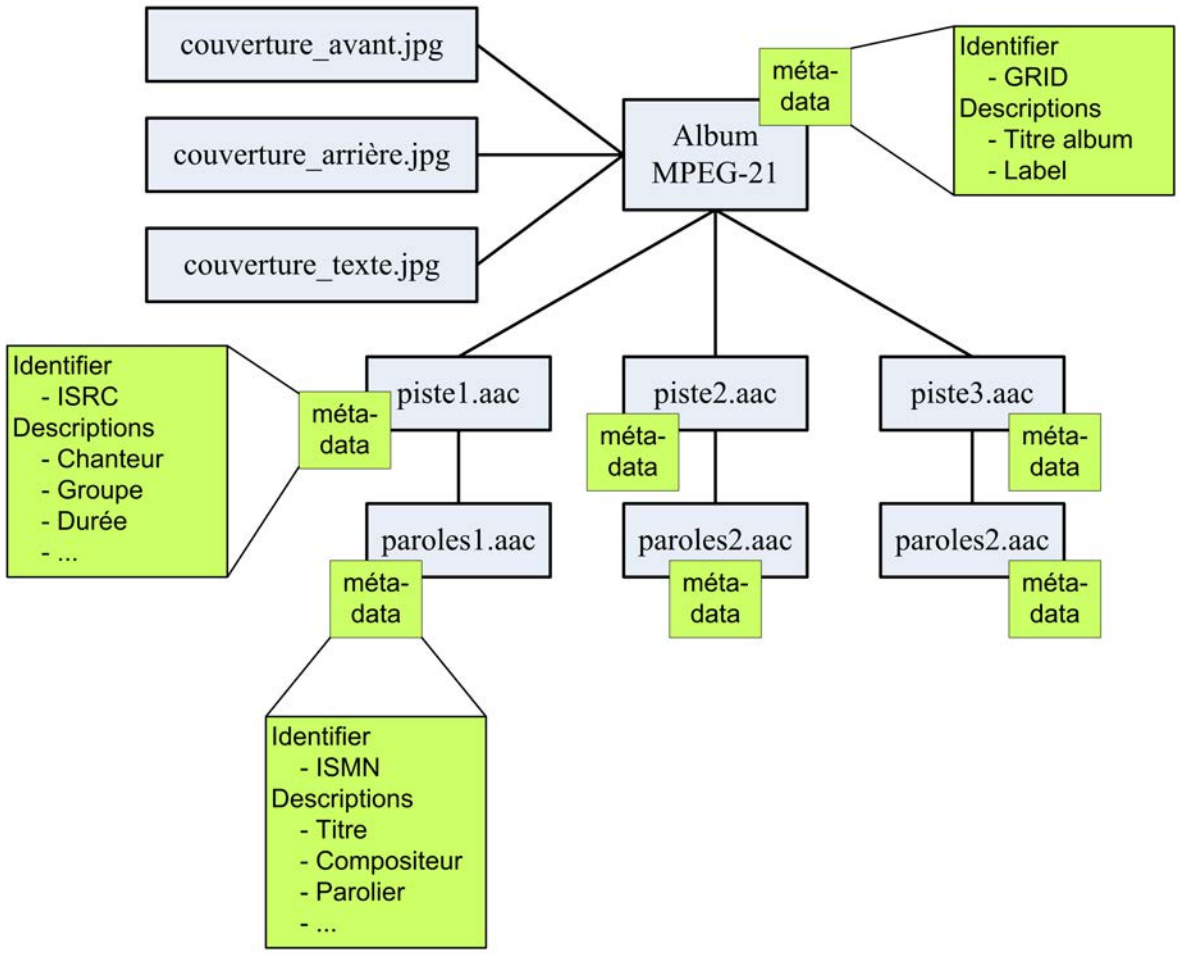


FIG. 8.3 – *Métadata et identifiant d'un album de musique MPEG-21*

Comme chaque utilisateur de *MPEG-21* peut avoir un schéma de description différent des autres utilisateurs pour décrire son contenu, il faut que la DII puisse différencier ces différents schémas de description. Pour cela la DII utilise le mécanisme des namespace du XML

*Identifier les différents types de DI*

Les différentes parties de *MPEG-21* définissent différents types de DI. On retrouve par exemple la Digital Item Adaptation (DIA) qui définit le Context Digital Item (XDI) en plus du Content Digital Item (CDI). Alors que le CDI contient des ressources telles que des fichier MP3 ou des flux vidéo *MPEG-2*, le XDI contient des informations sur le contexte dans lequel un CDI sera utilisé.

La DII fournit un mécanisme permettant à un terminal *MPEG-21* de distinguer les différents types de DI. Cela simplement en utilisant une URI placé dans le Type

Tag d'un statement d'un descriptor d'un item. Si aucun Type Tag n'est présent, le DI sera considéré comme un CDI.

### 8.4.4 MPEG-21 partie 4 : Intellectual Property Management and Protection

Cette quatrième partie de *MPEG-21*, IPMP[9], tente de définir une framework interopérable pour la gestion et la protection des droits intellectuels (IPMP). Cette partie a été développée dans la continuité de la protection des droits intellectuels de *MPEG-4*. Une fois *MPEG-4* devenu un standard international, le problème de l'interopérabilité fut rapidement soulevé. En effet, des appareils *MPEG-4* (lecteurs, encodeurs,...) très similaires furent développés par différents constructeurs, mais ces différents appareils n'interagissaient pas entre eux.

Voila pourquoi *MPEG* a décidé de développer de nouveaux systèmes et outils permettant une plus grande interopérabilité au point de vue de la protection des droits intellectuels. Ce projet permet notamment de télécharger des outils IPMP à partir de diverses sources, d'échanger des messages entre outils IPMP (watermarking, décryptage), d'authentifier les outils IPMP (certificats), ....

### 8.4.5 MPEG-21 partie 5 : Right Expression Language

REL[10] est un langage machine, défini via XML Schema, permettant de décrire les droits et les permissions associés aux oeuvres numériques, utilisant les termes définis dans le Right Data Dictionary[11]

REL tente de fournir des mécanismes flexibles, interopérables et transparents aidant à la protection des contenus numériques et respectant les droits, les conditions et les éventuels tarifs associés aux créations numérique (films, musique, livres électroniques, jeux, programmes, ...), y compris les créations et données privées, sur toute la chaîne de distribution (publication, distribution, consommation, ...).

*Le modèle REL*

Le modèle REL, exposé à la figure 8.4 page 129, est basé sur quatre entités de base et sur les relations entre ces entités. Les relations entre entités sont définies par une "MPEG REL grant", une permission du modèle REL de *MPEG*. Chaque permission exprimant un droit contient les quatre entités de base :

[9] Intellectual Property Management and Protection
[10] Right Expression Language
[11] voir points suivant : partie six

– *principal* qui est la personne à qui la permission est concédée
– *right* qui est le droit concerné par la permission
– *resource*, la ressource à laquelle s'applique la permission
– *condition*, les conditions qui doivent être remplies avant que le droit puisse être exercé.

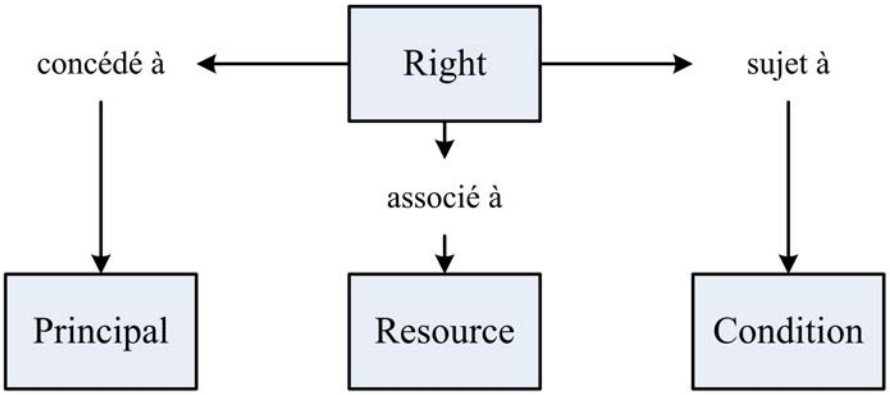


FIG. 8.4 – *Le modèle REL*

*L'entité Right*

Le *right* spécifie l'action, l'activité ou l'ensemble d'actions que le *principal* a le droit d'effectuer sur la ressource concernée.

Via REL, l'entité *right* contient des information sur le droit. REL fournit un ensemble de droits communément utilisés. De plus, des extensions de REL permettent de définir des droits plus appropriés à l'utilisation d'un type spécifique de ressources. Par exemple l'impression de tous les documents d'un certain type.

*L'entité Princial*

Le *principal* contient l'identification de la partie à qui le droit est concédé (la partie peut être un ensemble de personnes, c'est le User définit par *MPEG-21*). Dans la pratique, les données se trouvant dans le *principal* servant à l'identification de celui-ci, sont associées à des mécanisme d'authentification tel que : clé secrète, clé privée / clé publique, ....

```
User XXX, vous pouvez visionner mon film
```

*L'entité Resource*

La *resource* est "l'objet" pour lequel le *principal* détient le droit tel que : un fichier numérique (e-book, fichier vidéo ou audio, une image, ...), un service (service de mail, service de transaction B2B, ...) et même des informations personnelles telles qu'une adresse e-mail.

REL fournit des mécanismes d'encapsulation d'information nécessaires à l'identification de *resources* particulières ou de *resources* d'un certain type (des ressources ayant des caractéristiques communes). Des extensions de REL permettent d'identifier des *resource* d'un business particulier ou d'applications techniques (plan d'un architecte par exemple).

```
User XXX vous avez le droit de visionner mon film n°YYY.
```

*L'entité condition*

Une *condition* spécifie les conditions, les termes et les obligations sous lesquelles les droits peuvent être exercés. Un exemple de condition simple : un intervalle de temps durant lequel le droit peut être exercé. Mais la condition pour pouvoir exercer un droit peut être aussi l'acquisition préalable d'un autre droit sur la *resource*. Des extention de REL permettent d'établir des conditions appropriées à des modèles d'utilisation.

```
User XXX vous avez le droit de visionner mon film n°YYY, si
vous payez 5€
```

*Les extensions et les profiles*

*MPEG* reconnaît que pour certaines industries et communautés d'utilisateurs, des modifications doivent être apportées au langage pour qu'il puisse répondre à leurs attentes. C'est donc pour cela que *MPEG* a mis en place des procédures d'extension et de profil du langage.

Les extensions permettent aux utilisateurs de définir des éléments de langage spécifiques à leur besoins. Ceci inclut la définition de nouvelles actions aux activités (servant à définir des entités *right*). *MPEG* a aussi développé un dictionnaire des droits (Rights Data Dictionary) pour éviter les problèmes d'ambiguïté lié à l'interopérabilité avec les nouvelles actions et activités.

Les profiles permettent aux utilisateur de ne sélectionner qu'une partie du langage, uniquement ce dont ils ont besoin pour leurs applications. Ceci permet d'alléger la charge utile des Digital Item et ainsi donc que de réduire les exigences de puissance de calcul de leur appareil *MPEG*.

### 8.4.6 MPEG-21 partie 6 : Rigts Data Dictionary

Le RDD[12] est un ensemble de termes clairs, cohérents, structurés et identifiés de manière unique pour permettre la génération d'expressions de droits en REL.

[12] Rigts Data Dictionary

Le RDD définit une seule signification par terme représenté par un RDD name ou headword. Il reconnaît aussi les définitions d'autres headwords définis par d'autre autorités et les intègre après adaptation. Le RDD tolère aussi différentes définitions pour le même headword, pour autant que ces définitions viennent d'autorités différentes.

Les seules définitions légales que contient le RDD proviennent d'autres autorités qui peuvent être transcrites dans le RDD. Dès lors les seules définitions qui sont directement ajoutées dans le dictionnaires sont celles qui ne concernent pas des droits de propriété intellectuelle ou autres règles légales.

En plus de fournir des descriptions de termes utilisés pour REL, le RDD permet le passage d'une terminologie de droits à une autre (venant de deux autorités différentes) de manière automatisée ou semi automatisée.

### 8.4.7 MPEG-21 partie 7 : Digital Item Adaptation

Un des buts principaux de *MPEG-21* est d'arriver à fournir un accès transparent à des contenus multimédia complexes distribués, sans que l'utilisateur n'ait à s'inquiéter des problèmes liés aux réseaux, aux terminaux, ….

Pour atteindre cet objectif, il est nécessaire que les DI puissent être adaptés afin de satisfaire à des exigences de transmission, de stockage et de consommation. La DIA[13] spécifie donc la syntaxe et la sémantique d'outils qui pourraient être utilisés pour l'adaptation des DI, assurant ainsi que les contraintes fixées par l'utilisateur sont bien respectées.

Comme le montre l'architecture conceptuelle illustrée à la figure 8.5 page 132, le DI va passer à travers un moteur un d'adaptation de descriptor et moteur d'adaptation de resource qui vont produire le DI adapté. Il faut noter que ces moteurs d'adaptation ne sont pas des outils normalisés. Cependant les mécanismes de description ainsi que les mécanismes indépendants du format de sortie permettant l'adaptation de DI en terme de resource, de descriptor et de QoS[14], font partie des exigences de cette septième partie de *MPEG-21*.

Les outils de DIA sont repartis en huit catégories principales :

– Les outils basés sur la description de l'environnement d'utilisation qui inclut les caractéristiques de l'utilisateur (préférence à des ressources particulières, à une présentation spéciale des ressources, de l'appareillage audio-visuel, …),

[13] Digital Item Adaptation
[14] Quality of Service : Qualité de Service

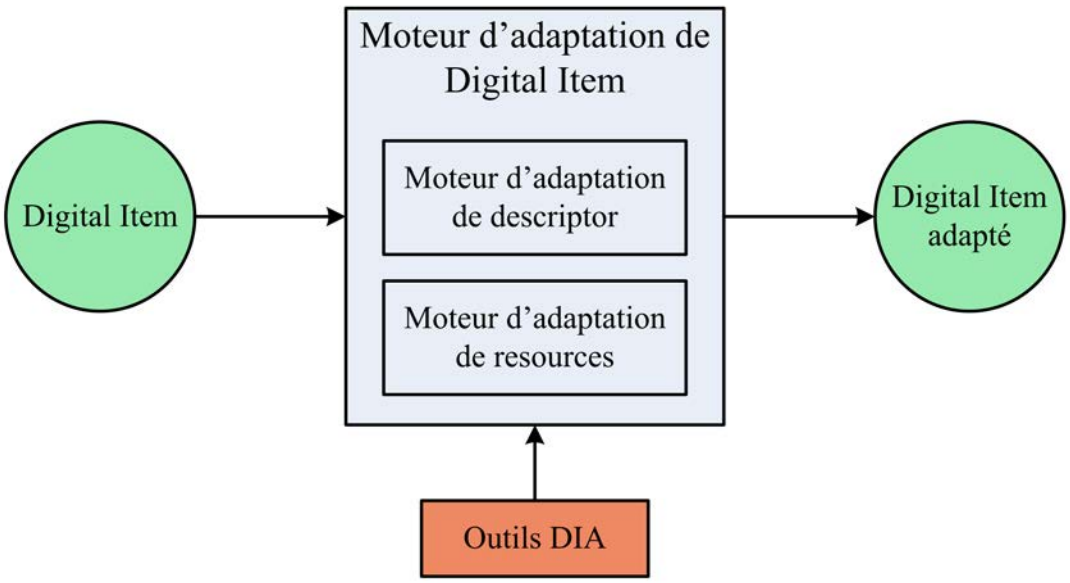


FIG. 8.5 – *Architecture conceptuelle d'un adaptateur de DI*

les capacités du terminal de l'utilisateur (capacités de codage, décodage, hardware, software, ...), les caractéristiques du réseau (bande passante, caractéristiques des délais et des erreurs, ...) et les caractéristiques de l'environnement naturel de l'utilisateur (lieu, période de la journée, caractéristiques de l'environnement audio-visuel, ...).

– Les outils basés sur la description syntaxique du flux de données, c'est-à-dire la syntaxe binaire de la ressource. Ces outils permettent de grouper les bits, ajouter des en-têtes aux groupes de bits, tronquer certain bits, ....
– Les outils de type BSD[15] Link permettant une quantité d'adaptations basées sur des outils spécifiés par *MPEG-21*, *MPEG-21*DID et *MPEG-7*MDS.
– Les outils de qualité de service des terminaux et des réseaux. Ces outils essayent de satisfaire à la fois les contraintes de QoS des terminaux et réseaux et la qualité de la ressource résultant de l'adaptation. Ils permettent de développer des stratégies d'adaptation qui permettent de choisir les meilleurs paramètres pour l'adaptation.
– Les outils basés sur la description de contraintes universelles qui se basent sur des contraintes de limitation et d'optimisation des adaptations.
– Les outils adaptant les métadonnées. Ils fournissent des informations permettant de réduire la complexité de l'adaptation des métadonnées contenues dans les DI.

[15] Bitstream Syntax Description

– Les outils de mobility session. Ces outils transmettent les informations d'état de configuration d'un terminal à un autre. Ceci permet à un terminal de recevoir un DI de façon adaptée d'un autre terminal.
– Les outils de configuration DIA qui fournissent les informations requises pour la configuration des moteurs d'adaptation de DI.

### 8.4.8 MPEG-21 partie 8 : Reference Software

Cette partie décrit les programmes de références implémentant les clauses normalisées des autres parties de *MPEG-21*. L'information fournie sert à déterminer quels sont les modules de programme de référence disponibles pour les différentes parties de *MPEG-21*, en comprendre les fonctionnalités et comment les utiliser.

En plus des programmes de références, on peut trouver aussi des descriptions de programmes utilitaires utilisant ces programmes de références. Ces utilitaires peuvent aider à comprendre comment utiliser les programmes de références, mais aussi apporter des améliorations aux différentes parties de *MPEG-21*.

On y retrouve principalement des programmes de références et des utilitaires pour les parties 2, 3, 6 et 7 de *MPEG-21*.

### 8.4.9 MPEG-21 partie 9 : File Format

Un DI *MPEG-21* est une collection complexe de données. On peut y retrouver des images fixes, des vidéo, des métadonnées, .... On y retrouve ces données soit sous forme textuelle (données XML par exemple) ou sous forme binaire (une images par exemple). C'est pour cela que *MPEG-21* intègre certains concepts de *MPEG-4* afin de proposer des fichiers "universels". Par exemple pour un ficher contenant à la fois des données *MPEG-4* et *MPEG-21*, un lecteur *MPEG-4* pourra lire les données *MPEG-4*, tandis que le lecteur *MPEG-21* lira les données *MPEG-21*.

On a donc besoin d'un container physique pour ces différent types de données fournies par le File Format de *MPEG-21*. Un exemple de format de fichier *MPEG-21* est illustré à la figure 8.6 page 134.

### 8.4.10 MPEG-21 partie 10 : Digital Item Processing

Le DID Language est un langage statique XML de déclaration. A cause de cela un DI est vu comme un simple container combinant des ressources et des métadonnées

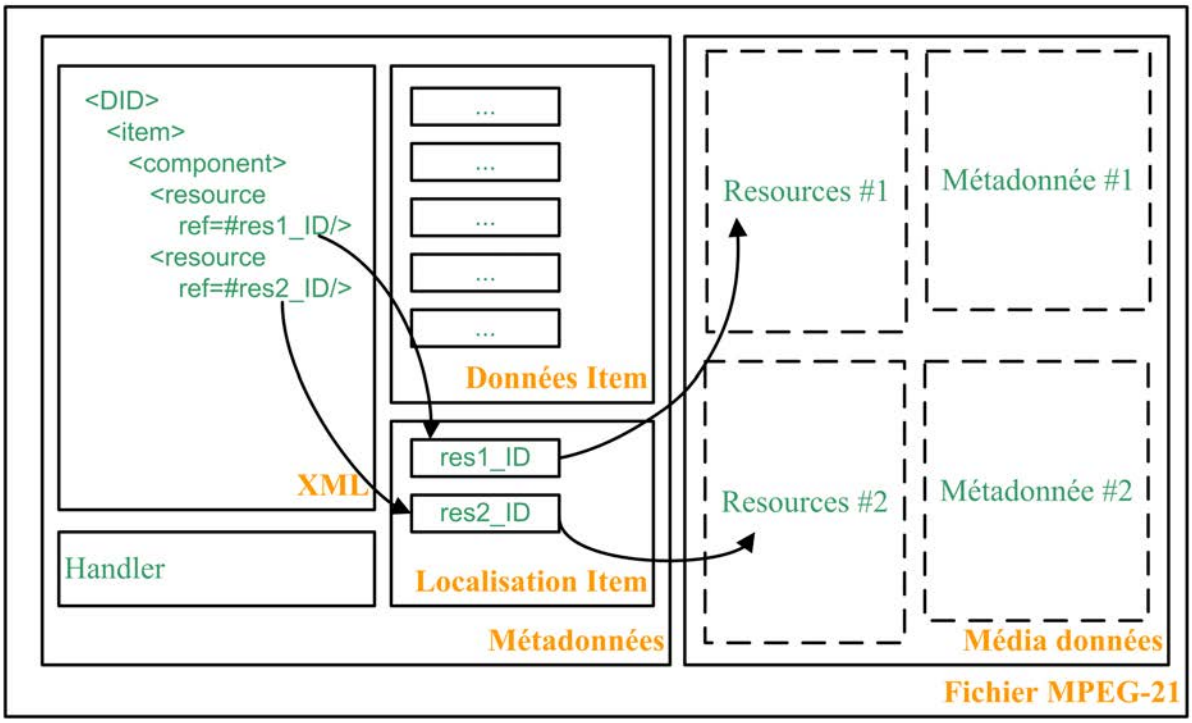


FIG. 8.6 – *Exemple de format de fichier MPEG-21*

et définissant les relations entre eux. Et donc l'auteur d'un DI n'a aucun contrôle sur la manière dont va être traité le DI car cela dépend entièrement du terminal *MPEG-21* de l'utilisateur. A partir de ce moment il est possible que deux terminaux *MPEG-21* ne traitent pas le même DI de la même façon, ce qui est perçu comme un manque d'interopérabilité.

Le principe de l'architecture du DIP[16] est assez simple. Un élément de la DID contient une liste de méthodes applicables au DI et ces méthodes sont présentées au User. Ensuite celui-ci peut choisir les méthodes qui seront alors exécutées par le DIP Engine. Par exemple le DIP permet à l'auteur d'un DI de choisir les méthodes de traitement qui peuvent être appliquées au DI.

Les méthodes applicables à un DI, les DIMs[17], sont écrites avec le Digital Item Method Language (DIML) basé sur ISO/IEC 16262 :2002 (ECMAScript language).

[16] Digital Item Processing
[17] Digital Item Methodes

### 8.4.11 MPEG-21 partie 11 : Evaluation Methods for Persistent Association Technologies

Ce rapport technique permet d'évaluer les technologies persistantes utilisées mais ne contient pas de comportement normalisé. Cette partie fournit en fait une méthodologie commune permettant d'évaluer les technologies plutôt qu'une standardisation des technologies elles-mêmes. Cette partie fournit des tests appropriés des technologies qui donneront des estimations de performance comme si ces technologies étaient utilisées dans le "monde réel", et permet aussi de comparer ses résultats avec ceux d'autres tests effectués en utilisant la même méthodologie. Ce rapport se concentre sur deux technologies liées à la partie audio : les watermarks et les fingerprints. Cependant il est prévu que ce rapport s'étoffe dans le futur afin de couvrir d'autres types de média.

### 8.4.12 MPEG-21 partie 12 : Test Bed for MPEG-21 Resource Delivery

Un "test bed" est un environnement d'exécution configuré pour pouvoir effectuer des tests sur des composants logiques ou physiques éloignés. Cette partie spécifie des "test bed" conçus pour évaluer les performance de Scalable Video Codec pour les applications de streaming ainsi que pour évaluer les technologies de "delivry resource" utilisant des réseaux non fiables. Le protocole de streaming utilisé est basé sur RTSP et RTP.

### 8.4.13 MPEG-21 partie 13 : Scalable Video Coding

Ce projet est en fait une amélioration du codage hiérarchique vidéo de *MPEG-2*. La partie Scalable Video Coding ou codage vidéo hiérarchique fut abandonnée l'an passé, dès que les premiers tests ont montré que les résultats obtenus étaient étaient assez proches de ceux obtenus avec la partie 10 de *MPEG-4* : AVC/H.264. Cependant le projet ne fut pas complètement abandonné, c'est ainsi que cette partie fut intégrée dans la partie 10 de *MPEG-21* : Digital Item Adaptation.

### 8.4.14 MPEG-21 partie 14 : Conformance Testing

Le modèle de Conformance Testing est un modèle fournissant des modèles de test de conformité des spécifications de chacune des parties de *MPEG-21*, illustré à la

figure 8.7 page 136. Son but est aussi de fournir des tests d'intégrations concernant les spécifications du standard en entier.

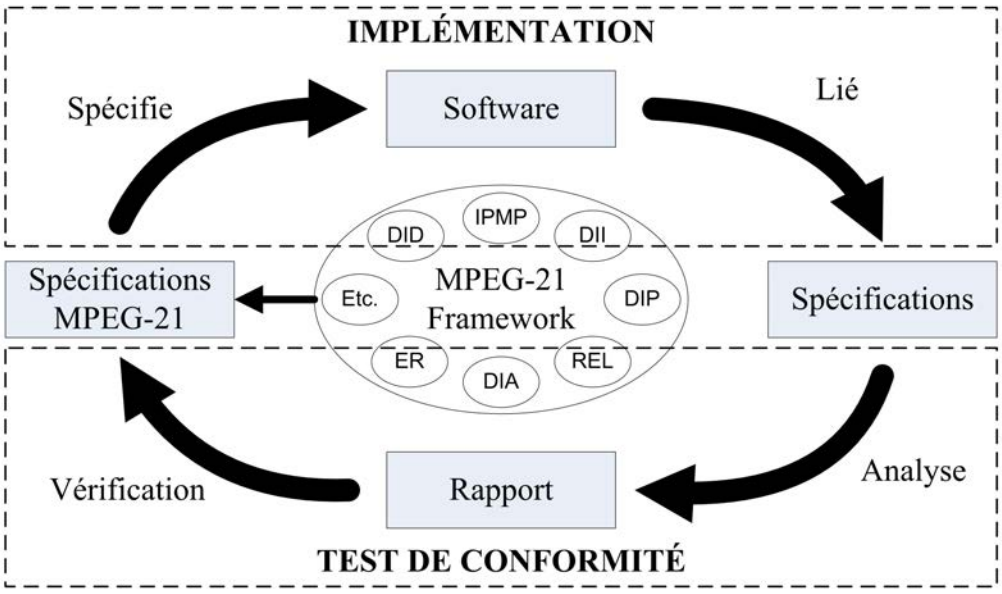


FIG. 8.7 – *Modèle de test de conformance*

### 8.4.15 MPEG-21 partie 15 : Event Reporting

L'Event Reporting fournit des moyens standardisés pour échanger de l'information à propos de certains événements qui peuvent se produire entre Users et Peers[18] (appareils ou applications capables de traiter un DI).

Une Event Report Request[19] (ER-R) est utilisée pour définir les conditions (prédicats) sous lesquels un Event est censé se déclencher. L'Event repris dans une ER-R déclenchera la création d'un Event Report (ER) contenant les informations décrivant l'Event, comme spécifié dans l'ER-R. Chaque ER-R doit comprendre au moins : une description de l'Event, la syntaxe ou le format de l'ER, celui à qui l'ER doit être envoyé et les paramètres d'envoi de l'ER (cryptage, mécnisme de transport, ...).

Prenons l'exemple de la gestion de ressources faisant l'objet d'un copyright. Le fournisseur mettant à disposition des DIs via téléchargement spécifiera dans une

[18]Peer to Peer désigne un modèle de réseau informatique dont les éléments (les noeuds) ne jouent pas exclusivement les rôles de client ou de serveur mais fonctionnent des deux façons, en étant à la fois clients et serveurs des autres nœuds de ces réseaux, contrairement aux systèmes de type client-serveur.

[19] requête de rapport d'événement

ER-R que chaque fois qu'une ressource d'un DI sera visionnée, il recevra un ER lui permettant de demander sa contre-partie financière accordée par les droits associés à la ressource visionnée. Une fois qu'il aura visionné sa ressource, le terminal de l'utilisateur générera un ER qui sera envoyé au détenteur des droits. Cet ER contiendra des informations à propos du DI, de la ressource et des conditions sous lesquelles la ressource peut être visionnée.

L'Event Reporting sert principalement à standardiser des métriques et des interfaces pour tous les événements qui peuvent être rapportés, ainsi qu'à fournir des moyens pour capturer et contenir ces différentes métriques et interfaces faisant références à des DIs, Peers, environnements, processus, transactions et Users tous identifiés.

L'Event Reporting s'occupe uniquement des Events liés directement ou indirectement à des DIs ou à des Peers. Mais il est aussi lié aux événements entre Peers et non aux Events internes d'un seul Peer.

Voici une liste non exhaustive d'exemples où les Event Report sont utilisés :

- Les usage reports (rapports d'utilisation) concernant :
  - Le copyright
  - Les performances
  - Les copies
  - Les information marketing
- Les technical reports (rapports techniques) concernant :
  - L'utilisation ou la disponibilité d'une bande passante
  - La congestion du réseau
  - La charge
- Les financial reports (rapports financiers) concernant :
  - L'acceptation et la délivrance de licence
  - La preuve d'acceptation

### 8.4.16 MPEG-21 partie 16 : Binary Format

Cette partie spécifie le format binaire de ISO/IEC 21000 permettant de produire efficacement des descriptions basé sur XML respectant les spécification des autres parties de *MPEG-21*. Ceci permet de stocker ou d'échanger de manière efficace de description *MPEG-21*.

### 8.4.17 MPEG-21 partie 17 : Fragment Identification for MPEG Ressources

Cette partie spécifie une syntaxe normalisée pour les URI identifiant des fragments, permettent ainsi de localiser n'importe quelles partie d'une ressource. La syntaxe de ces identifiants est basée sur la W3C XPointer Framework Recommendation. On retrouve ces identifiants après le caractères # de l'URI.

```
http://example.com/myFile.mp21#ffp(track_ID=101)*mp(/~time('npt','50'))
```

Cela permet d'adresser :
– une séquence de bits d'une ressource
– une ou plusieurs unité(s)logique(s) d'un modèle logique
– des Items ou des Tracks d'une ISO Base Media File
– un intervalle spatial, de temps ou spatio-temporel

### 8.4.18 MPEG-21 partie 18 : Digital Item Streaming

Cette partie donne les spécifications d'outil pour le Digital Item Streaming. Le premier outil est le Bitstream Binding Language décrivant la manière de transformer les Digital Item (DID, métadonnées, ressources) pour la diffusion sur des canaux de type *MPEG-2* Transport Streams ou Real Time Protocol.

## 8.5 Références

Afin de réaliser ce chapitre nous nous sommes inspirés des documents suivants : [AD05], [Con04], [KNL04], [Dev06], [Var03], [DJ05], [ISO05b], [ISO05a], [ISO05c], [DJ05], [ISO03], [ISO02a].

# Quatrième partie

# Multimedia Document Manager

CHAPITRE 9

# Description du Multimedia Document Manager

Afin de mettre en pratique les normes et standards étudiés dans les chapitres précédents, il nous a parut évident de concevoir une architecture permettant l'indexation et la recherche de documents multimédia (Le Multimedia Document Manager) . Cette architecture tentera d'utiliser au mieux les technologies vues dans les trois premiers chapitres (*XML*, *MPEG-7*,...). Les trois principales fonctionnalités[1] de l'architecture qui seront présentées sont :

- la description sémantique et visuelle des documents multimédia ;
- le stockage et l'indexation de ces descripteurs ;
- la recherche de document ;

Nous étudierons à travers cette partie deux solutions différentes. La première sera construite autour du moteur d'indexation *Lucene* du projet Jakarta Apache tandis que la deuxième sera réalisé en "full-*XML*". A la suite de cette description générale, nous analyserons étape par étape la mise en place de ces deux solutions et en critiquerons les avantages et inconvénients.

Les exemples de document *MPEG-7* que vous trouverez dans ce chapitre et les suivants seront des exemples non normatifs. L'accès aux grammaires de référence *MPEG-7* n'est disponible que sous une licence commerciale.

[1] Nous n'étudierons pas la manière dont les documents multimédia sont stockés, ceci étant hors du cadre du sujet du mémoire.

La figure 9.1 illustre le fonctionnement général de production de description sémantique d'un document multimédia.

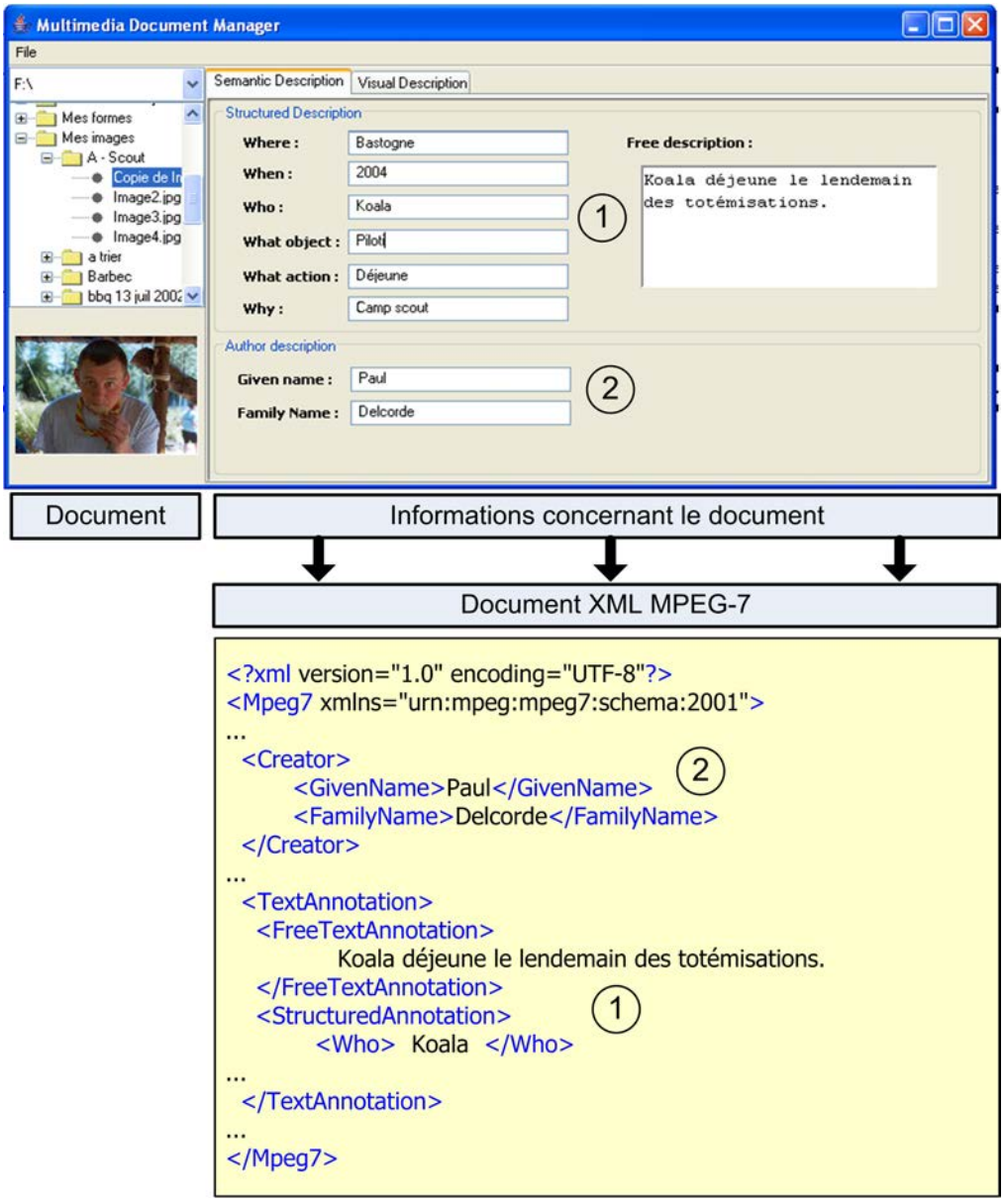


FIG. 9.1 – *Transformation des informations en descripteur* MPEG-7

Cette description est réalisée par l'intermédiaire d'un formulaire de description que l'utilisateur doit remplir manuellement. Ces informations décrivent le "contenant" : son auteur et le lieu de la prise de vue par exemple et le "contenu " : Quels objets voit-on sur le document ? ; Qui est présent sur le document ?,...Une fois les champs du formulaire complétés, le système enregistre les différentes informations sous forme d'un document *XML MPEG-7*. En effet, le *MPEG-7* permet la retranscription des informations des documents à indexer sous un format lisible par l'homme et assure l'interopérabilité de nos deux systèmes. De plus *MPEG-7* est un standard reconnu par les milieux professionnels de l'informatique et de l'audiovisuel et qui est de plus un sous-ensemble de la famille *XML*. Cette reconnaissance et cette appartenance à la famille *XML* implique l'existence d'outils libres de droit facilitant le traitement de ces données. On peut citer en autre des API[2] de manipulation des données *XML* (JDOM[3],SAX[4],...) ; de validation de fichier *XML* par rapport à leur grammaire ou encore de production automatique de descripteurs visuels conformes au standard *MPEG-7*. Le description sémantique se réalise de la même façon dans nos deux solutions.

Les descripteurs visuels seront quant à eux calculés de manière transparente pour l'utilisateur. Dans le cas de la solution "full-*XML*", le document *MPEG-7* sera complété automatiquement par les descripteurs *MPEG-7* extraits de la photo par une analyse de l'image. La figure 9.2 nous montre un descripteur *MPEG-7* de couleur dominante de l'image illustré à la figure 9.1. Dans le cas de la solution Lucene, les descripteurs visuels seront directement calculés et indexés sans être décrit en *MPEG-7* grâce à l'API *LIRE*[5].

```
<VisualDescriptor xsi:type="DominantColorType">
 <SpatialCoherency>0</SpatialCoherency>
 <Value>
  <Percentage>3</Percentage>
  <Index>5 3 2</Index>
 </Value>
 <Value>
  <Percentage>2</Percentage>
  <Index>14 19 22</Index>
 </Value>
 ...
</VisualDescriptor>
```

FIG. 9.2 – *Descripteur* MPEG-7 *de couleur dominante calculé par l'API Caliph*

[2]API : Application Programming Interface
[3]JDOM : Java Document Object Model
[4]SAX : Simple Api for XML
[5]LIRE : Lucene Image REtrieval

Une fois la production de la description du document multimédia terminée, celle-ci est stockée et indexée à l'aide d'un moteur d'indexation. Le stockage de ces descripteurs se fera de manière différente dans les deux solutions retenues. L'architecture "full-*XML*" utilisera une base de données *XML* Native tandis que celle basée sur Lucene stockera ses descriptions sémantiques dans un système de fichiers. L'indexation quant à elle est prise en charge par la base de donnée *XML* ou par Lucene (sous forme d'un fichier Lucene d'indexation) suivant le système. .Les figures 9.3 et 9.4 illustrent les différentes étapes des chaînes de traitement "Production-Stockage-Indexation" des deux systèmes.

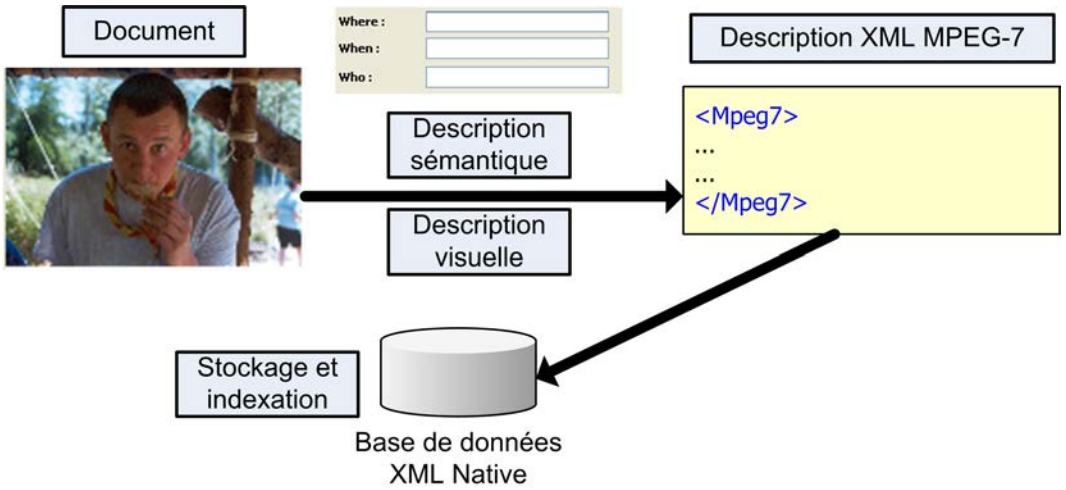


FIG. 9.3 – *"Production-Stockage-Indexation" de la solution "full-*XML*"*

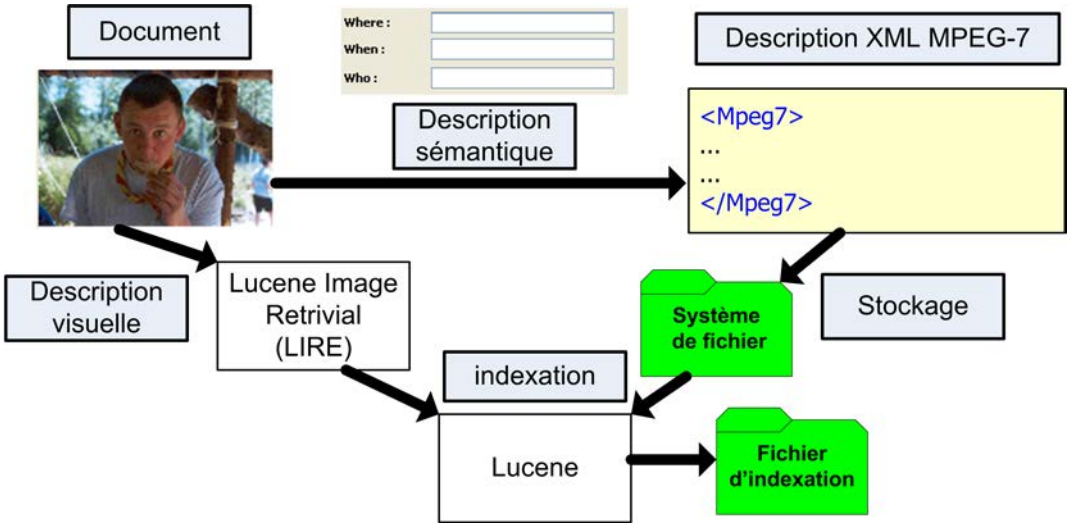


FIG. 9.4 – *"Production-Stockage-Indexation" de la solution Lucene*

La recherche d'un document multimédia se fait grâce à un moteur de recherche et par l'intermédiaire de requêtes. Le langage de requête utilisé pour la solution "full-*XML*" est le *XQuery*[6]. Lucene quant à lui propose se propre syntaxe de requête. Il est important de préciser que, lors d'une recherche de document multimédia, les moteurs de recherche des deux solutions proposées ne délivrent pas directement les documents multimédia recherchés mais bien les descriptions *XML MPEG-7* leur correspondant. Les descriptions devront donc être analysées afin de permettre l'affichage des documents multimédia et de leurs informations.

Pour la recherche sémantique, les requêtes seront exprimés au travers d'un formulaire d'interrogation encodé manuellement par l'utilisateur. Le moteur de recherche analysera ensuite les différentes requêtes et cherchera les descripteurs indexés correspondant au mieux aux critères de recherche de l'utilisateur. Sur base de ces descripteurs, il sera possible pour l'utilisateur de visionner les documents multimédia correspondant au mieux à ses attentes. La figure 9.5 illustre la chaîne de traitement "Recherche sémantique-Présentions" des solutions "full-*XML*" (flèches en rouge) et Lucene (flèches en bleu).

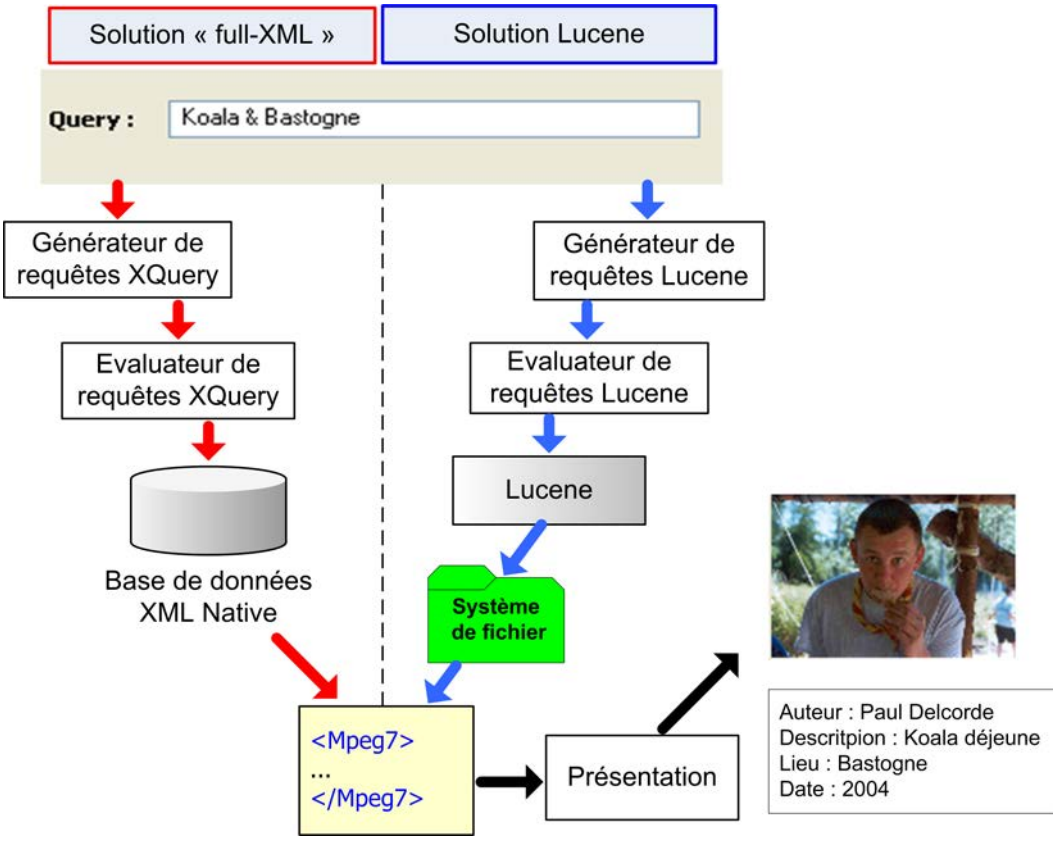


FIG. 9.5 – *Chaîne de traitement "Recherche sémantique-Présentation"*

[6]XQuery : Xml Query

Pour la recherche par similarité, l'utilisateur devra spécifier une image dont seront calculé ses différents descripteurs visuels. Les descripteurs visuels de l'image de recherche ainsi calculés, le moteur de recherche parcourra les différents ensembles descripteurs visuels indexés. Chaque ensemble de descripteurs correspond à une image répertoriée dans le système. Une fonction de similarité sera ensuite calculée entre les différents ensembles de descripteurs indexés et ceux de l' image de recherche. Les images dont le coefficient de similarité est estimé suffisant seront donc proposées à l'utilisateur. La figure 9.6 illustre la chaîne de traitement "Recherche par similarité-Présentation" de la solution "full-xml".

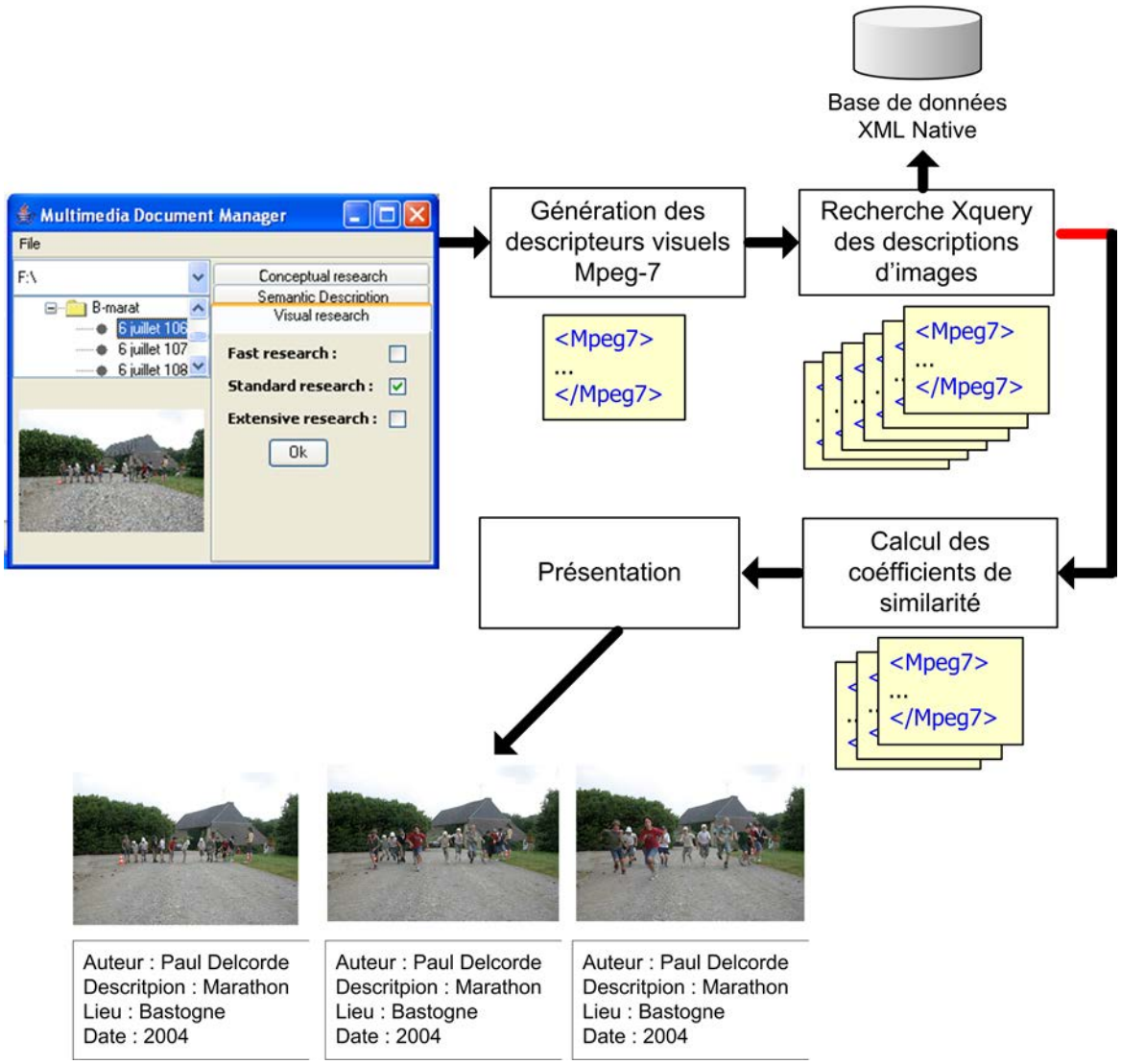


FIG. 9.6 – *"Recherche par similarité-Présentation" de la solution "full-*XML*"*

Lors de l'indexation d'une image dans le système Lucene, seuls les descripteurs sémantiques sont enregistrés et stockés au format *MPEG-7* dans un système de fichiers. Les descripteurs visuels sont quant à eux directement stockés dans le fichier Lucene d'indexation. C'est l'API *LIRE* qui va se charger du traitement de l'image en entrée et de la recherche par similarité avec les images indexées dans le système. Une fois les descriptions visuelles des images pertinentes trouvées, le système retrouvera leurs descriptions *MPEG-7* sémantiques et affichera le résultat à l'utilisateur. La figure 9.7 illustre la chaîne de traitement "Recherche par similarité-Présentation" de la solution Lucene.

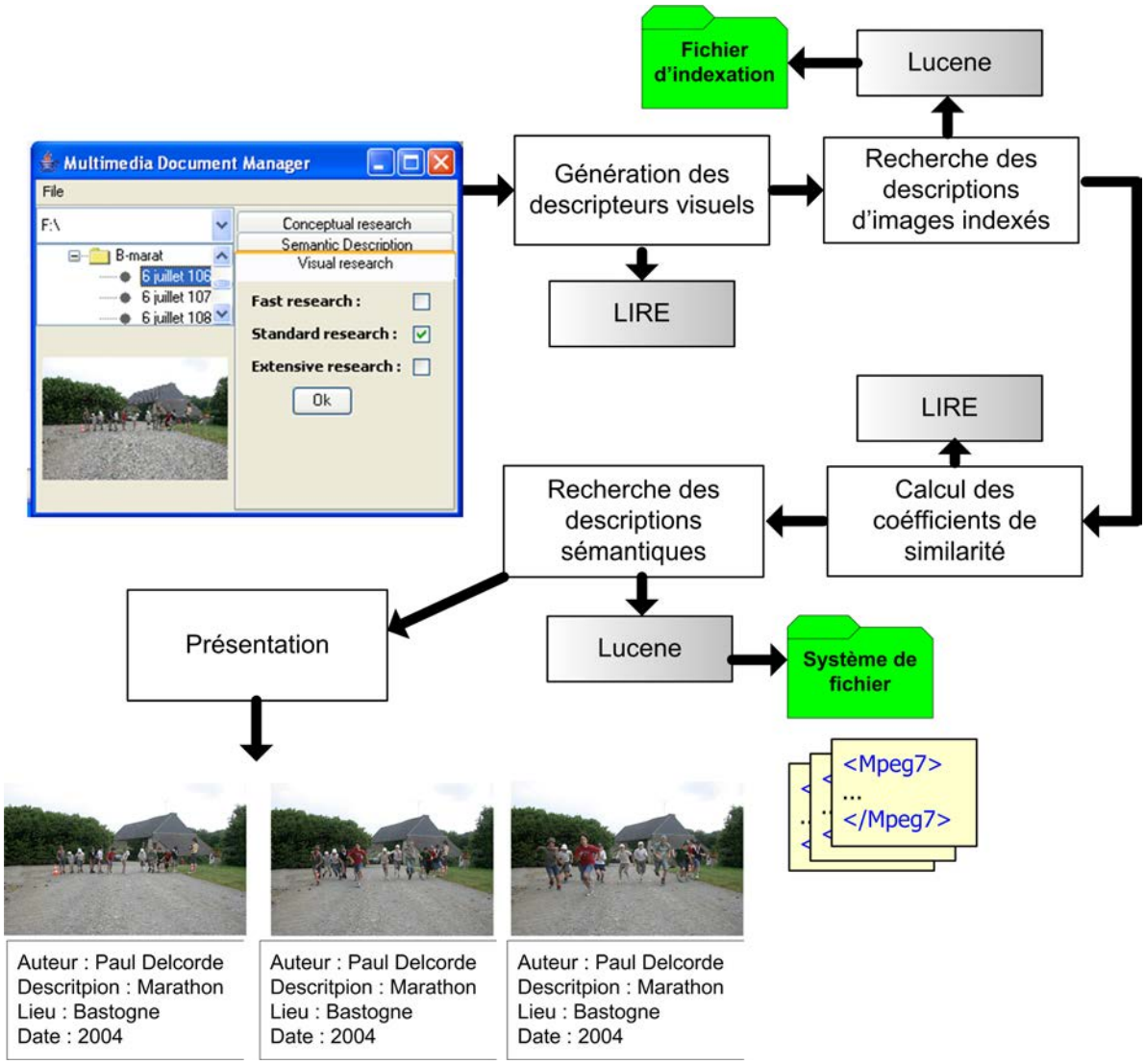


FIG. 9.7 – *"Recherche par similarité-Présentation" de la solution Lucene*

CHAPITRE 10

# La production de descripteurs

Comme nous l'avons mentionné à plusieurs reprises, le quantité de documents multimédia disponible est gigantesque et ne cesse d'augmenter. Sans procédé "universel" de description, la recherche d'une donnée multimédia peut s'avérer chaotique. En effet, les informations concernant une donnée stockée peuvent prendre des significations variables, voire incompatibles. Un objet peut être perçu de différentes manières par plusieurs individus. Chaque règle de description reflète un point de vue porté sur un objet dans un contexte particulier. Ce point de vue dépend, par exemple, de la culture de chacun et des objectifs associés à la numérisation d'une information. En utilisant des métadonnées, il est possible de pallier ce problème.

Au sens le plus général, en informatique, une métadonnée est un mot ou une locution caractérisant l'information contenue dans un élément afin d'en faciliter l'utilisation et la recherche. Citons comme exemple, les "méta balises" HTML[1] qui sont des balises spéciales situées dans l'en-tête d'un document HTML permettant notamment de fournir des informations permettant aux moteurs de recherche d'indexer les page WEB. Un site de location de voiture devrait, pour être référencé correctement, contenir les métadonnées illustré à la figure 10.1.

[1]HTML : HyperText Markup Language

```
<html>
 <head>
  <meta name="keywords" content="location de voiture,location de véhicule,location,... ">
 </head>
 ...
</html>
```

FIG. 10.1 – *Exemple de méta-balise HTML*

Mais l'utilisation seule de métadonnées n'est pas suffisante pour résoudre les problèmes de l'indexation et de la recherche des documents multimédia. Encore faut-il que ces métadonnées reposent sur des standards. Un standard de métadonnées doit fournir un langage permettant de définir la syntaxe des métadonnées ainsi que des principes de notation décrivant leur mode de représentation. Nous avons dans les premiers chapitres de ce mémoire présenter plusieurs standards de représention de métadonnées tels que le *Dublin Core* et le *MARC* pour le monde bibliothécaire ou encore le *MPEG-7* pour le monde du multimédia.

Nos deux systèmes s'axant sur l'indexation et la recherche de documents audiovisuels, nous avons choisi d'opter pour le standard *MPEG-7* pour la réalisation de nos descripteurs. Plutôt que de métadonnée, nous parlerons maintenant de descripteurs renfermant des informations concernant des document multimédia. Chaque descripteur *MPEG-7* représente un moyen d'identifier une caractéristique d'un document multimédia. Ces caractéristiques peuvent être regroupées en deux niveaux : les caractéristiques de bas et de haut niveau[2] :

1. Les caractéristiques de bas niveau décrivent les caractéristiques comme la couleur, la texture, la taille ou encore les formes reconnaissables. C'est à ce niveau qu'on retrouve les descripteurs visuels utilisés dans nos deux solutions.
2. Les caractéristiques de haut niveau visent à fournir une description de ce que contient une image. Il peut s'agir de personnages, de lieux, d'actions ou encore d'interaction. On tente ici d'exprimer et de modéliser l'histoire véhiculée par le contenu de l'image. On peut encore retrouver à ce niveau des informations concernant le contexte tels que l'auteur ou la date de la prise de vue. C'est ce que nous appelons dans nos solutions, les descripteurs sémantiques.

Selon le niveau de caractéristiques ciblé, il est possible ou non d'extraire automatiquement les informations recherchées. Ainsi pour la plupart des descripteurs visuels, l'extraction d'informations les concernant est automatisable. Pour les descripteurs sémantiques, une assistance humaine est toujours indispensable.

[2]Étant donné que *MPEG-7* décrit un nombre considérable de descripteur, il nous est impossible de les étudier tous en détail, nous nous focaliserons donc sur les descripteurs d'image fixe.

## 10.1 Les descripteurs sémantiques

Pour les analyseurs d'image, l'indexation correspond avant tout à une recherche par l'exemple. Cela consiste à la recherche d'images visuellement similaires à une image donnée en entrée par un utilisateur. Cette similarité se fonde sur des caractéristiques tels que la forme, la couleur, la texture ou encore la détection de visage. Les traiteurs d'image se limitent donc à la description de l'apparence d'une image. En effet, le passage de l'apparence au concept (c'est-à-dire au niveau sémantique) pose des problèmes bien plus complexes. Dans le domaine de la description sémantique, les capacités de traitement informatique sont encore bien inférieures aux capacités humaines. Un système informatique d'indexation, bien que disposant d'une base de connaissances étendue, ne peut pas rivaliser avec les années d'apprentissage d'un être humain. Que ce soit pour l'interprétation du sens ou de l'action d'une prise du vue, le documentaliste a encore de beaux jours devant lui...

### 10.1.1 L'encodeur sémantique

Comme nous l'avons montré au chapitre précédant, nos deux solutions comportent un encodeur sémantique. Une description sémantique est réalisé manuellement par un utilisateur désirant indexer une image. Une fois les champs remplis, l'encodeur analyse les caractéristiques entrées et les transforme en descripteurs *XML MPEG-7*.

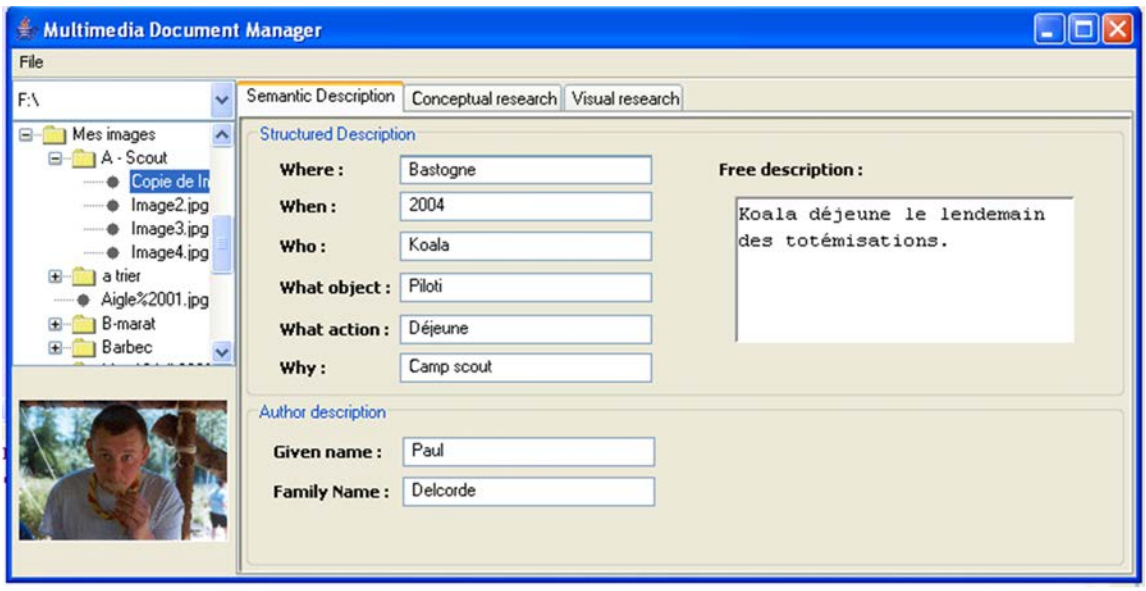


FIG. 10.2 – *L'encodeur sémantique*

Le formulaire d'encodage se compose de deux parties : la première que nous appellerons `Annotation` se réfère aux informations sémantiques de l'image à indexer ; la deuxième nommé `Creator` permet d'identifier le créateur de la prise de vue.

Le formulaire `Annotation` comporte les champs suivant :

- `Where` permettant d'indiquer le lieu de la prise de vue ;
  (Dans l'exemple *Bastogne*)
- `When` permettant de préciser le date de la prise de vue ;
  (Dans l'exemple *2004*)
- `Who` permettant d'indiquer le ou les personnages présents sur la prise de vue ;
  (Dans l'exemple *Koala*)
- `What object` permettant d'indiquer les objets présent sur la prise de vue ;
  (Dans l'exemple *Pilotis*)
- `What action` permettant de décrire l'action se déroulant sur la photo ;
  (Dans l'exemple *Déjeune*)
- `Why` permettant d'indiquer les raisons pour lesquelles la photo a été prise ;
  (Dans l'exemple *Camp scout*)
- `Free description` permettant d'insérer des informations supplémentaires.
  (Dans l'exemple *Koala déjeune le lendemain des totémisations*)

Le formulaire `Creator` donne quant à lui la possibilité à l'utilisateur d'encoder le nom (`Family name`) et le prénom (`Given name`).

### 10.1.2 Grammaire des descripteurs sémantiques

La réalisation d'une grammaire s'est avérée nécessaire afin de spécifier la structure et la sémantique des différents descripteurs et schémas de description formant la description sémantique. Cette grammaire se divise en trois parties : la description de l'image, la description de son créateur et la localisation du média (bien que ce ne soit pas une caractéristique sémantique à proprement parler). Pour des raisons de licence, cette grammaire a été réalisé autour du langage *XML* Schema et non en *DDL*.

**A - La description sémantique de l'image**

Dans cette partie on retrouve :

- le descripteur `Term`, illustré à la figure 10.3, qui sera utilisé dans les schémas de description `TextAnnotation` et `Creator` ;
- le schéma de description `TextAnnotation` illustré à la figure 10.4.

```
<xsd:simpleType name="Term" type="xsd:string" />
```

FIG. 10.3 – *Le descripteur* `Term`

```
<xsd:element name = "TextAnnotation" type="TextAnnotationType"/>

<xsd:complexType name="TextAnnotationType" >
 <xsd:sequence>
  <xsd:element name="FreeTextAnnotation"     type="xsd:Term"  minOccurs="0"/>
  <xsd:element name="StructuredAnnotation"   type="xsd:StructuredAnnotationType"  minOccurs="0"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="StructuredAnnotationType" >
 <xsd:sequence>
  <xsd:element name="Where"          type="xsd:Term"  minOccurs="0"/>
  <xsd:element name="When"           type="xsd:Term"  minOccurs="0"/>
  <xsd:element name="Who"            type="xsd:Term"  minOccurs="0"/>
  <xsd:element name="WhatObject"     type="xsd:Term"  minOccurs="0"/>
  <xsd:element name="WhatAction"     type="xsd:Term"  minOccurs="0"/>
  <xsd:element name="Why"            type="xsd:Term"  minOccurs="0"/>
 </xsd:sequence>
</xsd:complexType>
```

FIG. 10.4 – *Le schéma de description* `TextAnnotation`

### B - La description sémantique de l'auteur

Dans cette partie on retrouve :

– le descripteur `Term` ;
– le schéma de description `Creator` illustré à la figure 10.5.

```
<xsd:element name = "Creator" type="CreatorType"/>

<xsd:complexType name="CreatorType" >
 <xsd:sequence>
  <xsd:element name="GivenName"     type="xsd:Term"  minOccurs="0"/>
  <xsd:element name="FamilyName"    type="xsd:Term"  minOccurs="0"/>
 </xsd:sequence>
</xsd:complexType>
```

FIG. 10.5 – *Le schéma de description* `Creator`

### C - La localisation du média

Dans cette partie on retrouve :

– le descripteur `Term` ;
– le schéma de description `MediaLocator` illustré à la figure 10.6.

```
<xsd:element name = "MediaLocator" type="MediaLocatorType"/>

<xsd:complexType name="MediaLocatorType" >
 <xsd:sequence>
  <xsd:element name="MediaUri"      type="xsd:Term"  minOccurs="0"/>
 </xsd:sequence>
</xsd:complexType>
```

FIG. 10.6 – *Le schéma de description* `MediaLocator`

La figure 10.7 montre un exemple complet conforme à la grammaire d'une description sémantique.


FIG. 10.7 – *Exemple complet d'une description sémantique*

### 10.1.3 Appréciation des descripteurs sémantiques

Comme nous venons de le voir, l'indexation sémantique est très puissante vu qu'elle est très proche de l'utilisateur et qu'elle peut décrire presque tous les aspects du contenu d'un média. Par contre, elle a l'inconvénient d'être dans la plupart des cas manuelle et requiert donc une charge de travail élevée. De plus l'interprétation d'une image peut s'avérer délicate. Le sens donné à une image dépend entièrement de la vision que le documentaliste s'en fait et donc de sa culture et de son environnement. Certains mots peuvent avoir aussi plusieurs significations, prenons l'exemple du mot *restauration* qui peut désigner à la foi la réfection d'un bâtiment, une période de la monarchie française et se référer à la gastronomie !

Pour réduire les ambiguïtés du langage naturel (synonymie, homonymie, ...) lors de l'indexation, il serait intéressant d'intégrer un thésaurus. Un thésaurus est une liste de termes contrôlés organisés entre eux par des liens sémantiques et hiérarchiques. Cet ensemble de termes peut être divisé en deux sous-ensembles : les descripteurs et les non descripteurs. Un descripteur représente dans notre cas un terme qui est utilisé pour indexer une image. Un non descripteur serait quant à lui un terme spécifiant le descripteur. Comme le montre la figure 10.8, lorsque le documentaliste encode un terme d'indexation, le système pourrait l'avertir des ambiguïtés pouvant intervenir. Bien entendu, pour que ce soit réalisable, il faudrait encore que l'indexation se limite à un domaine bien particulier.

| Descripteur | Non descripteur |
|---|---|
| Restauration | Gastronomie<br>Monarchie<br>Réfection |

FIG. 10.8 – *Exemple de thésaurus*

Au vu de ces caractéristiques, il est important maintenant d'étudier comment les descripteurs visuels combinés aux descripteurs sémantiques peuvent faciliter l'indexation et améliorer la recherche de document multimédia.

## 10.2 Les descripteurs visuels

Les descripteurs visuels d'une image constituent des éléments d'indexation que des algorithmes peuvent extraire automatiquement de l'image et ce, sans connaissance particulière du contexte. Ces descripteurs, une fois extraits, sauvegardés dans

un fichier *MPEG-7* et stockés dans une base de données *XML* native, permettront d'effectuer des recherches sur base de caractéristiques telles que les couleurs, les formes, les structures, etc.

Les recherches peuvent être effectuées sur base d'une caractéristique visuelle données par l'utilisateur. Par exemple celui-ci peut demander : "*Affichez-moi toutes les images contenant des cercles et des carrés*", ou bien encore "*Affichez-moi toutes les images contenant beaucoup de rouge*". Il est vrai de prime abord que ce type de requêtes ne semblent pas être pertinente, mais le but est de pouvoir exprimer des requêtes en combinant plusieurs descripteurs visuels ou en les associant avec des descripteurs sémantiques.

Mais les recherches peuvent aussi être basées sur une image fournie par l'utilisateur. L'utilisateur demandera alors : "*Affichez-moi toutes les images similaires à celle-ci*". Dans ce cas, l'application va extraire de l'image les mêmes descripteurs visuels que ceux qui ont été extraits des images stockées, ensuite elle va comparer les descripteurs visuels de l'image de référence à ceux des autres images afin de retrouver les images similaires à celle que l'utilisateur a fourni.

Le problèmes des descripteurs visuels est qu'ils sont de trop bas niveau pour pouvoir être aisément manipulés dans un processus d'interrogation par des utilisateurs non avertis. Cependant leur principal intérêt est qu'ils peuvent être extraits automatiquement et très rapidement (comparé au temps qu'il faut à une personne pour décrire une image). Le courant actuel des recherches dans le domaine vise à associer ces caractéristiques bas niveau à des informations de plus haut niveau, afin d'automatiser le processus d'indexation de contenu.

Au niveau des descripteurs visuels, *MPEG-7* en définit un grand nombre. Il serait impossible de les décrire tous dans le cadre de ce mémoire, mais nous allons en décrire quelques uns parmi plus couramment utilisés.

### 10.2.1 Les histogrammes de couleurs

#### *A - Présentation de l'espace de couleur HSV*

Avant de passer à la présentation des histogrammes de couleur, nous allons vous présenter l'espace de couleur HSV qui nous sera utile plus tard dans cette section. HSV signifie : Hue (teinte), Saturation et Value (qui exprime la luminosité).

L'espace de couleur RGB n'est pas très pratique pour l'humain lorsqu'il veut exprimer sa perception d'une couleur. Lorsque l'on veut choisir une couleur pour repeindre les murs d'une pièce de la maison, on n'exprime pas la couleur choisie en

indiquant les concentrations de rouge, de bleu et de vert de cette couleur. On parlera plutôt d'une nuance de couleur, de sa luminosité et si elle doit être vive ou plutôt pastel. L'espace de couleur HSV permet de décrire une couleur dans ces différents termes.

Lorsque l'on parle d'une couleur, la première chose que l'on exprime est la teinte de la couleur. Elle décrit la nuance de la couleur, c'est-à-dire sa place dans le spectre de couleur. La figure 10.9 nous montre toutes les nuances de couleurs possibles. L'intervalle des teintes H est représenté sous forme de cercle, H peut donc prendre une valeur entre 0 et 360 degrés. Cependant H n'est pas exprimé en degrés, la valeur de H varie de 0 à 1. On démarre à 0 qui correspond au rouge, et en augmentant la valeur de H on passe par les couleurs jaune, vert, cyan, bleu, magenta et on revient au rouge quand on arrive à 1.

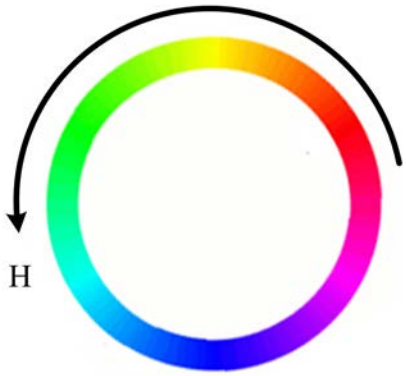


FIG. 10.9 – *Nuances de couleur*

Une fois que l'on a choisi sa nuance de couleur, on exprime la saturation de cette couleur, S. La saturation décrit la pureté de la teinte, c'est à dire la portion de couleur blanche qu'elle contient. Si on prend l'exemple de la couleur rouge, si elle ne contient pas de blanc elle est entièrement saturée, c'est-à-dire entièrement rouge. Si on lui ajoute du blanc elle devient plus pastel et vire au rose. La teinte est toujours rouge, mais elle est moins saturée. Lorsque la couleur n'est plus saturée, elle devient entièrement blanche. Ce phénomène est illustré sur l'image de gauche à la figure 10.10. La valeur de la saturation varie entre 0 et 1, 1 correspondant à 100% de saturation, on dit alors que la couleur est pure.

La dernière composante est la luminosité, elle est exprimé par la composante Value. Elle exprime la quantité de lumière qu'émet la couleur. Plus la couleur est lumineuse, plus elle parait claire et vice versa. Value varie de 0 à 1. La luminosité augmente lorsque la composante Value augmente. Ce phénomène est illustré sur l'image de droite à la figure 10.10.

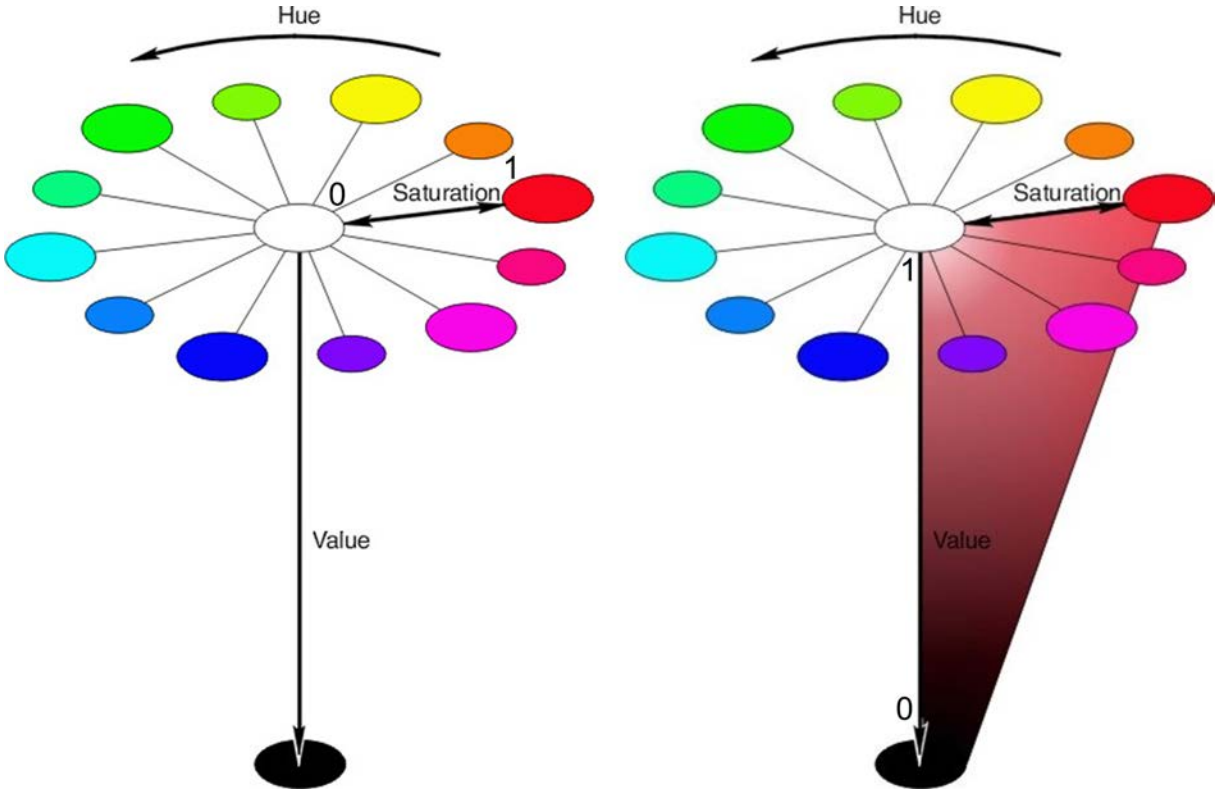


FIG. 10.10 – *Hexacone représantant l'espace de couleur HSV*

Pour pouvoir passer de l'espace de couleur RGB au HSV, il suffit d'utiliser les équations suivantes :

$$\begin{cases} H = \cos^{-1}\left[\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right] \\ S = 1 - \frac{3}{R+G+B}[\min(R,G,B)] \\ V = \frac{1}{3}(R+G+B) \end{cases} \tag{10.1}$$

## *B - Présentation*

Dans le domaine de la récupération d'image, les histogrammes de couleur sont généralement utilisés pour décrire les images. En effet ils fournissent un aperçu très compact de l'image et en cas de translation ou de rotation ils restent inchangés. En fait un histogramme exprime la distribution des couleurs d'une image. Pour obtenir un histogramme de couleur, il suffit de compter tous les pixels de chaque couleur de l'espace de couleur RGB ou HSV.

Avant de pouvoir calculer un histogramme de couleur, il faut d'abord quantifier l'espace de couleurs, c'est-à-dire regrouper les couleurs très proches en un intervalle de couleurs, ensuite il faut compter le nombre de chaque pixel appartenant à chaque intervalle. L'étape de quantification est très importante vu qu'en général les images sont encodées en 24 bits, soit 16.777.216 couleurs, elle permet ainsi de réduire le volume d'information à conserver. Dès lors les principaux problèmes liés à l'élaboration d'un histogramme sont le choix d'un espace de couleur, le type de quantification et le degré de quantification.

Un histogramme de couleur peut être vu comme un ensemble de vecteurs. Pour une image en niveaux de gris, les vecteurs appartiennent à un espace à deux dimensions. La première donne le niveau de gris et l'autre dimension donne le nombre de pixels de ce niveau de gris. Mais pour les images couleurs, les histogrammes de couleur sont en fait composés d'un ensemble de vecteurs appartenant à un espace à quatre dimensions : 3 dimensions pour chaque composante de l'espace de couleur et une dimension pour le nombre de pixels. Ce qui fait que ces histogrammes sont très difficiles à visualiser. Cependant la façon la plus simple de se représenter un histogramme de couleur consiste à calculer un histogramme pour chaque composante de l'espace de couleur. Pour des considérations pratiques telles que la compatibilité avec les écrans des terminaux, c'est l'espace de couleur RGB qui est souvent choisi.

Un autre avantage non négligeable de la division de l'histogramme des couleurs en histogrammes de chacun de ses composantes de couleur est que cette représen-

tation est plus compacte. En effet une image RGB est encodée en 24 bits, 8 bits pour chacune des composantes rouge, bleue et verte. Ce qui fait que chaque histogramme exprime le nombre de pixels pour seulement 256 valeurs ($2^8$ couleurs). Donc pour les trois histogramme ensembles on n'a plus que 768 valeurs au lieu des 16.777.216 pour l'histogramme unique.

La figure 10.11 illustre le calcul des histogrammes de couleurs pour une image RGB. Tous d'abord, l'image est décomposée dans chacune de ses composantes de couleur de base. En fait, à chaque couleur de base correspond une image en niveaux de gris où le niveau de d'intensité chaque pixel exprime l'intensité de la couleur. Plus le pixel est clair, plus l'intensité de la couleur est forte. Pour chaque couleur de base, il suffit pour chaque intensité de couleur de compter le nombre de pixels ayant cette intensité.

*MPEG-7* utilise trois types d'histogrammes de couleur :

1. Le descripteur Scalable Color est un histogramme de couleur extrait de l'espace de couleur HVS. L'extraction de ce descripteur commence avec le calcul de l'histogramme avec 256 intervalles dans l'espace de couleur HVS : 128 pour la composante H et 64 pour les composantes pour S et V. Une fois l'histogramme calculé, on lui applique la transformée de Haar en commençant avec H, puis S et V.

2. Le Color Structure est une généralisation de l'histogramme de couleur exprimant certaines caractéristiques spatiales de la distribution des couleurs d'une image. Pour son extraction, l'image est divisée en blocs de 8x8 pixels, au lieu de considérer chaque pixel séparément.

3. Le GoF/GoP Color Descriptor qui est utilisé pour la description de vidéo.

La figure 10.14 montre la description *XML MPEG-7* des différents descripteurs que nous venons de voir.

## *C - Similarité des images*

Lors de la recherche d'image basée sur des histogrammes de couleur (illustrée à la figure 10.12), l'histogramme de couleur de l'image de référence va être comparé avec les histogrammes de chaque image. Afin de mesurer la similarité de deux histogramme, on utilise des formules de calcul de distance. Les autres techniques de comparaison de distribution de probabilité tels que les tests de Kolmogoroff-Smirnov, ne sont pas appropriées dans le cas des histogrammes de couleurs. Ceci à cause du fonctionnement de la perception visuelle qui préfère la similarité à la proximité des images. Les formules de distance de couleur sont en fait des mesures de similarité

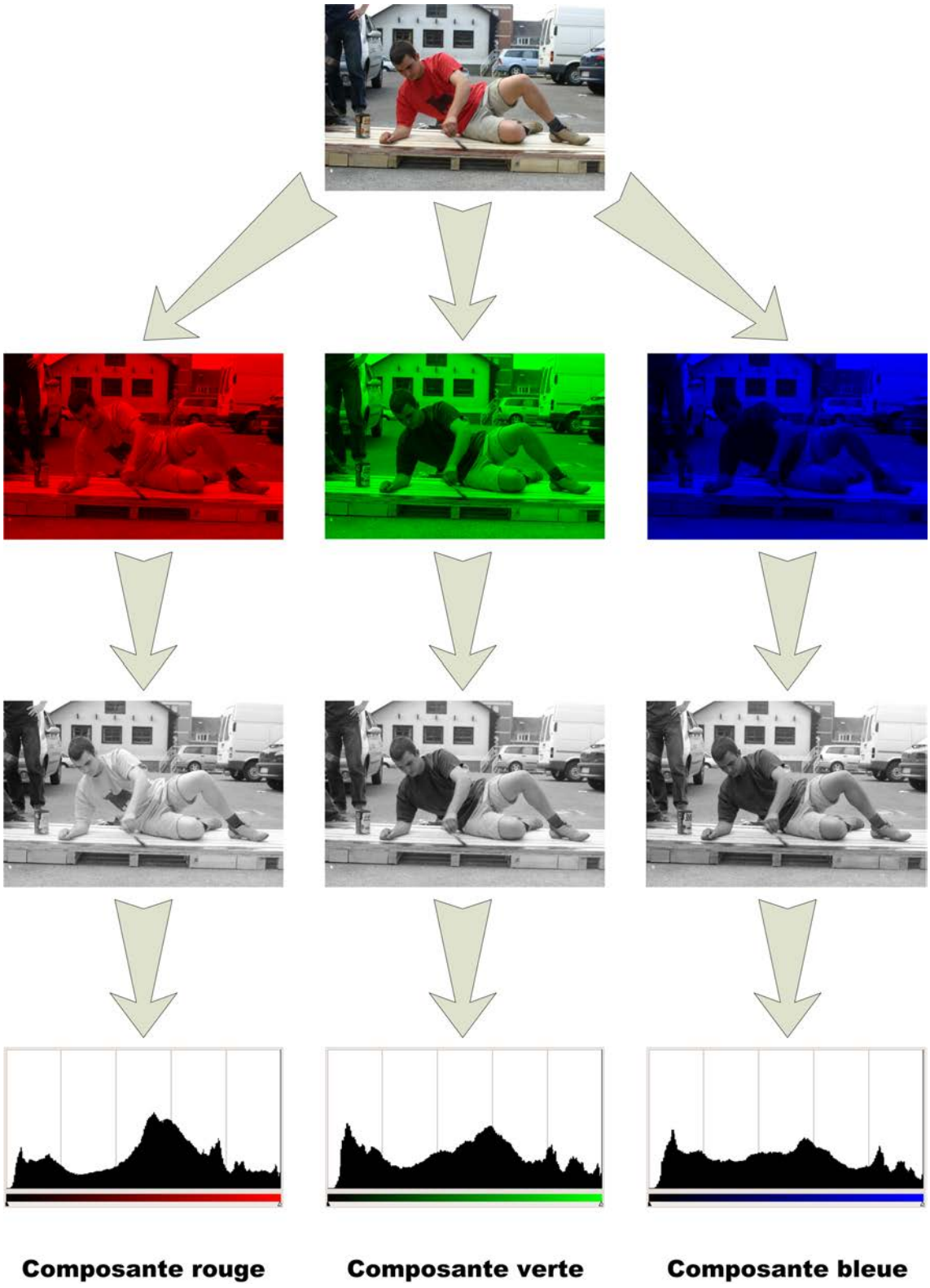


FIG. 10.11 – *Etablissement d'un histogramme dans l'espace de couleur RGB*

entre images, basés sur la perception des couleurs contenues dans les images. Les formules de distances utilisées sont au nombre de trois : distance euclidienne, la distance d'intersection et la distance croisée quadratique.

*Distance euclidienne*

Soit $h$ et $g$ deux histogrammes de couleur à comparer. La distance euclidienne entre les histogrammes de couleur $h$ et $g$ est calculée avec la formule 10.2. Cette formule est appliquée pour des histogrammes de couleur dans l'espace de couleur RGB, mais elle s'applique de la même façon pour n'importe quel autre espace de couleur.

$$d^2(h,g) = \sum_R \sum_G \sum_B (h(r,g,b) - g(r,g,b))^2 \tag{10.2}$$

Dans cette formule chaque intervalle de l'histogramme $h$, le $i^{ieme}$ par exemple, est comparé avec son équivalent, le $i^{ieme}$ dans l'histogramme $g$. Tous les intervalles ont le même poids dans le calcul de la distance.

*Distance d'intersection*

Soient $h$ et $g$ deux histogrammes de couleur à comparer. L'intersection euclidienne entre les histogrammes de couleur $h$ et $g$ est calculée avec la formule 10.3. Cette formule est appliquée pour des histogrammes de couleur dans l'espace de couleur RGB, mais elle s'applique de la même façon pour n'importe quel autre espace de couleur.

$$d(h,g) = \frac{\sum_R \sum_G \sum_B \min\left(h(r,g,b), g(r,g,b)\right)}{\min(|h|,|g|)} \tag{10.3}$$

Dans la formule 10.3, |h| et |g| représentent la magnitude de chaque histogramme, elle est égale au nombre d'échantillons. L'avantage ici est que les couleurs qui ne sont pas présentes dans l'image de référence de la requête ne sont pas prises en compte pour le calcul de la distance d'intersection. Ce qui réduit l'impact des couleurs de l'arrière plan. La fonction d'intersection prend des valeur entre 0 et 1, les valeurs proche de 1 indique une bonne ressemblance.

*Distance croisée quadratique*

La distance quadratique entre histogrammes de couleur est utilisée dans le système QBIC. Soient $h$ et $g$ deux histogrammes de couleur à comparer. La distance croisée quadratique entre les histogrammes de couleur $h$ et $g$ est calculée avec la formule 10.4.

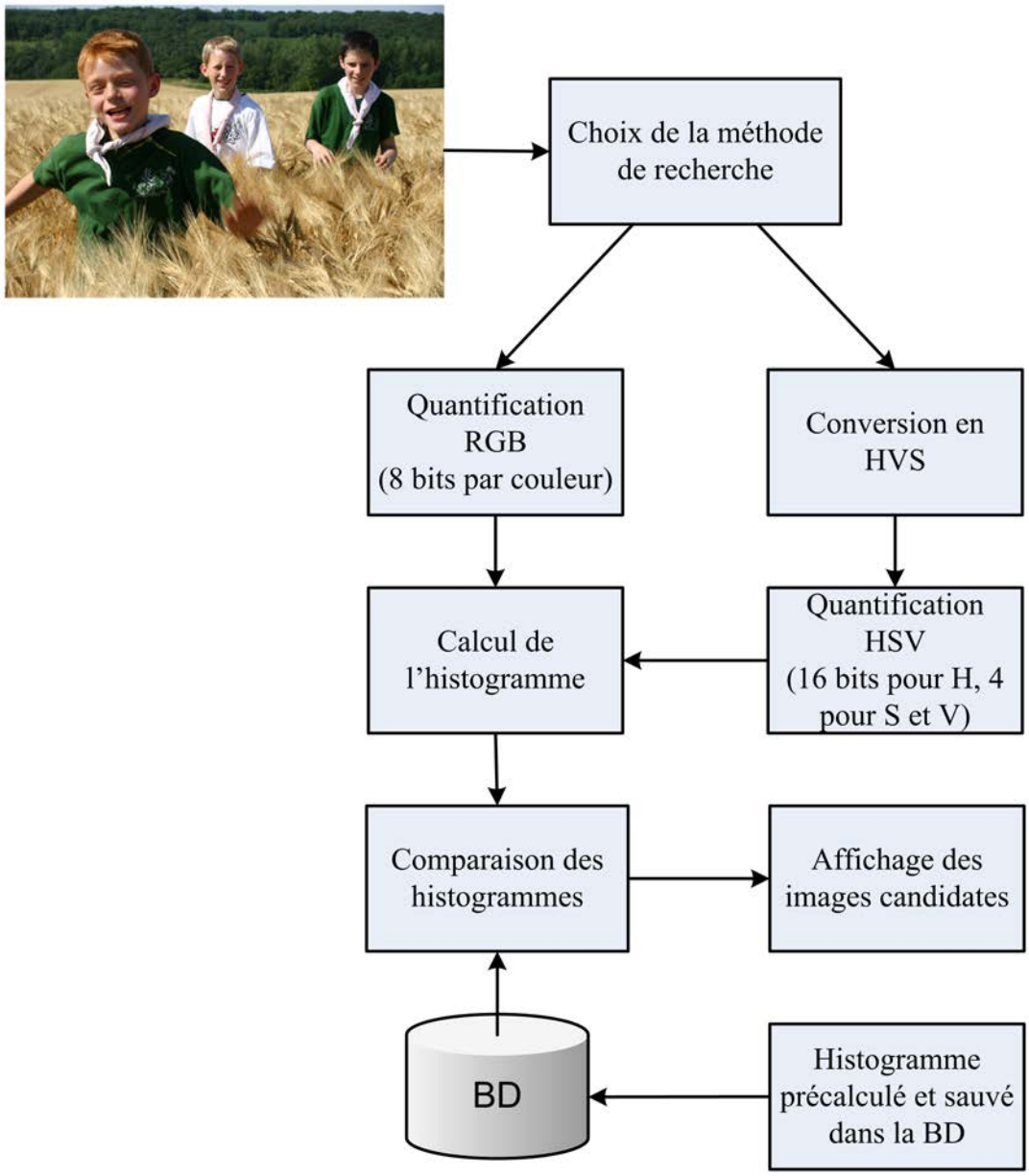


FIG. 10.12 – *Architecture de la recherche par histogramme de couleur*

$$d(h,g) = (h-g)^t A(h-g) \tag{10.4}$$

La formule de distance croisée tient compte de la corrélation croisée entre les intervalles des histogrammes basés sur la similarité des couleurs représentées par les intervalles. L'ensemble de toutes les valeurs des corrélations croisées sont représentées par la matrice $A$, appelée matrice de similarité. Cette matrice se calcule différemment suivant que l'on choisisse l'espace de couleur RGB ou HSV.

Pour l'espace de couleur RGB elle se calcule comme suit :

$$a_{i,j} = 1 - \frac{d_{i,j}}{\max(d_{i,j})} \tag{10.5}$$

où $d_{i,j}$ est la distance euclidienne entre la couleur $i$ et la couleur $j$, et $\max(d_{i,j)}$ est la distance maximum entre les couleurs. Les $a_{i,j}$ permettent de donner un poids à la différence entre la couleur $i$ et la couleur $j$

Pour l'espace de couleur HSV elle se calcule comme suit :

$$a_{i,j} = 1 - \frac{1}{\sqrt{5}}\left[(v_i - v_j)^2 + (s_i \cos h_i - s_j \cos h_j)^2 + (s_i \sin h_i - s_j \sin h_j)^2\right]^{\frac{1}{2}} \tag{10.6}$$

### 10.2.2 Color Layout

Le Color Layout Descriptor (CLD) caractérise de façon compacte la distribution spatiale des couleurs d'une image. Le CLD utilise un tableau de 8x8 cases contenant les couleurs représentatives de l'image. Les couleurs sont exprimées dans l'espace de couleur YUV. La taille du tableau est fixée pour rendre le descripteur indépendant de l'échelle. C'est à dire que deux images identiques mais de tailles différentes auront le même CLD. La figure 10.13 illustre la méthode pour extraire ce descripteur.

Après avoir déterminé la couleur représentative de chacune des 64 cases du tableau, l'image résultante subit une DCT[3]. Le résultat de la DCT passe ensuite par la méthode de diagonalisation. On prend ensuite les six premiers coefficients basses fréquences pour Y et les 3 premiers pour U et V.

Ce type de descripteur permet d'effectuer des recherches par images, mais aussi par croquis. Aucun des autres descripteurs de couleurs ne permettent la recherche

[3] Voir chapitre 3 sur le MPEG-1

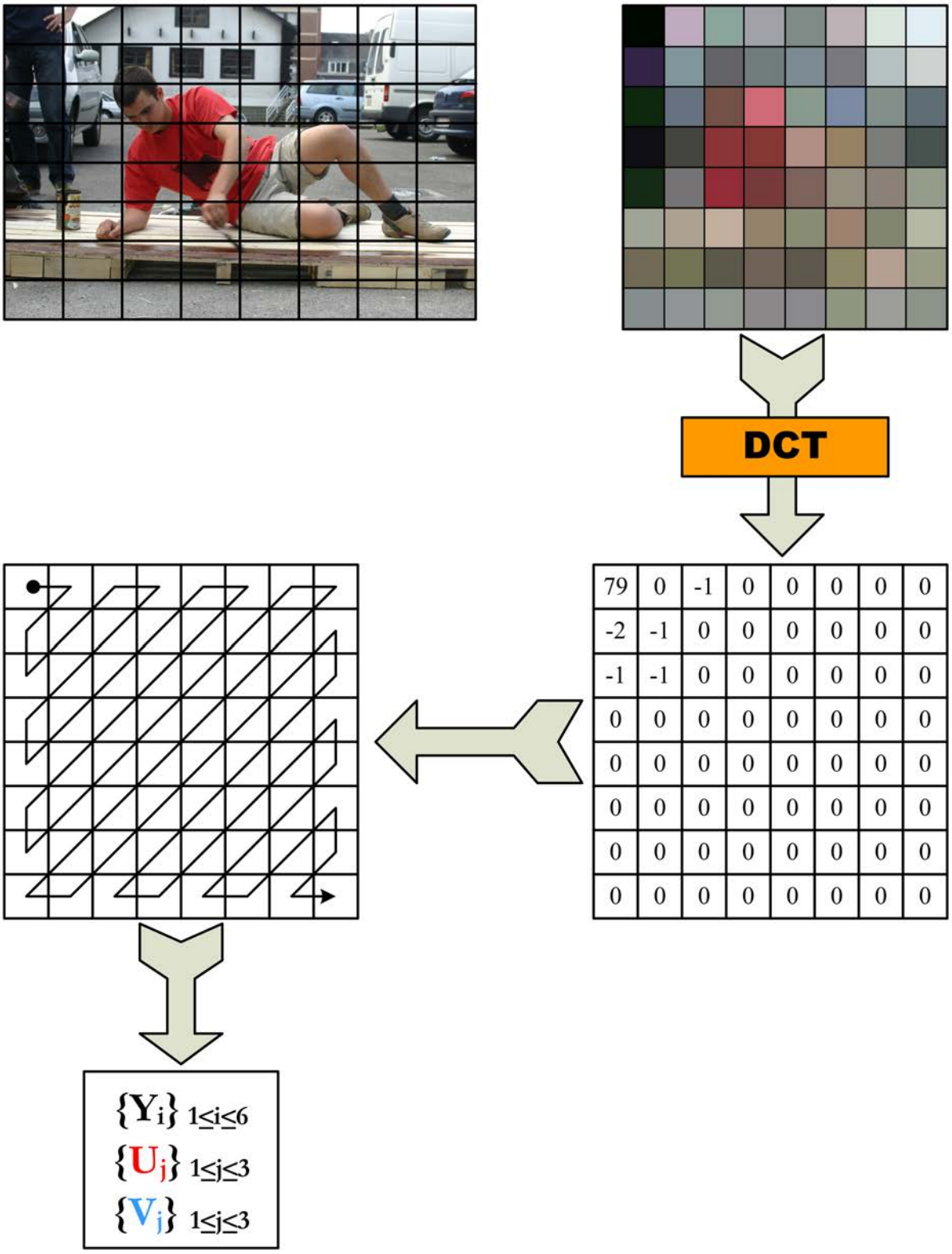


FIG. 10.13 – *Technique d'extraction du Color Layout*

par croquis. Cette technique de recherche est très utile car elle est simple et proche de l'utilisateur. Le CLD est utilisé pour des recherches rapides dans des base de données et comme filtre dans des application de radiodiffusion.

La figure 10.14 montre la description *XML MPEG-7* du Color Layout.

### 10.2.3 Dominant Color

Ce descripteur fournit une description compacte des couleurs représentatives d'une image ou d'une région d'une image. La technique est assez simple, elle consiste à compter le nombre de pixels pour chacune des couleurs de l'image ou de la région de l'image. On ne retient ensuite que les $n$ couleurs dominantes. Les couleurs représentatives sont calculées pour chaque image plutôt que d'être fixées dans un espace de couleur. C'est-à-dire que les couleurs dominantes sont réellement présentes dans l'image, elles ne sont pas une approximation via un intervalle de couleur. En général on ne retient que quatre couleurs dominantes.

Le Dominant Color Descriptor fournit pour chacune des $n$ couleurs dominantes les informations suivantes :

$$DCD = ((c_i, p_i, v_i)_{1 \leq i \leq n}, s)$$

- $c_i$ : est la $i^{ieme}$ couleur dominante.
- $p_i$ : est le pourcentage des pixels de l'image ou de la région de l'image associés à la couleur $c_i$.
- $v_i$ : est la variance des couleurs associées à $c_i$, étant optionnelle, elle n'est pas toujours reprise.
- $s$ : est la cohérence spatiale des couleurs dans l'image.

La recherche via ce descripteur est assez triviale. Il suffit de rechercher les images qui ont plus ou moins les mêmes couleurs dominantes.

La figure 10.14 montre la description *XML MPEG-7* des différents descripteurs que nous venons de voir.

## 10.3 Références

Afin de réaliser ce chapitre nous nous sommes inspirés des documents suivants : [Don04], [Jeo06], [Mod06], [OAM02], [Wan06].

```
<?xml version="1.0" encoding="UTF-8"?>
<Mpeg7>
  <Creator>
       <GivenName>Paul</GivenName>
       <FamilyName>Delcorde</FamilyName>
  </Creator>
<!-- Descripteurs visuels -->
  <VisualDescriptors>
    <ColorLayout>
        <YDCCoeff>17</YDCCoeff>
        <CbDCCoeff>16</CbDCCoeff>
        <CrDCCoeff>32</CrDCCoeff>
        <YACCoeff63>14 13 17 ... 15 16 15</YACCoeff63>
        <CbACCoeff63>16 15 18 ... 15 16 15</CbACCoeff63>
        <CrACCoeff63>13 14 14 ... 16 15 16</CrACCoeff63>
    </ColorLayout>
    <ScalableColor numOfBitplanesDiscarded="0" numOfCoeff="256">
        <Coeff>-202 71 27 ... 1 3 1</Coeff>
    </ScalableColor>
    <DominantColorType>
        <SpatialCoherency>0</SpatialCoherency>
        <Value>
          <Percentage>8</Percentage>
          <Index>10 10 10</Index>
        </Value>
        <Value>
          <Percentage>5</Percentage>
          <Index>10 7 5</Index>
        </Value>
        <Value>
          <Percentage>3</Percentage>
          <Index>21 25 26</Index>
        </Value>
        <Value>
          <Percentage>2</Percentage>
          <Index>27 28 26</Index>
        </Value>
    </DominantColorType>
  </VisualDescriptors>
  <MediaLocator>
    <MediaUri>D:Mes Images/camp2004-23.jpg</MediaUri>
  </MediaLocator>
</Mpeg7>
```

FIG. 10.14 – *Description XML MPEG-7 des descripteurs de couleurs*

CHAPITRE 11

# Stockage des descriptions MPEG-7

## 11.1 Introduction

Comme nous l'avons vu dans le chapitre 7 consacré au *MPEG-7*, les descripteurs *MPEG-7* sont stockés dans des fichiers *XML*. Dès lors dans le cadre du développement d'une application de gestion d'images utilisant les descripteurs *MPEG-7*, un des problèmes cruciaux à résoudre est le stockage des fichiers *XML* servant à décrire les images que l'application devra gérer.

En ce qui concerne le stockage des fichiers *XML*, à l'heure actuelle différents types de solutions sont proposés. Les fichiers *XML* peuvent être stocké dans des bases de données relationnelles étendues au *XML*, dans des bases de données orientées objets ou dans des bases de données *XML* natives. Ce chapitre a donc pour objectif de passer en vue ces différents types de bases de données et de voir lequel est le plus approprié au stockage de fichiers *XML*.

Avant de décrire les solutions de stockage disponibles pour les documents *XML*, il faut bien comprendre le fait qu'un document *XML* présente de nombreuses différences par rapport aux données relationnelles classiques.

Premièrement, un document *XML* présente une sémantique intrinsèque due à sa structure d'arbre (voir figure 11.1). Les éléments du fichier peuvent avoir entre

eux soit une relation de contenance (HAS-A), soit une relation de sous-classe (IS-A). Dans un arbre *XML*, les noeuds peuvent être vus comme les membres d'une dimension déterminée par les noeuds parents, et les feuilles comme des mesures. De plus chaque élément de l'arbre peut avoir ses propres mesures (attributs).

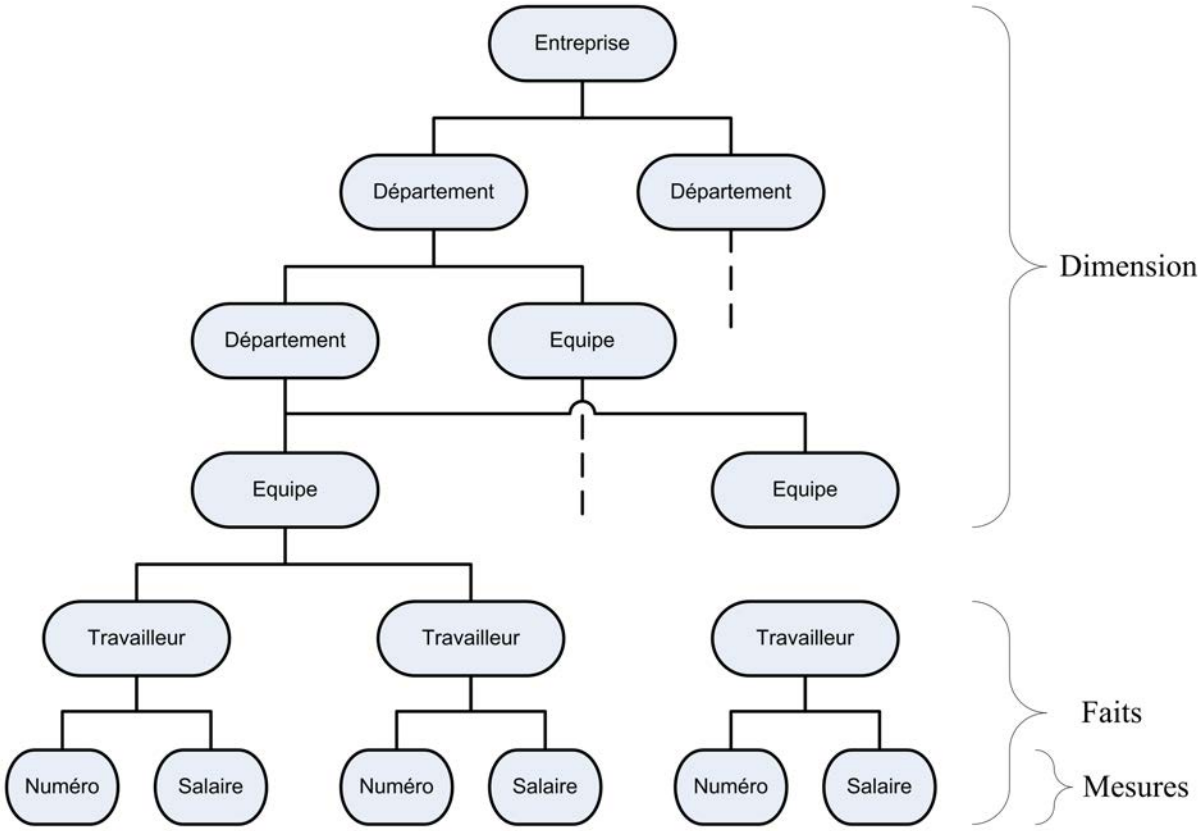


FIG. 11.1 – *Exemple d'arbre XML*

Deuxièmement, les documents *XML* peuvent être semi-structurés, c'est-à-dire qu'ils ne respectent pas obligatoirement une structure rigide et peuvent avoir une structure plus ou moins irrégulière, contrairement aux schémas relationnels. Les documents *XML* respectent quand même une certaine structure, mais celle-ci est assez souple. On peut avoir des structures récursives ou hiérarchiques contenant différents membres.

Les documents *XML* peuvent se parcourir suivant différents axes. Un noeud d'un arbre XML est accessible via de multiples chemins au contraire d'une case dans un système relationnel.

Pour finir un document *XML* peut contenir des données non numériques. Par exemple, des séquences ADN, des descripteurs MPEG-7 tels que des histogrammes

de couleurs. Ce type de données est assez complexe à gérer dans des bases de données traditionnelles.

Au vu de ces différence avec le modèle relationnel classique, on voit apparaître aussi des différences dans les types d'interrogations pouvant être effectuées sur des données *XML* :

- L'ordre d'un document *XML* est important à cause de sa sémantique. Donc les requêtes doivent savoir exploiter ces relations d'ordres. Par exemple pour une description détaillée des différentes procédures d'un soin médical il est obligatoire de retrouver les différentes opérations dans l'ordre exact dans lesquelles elles doivent être effectuées. Cela afin de ne pas ouvrir le patient avant de l'anesthésier.
- Il faut avoir la possibilité d'exécuter des requêtes sur des données non numériques tels que des chaînes ADN ou des descripteurs *MPEG-7*.
- Si l'on peut grouper des éléments d'un document *XML* par valeurs de certains de leurs attributs (comme dans les systèmes relationnels), on peut aussi grouper des éléments via leur chemin exprimé par exemple par une expression XPath[1]. Sur base de la figure 11.1, on peut regrouper les employés par département, on peut utiliser le chemin //département comme clé de groupement.

- Prévoir des tendances futures est une opération commune dans les systèmes relationnels. Pour cette opération on modifie certaines mesures et on étudie les impacts de ces changements sur l'ensemble des données. Avec une structure d'arbre cette opération est facilitée en modifiant simplement la structure de l'arbre. Par exemple, si l'on veut analyser l'impact d'une réorganisation des départements et des équipes, il suffit de déplacer les éléments équipes dans d'autres éléments départements et de comparer les résultats des requêtes sur cette nouvelle structure. Dans les systèmes relationnels, ces analyses structurelles sont compliquées car elles nécessitent un changement dans le schéma de la base de données, ce qui est une opération ardue.

## 11.2 Types de bases de données supportant XML

### 11.2.1 Les bases de données relationnelles étendues XML

Les bases de données relationnelles (SGBDR) sont massivement utilisées depuis les années 1980. Beaucoup d'entreprises utilisent ce type de base de données pour

[1] Language de requète sur des BD XML, voir chapitre suivante

stocker et récupérer leurs informations. A l'heure actuelle ces systèmes sont très performants et fiables. Depuis les années 2000, avec le développement du *XML*, de nombreux moteurs relationnels supportent la gestion des document *XML*.

Au départ, les documents étaient stockés et récupérés entièrement, sans traitement particulier. Le fichier *XML* était stocké dans une table de la base de données sous formes d'une longue chaîne de caractères dans une cellule de type CLOB[2] comme illustré à la figure 11.2. Cette approche permet de préserver le document dans son intégrité, mais n'offre aucun moyen de manipulation de son contenu.

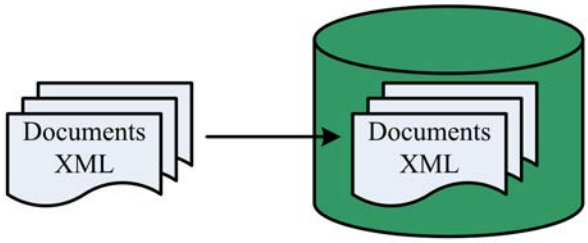


FIG. 11.2 – *Intégration des documents XML*

Une autre technique permettant de stocker un document *XML*est d'éclater sa structure en un ensemble de tables. Cette technique nécessite une table pour le nom du fichier (figure 11.4), une table reprenant tous les éléments présents dans le fichier (figure 11.5, chaque élément apparaît dans la table autant de fois que dans le fichier), une table reprenant les attributs de chaque éléments (figure 11.6) et une table reprenant les valeurs de chaque éléments (figure 11.7). Cette technique est illustrée au travers des figure 11.3 à 11.7.

[2] CLOB : Character Large Object, objet pouvant contenir une longue chaîne de caractères.

```
<BonCommandes>
 <BonCommande Numero= "123">
  <Date>2005-8-18</Date>
  <NumClient>456</NumClient>
  <Article Numero= "1">
   <NumeroLot>AB-12</NumeroLot>
   <Quantite>14</Quantite>
   <Prix>16.80</Prix>
  </Article>
  <Article Numero= "2">
   <NumeroLot>CD-34</NumeroLot>
   <Quantite>6</Quantite>
   <Prix>2.34</Prix>
  </Article>
 </BonCommande>
</BonCommandes>
```

FIG. 11.3 – *Fichier XML dont la structure va être éclatée en tables*

| ID | NomFichier |
|---|---|
| 1 | "BonDeCommandes.xml" |

FIG. 11.4 – *Table des fichiers*

| ID_Document | ID_Element | ID_Parent | Nom | OrdreDsPerant |
|---|---|---|---|---|
| 1 | 1 | NULL | "BonCommandes" | 1 |
| 1 | 2 | 1 | "BonCommande" | 1 |
| 1 | 3 | 2 | "Date" | 1 |
| 1 | 4 | 2 | "NumClient" | 2 |
| 1 | 5 | 2 | "Article" | 3 |
| 1 | 6 | 5 | "NumeroLot" | 1 |
| 1 | 7 | 5 | "Quantite" | 2 |
| 1 | 8 | 5 | "Prix" | 3 |
| 1 | 9 | 2 | "Article" | 4 |
| 1 | 10 | 9 | "NumeroLot" | 1 |
| 1 | 11 | 9 | "Quantite" | 2 |
| 1 | 12 | 9 | "Prix" | 3 |

FIG. 11.5 – *Tables de éléments*

| ID_Document | ID_Attribut | ID_Parent | Nom | Valeur |
|---|---|---|---|---|
| 1 | 1 | 2 | "Numero" | 123 |
| 1 | 2 | 5 | "Numero" | 1 |
| 1 | 3 | 9 | "Numero" | 2 |

FIG. 11.6 – *Table des attributs*

| ID_Document | ID_Valeur | ID_Parent | Valeur | OrdreDsParent |
|---|---|---|---|---|
| 1 | 1 | 3 | "2005-8-18" | 1 |
| 1 | 2 | 4 | "456" | 1 |
| 1 | 3 | 6 | "AB-12" | 1 |
| 1 | 4 | 7 | "14" | 1 |
| 1 | 5 | 8 | "16.80" | 1 |
| 1 | 6 | 10 | "CD-34" | 1 |
| 1 | 7 | 11 | "6" | 1 |
| 1 | 8 | 12 | "2.34" | 1 |

FIG. 11.7 – *Table des valeurs*

Un gros inconvénient de cette technique est qu'elle ne permet de stocker les valeurs que sous la forme d'une chaîne de caractères (en effet elles sont toutes reprise dans la même colonne). La solution à ce problème serait de faire une table par type de valeurs, ce qui devient vite très lourd comme technique s'il y a beaucoup de types différents dans le fichier. Comme on le remarque très vite, cette technique ne permet pas aux fichiers *XML* d'être flexibles. De plus elle nécessite beaucoup d'informations additionnelles pour pouvoir reconstruire le fichier. Le fichier *XML* pris comme exemple est assez court, cependant en *MPEG-7* la taille des fichiers est nettement plus conséquente. Donc cette technique serait beaucoup trop lourde pour ce genre de fichier et pour les document *XML* de grande taille en général.

C'est pourquoi les moteurs relationnels sont passés au stockage structuré des documents *XML*. L'information contenue dans le document est éclatée dans différentes tables de la base de données (opération de shredding). Ensuite pour récupérer le document, il suffit d'exécuter des requêtes pour récupérer les différentes informations du fichier et de reconstruire le document *XML* (opération de publishing). Ces techniques sont illustrées à la figure 11.8.

Avec cette méthode il n'est bien entendu pas possible de retrouver le document via son nom de fichier. De plus il n'y a aucune garantie que le document original puisse être reconstruit à l'identique. Pour cette raison ce n'est pas une bonne idée d'utiliser cette technique pour stocker des document *XML*, surtout si on doit conser-

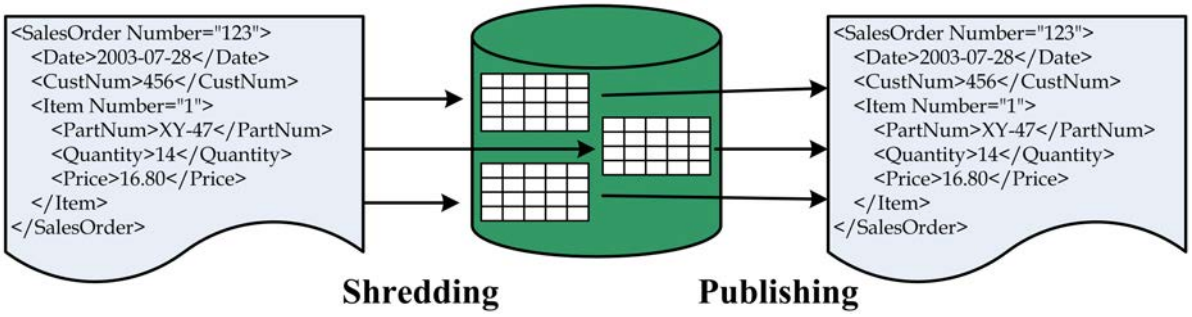


FIG. 11.8 – *Stockage structuré de document XML*

ver la structure du document intacte. Par contre cette technique peut être utilisé dans le cas d'applications qui reçoivent des informations sous forme de documents *XML*, traitent ces informations, et sortent ces informations dans de nouveaux document *XML* d'une autre forme. Cette approche ne permet pas de conserver l'intégrité du document, mais offre des possibilité de manipulation de son contenu.

Prenons l'exemple d'une application météorologique qui reçoit des documents *XML* de stations météo aux quatre coins de la Belgique. Les informations de chaque station vont être éclatées dans différentes tables que manipule l'application. Ensuite un Web service pourra demander à l'application de lui fournir un bulletin météo de la Belgique sous forme d'un document *XML*.

Pour que le stockage structuré fonctionne correctement, il faut pouvoir pouvoir faire entrer le schéma *XML* dans le moule du schéma de base de donnée, et inversement, il faut pouvoir faire rentrer le schéma de la base de données dans le moule du schéma *XML*, c'est ce qu'on appelle le mapping.

Il y a trois sortes de mapping importantes : le mapping table-based, le mapping object-relational et les langages de requête. Tandis que les langages ne permettent que de faire du mapping dans un seul sens, du schéma de la base de données vers le schéma *XML*, les mapping table-based et object-relational sont quant à eux bidirectionnels.

Pour le **mapping table-based**, le document *XML* doit avoir la même structure que celle de la base de données relationnelle, comme illustré à la figure 11.9. On retrouve d'abord tous les bon de commandes et ensuite tous les articles.

Cette illustration nous montre aussi les limites de cette techniques :

– Les données du fichier sont reprises table par table. Donc on retrouve les articles en dehors de leur bon de commande, ce qui ne serait pas le cas dans une structure classique de fichier *XML*.


FIG. 11.9 – *Exemple de fichier XML pour le mapping table-based*

- Le numéro de bon de commande, utilisé comme clé étrangère pour lier les deux tables, apparaît plusieurs fois, une fois dans les données du bon de commande et dans les données de chaque article du bon de commande.
- Le document *XML* dispose d'un élément qui englobe tous les autres éléments : Database. Or cet élément ne correspond à aucune structure de la base de données, il n'existe que parce que un fichier *XML* nécessite une balise racine unique.

Comme on peut le voir à la figure 11.9, le mapping table-based peut utiliser soit des noeuds fils, soit des attributs ou un mix des deux.

Quand on utilise le **mapping object-relational**, le document *XML* est vu comme un ensemble d'objets comme illustré à la figure 11.10. Les objet deviennent des tables, les attributs deviennent des colonnes, les éléments enfants simples deviennent des colonnes et les références à des éléments enfants complexes (un autre objet) deviennent des relations primary key/foreign key.

Contrairement au mapping table-based, on voit à la figure 11.10 que le mapping object-relational respecte la structure classique des fichiers *XML*, c'est à dire que les éléments enfants se retrouvent dans leur élément parents.

### 11.2.2 Les bases de données orientées objets

Fin des années 80, une nouvelle forme de système de gestion de base de données a été introduite sur le marché. Il s'agit des SGDBOO, les Systèmes de Gestion de Base de Données Orientées-Objet. Ces systèmes gèrent les données comme des objets, plutôt que comme des tables, des lignes ou des colonnes. En effet, il est plus naturel de représenter le monde réel comme une collection d'objets, chacun ayant un état et des comportements spécifiques. De plus, les langages de programmation orientés objet commençant à être à la mode à cette époque, les développeurs voulaient pouvoir stocker de manière persistante leurs objets, ce que les SGDBOO permettent de faire assez proprement.

Malheureusement, les SGDBOO souffrent de ne pas avoir un langage de requête commun et standardisé, comme l'est SQL pour les SGBDR. La communauté existante autour des bases de données orientées-objet a donc décidé d'adapter SQL. Le résultat en est OQL, un langage permettant uniquement de chercher et de récupérer des données, mais pas de les mettre à jour.

Ce type de base de données aurait pu être très intéressant pour notre étude car on peut considérer un document XML comme une collection hiérarchisée d'objets. En effet, chaque noeud d'un document XML a une identité unique, comme les objets et un objet peut en inclure d'autres. Malheureusement, l'ODMG, le groupe de gestion des bases de données objets, n'a jamais publié de version permettant de gérer l'XML. Seul un groupe de chercheurs a créé un tel système, devenu libre par la suite, Ozone, qui, selon eux, est totalement compatible avec le DOM du W3C.

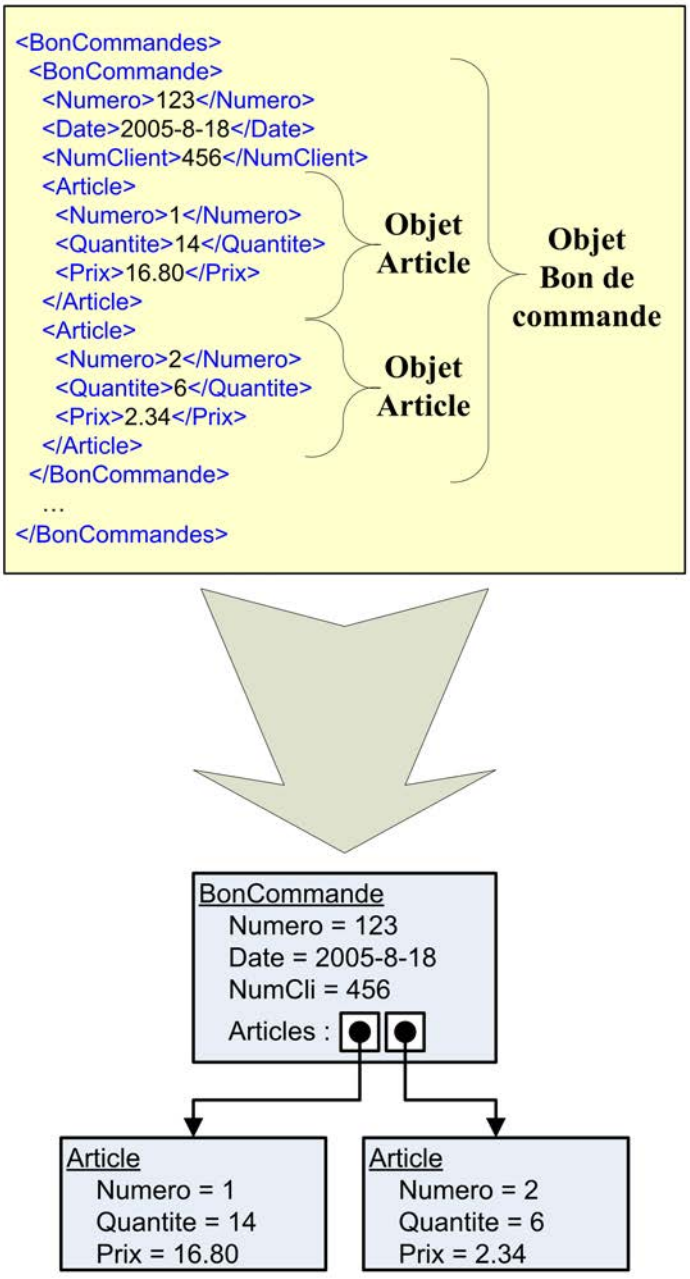


FIG. 11.10 – *Exemple de fichier XML pour le mapping object-relational*

### 11.2.3 Les base de données XML natives

#### *A - Présentation des BD XML natives*

Les bases de données *XML* natives sont des bases de données conçues spécialement pour stocker des documents *XML*. Comme les autres types de bases de données, elles offrent les fonctionnalités de transaction, sécurité, d'accès mutli-utilisateurs, un ensemble d'APIs, des langages de requête, ... La seule vraie différence c'est que leur modèle interne est basé sur *XML* et non sur un quelconque autre modèle tel que le modèle relationnel.

Les BD *XML* natives sont principalement utilisées pour stocker des documents orientés document. Ils sont caractérisés par une structure peu régulière, des données qui présentent une granularité élevée et beaucoup de contenus mixtes, l'ordre dans lequel les éléments enfants d'un même parent et les PCDATA apparaissent est presque toujours significatif, tout le contraire de documents orientés données. Une des raisons qui fait que que les BD *XML* natives sont utilisées pour ce genre de document, c'est que ces BD permettent de conserver l'ordre des document, les informations de traitement, les commentaires, les sections CDATA, l'utilisation des entités, ... ce que ne permettent pas les bases de données étendues *XML*.

Dans un système relationnel, les données doivent s'adapter au modèle relationnel assez rigide. A l'inverse les BD *XML* natives sont plus flexibles. De plus elles supportent plus facilement les modifications de schéma que leur homologues relationnels, elle permettent même de manipuler des données qui n'ont aucun schéma particulier.

Un autre avantage des BD *XML* natives, c'est qu'elles gèrent des données semi-structurées, c'est-à-dire que la structure des documents stockés ne doit pas être rigoureusement identique d'un document à l'autre. Ceci est très utile pour stocker des données venant de mondes qui changent rapidement, et pour lesquels il est difficile d'établir un schéma définitif, dans le monde de la finance ou de la biologie par exemple. Une liste plus longue des différents cas d'utilisation des BD *XML* natives est disponibles sur `http://www.rpbourret.com/xml/UseCases.htm`

L'une des définitions possibles, développée par les membres de la XML :DB mailing list, est la suivante :

*"Une base de données XML native définit un modèle (logique) de document XML (modèle est ici opposé aux données du document), et stocke et retrouve les documents en fonction de ce modèle. Le modèle doit au minimum inclure les éléments, les attributs, les PCDATA et l'ordre interne du document. Quelques exemples de tels modèles sont : le modèle de données de XPath, le glossaire XML Infoset, et les modèles implicites de DOM et des événements de SAX 1.0.*

*Le document XML est l'unité fondamentale du stockage (logique) dans une base de données XML native, tout comme une ligne d'une table constitue l'unité fondamentale du stockage (logique) dans une base relationnelle.*

*Une base de données XML native ne repose pas sur un modèle physique particulier pour le stockage. Elle peut par exemple être bâtie aussi bien sur une base relationnelle, hiérarchique, orientée-objet, ou bien utiliser des techniques de stockage propriétaires comme des fichiers indexés ou compressés."*

## *B - Les fonctionnalités des BD XML natives*

*Collections de document*

La plupart des bases de données *XML* natives fournissent la notion de collection. Une collection joue un rôle similaire à celui d'une table dans un modèle relationnel ou à celui de répertoire dans le système de fichiers. Par exemple supposons que l'on utilise une BD *XML* native pour stocker des bons de commandes. Dans ce cas il suffit de définir une collection de bons de commandes. Dès lors les requêtes concernant des bons de commandes seront limitées aux documents de cette collection.

Les bases de données *XML* natives fournissent la possibilité de définir des hiérarchies de collection. Supposons maintenant que l'on utilise une BD *XML* native pour stocker des films de cinéma. Dans ce cas on pourrait définir une collection pour les studios de production qui produisent les différents films, et au sein de celle-ci une autre collection pour les films produits.

Certaines bases de données *XML* sont dites indépendantes de schéma. C'est-à-dire qu'elles ne requièrent pas qu'un schéma définisse la structure des collections. Dès lors ces collections peuvent accueillir n'importe quel type de document ce qui les rend extrêmement flexibles et rend le développement d'application plus aisé.

*Les langages d'interrogation XML*

Presque toutes les base de données *XML* natives supportent un ou plusieurs langages de requêtes. Les langage les plus populaire sont XPath (qui possède des extensions pour les requêtes sur des documents multiples) et XQuery. Il existe aussi d'autres langages de requêtes propriétaires. Lors du choix d'une BD *XML* native, les langages de requêtes supportés sont un critère important.

Il existe différents types de langages permettant de manipuler les données d'un document *XML* :

– Les langages d'adressage : ils permettent de naviguer à l'intérieur d'un document à l'aide d'expressions de chemin. Ces expressions de chemin sont souvent

représentées de la même façon que dans les systèmes de fichiers UNIX. Le standard en la matière est le XPath qui sert de base pour les autre langages de requêtes *XML*.

- Les langages de transformation : ils permettent de modifier la structure des documents *XML* à l'aide de règles. Le standard actuel est XSLT.
- Les langages de requêtes : ils permettent de formuler des requêtes complexes pour récupérer des informations contenues dans des documents *XML*, à la manière de SQL pour les BD relationnelles. Dans ce cadre, le XQuery est recommandé par le W3C.

Les langages de requêtes XPath et XQuery feront l'objet d'un chapitre ultérieur. Cependant voici déjà une brève introduction concernant ces langages.

Le but de XPath est de pouvoir adresser les parties d'un document XML, c'est-à-dire qu'il permet de retrouver un ensemble de noeuds de l'arbre associé au document *XML*. C'est délibérément que la notation choisie pour Xpath est plus ou moins celle utilisée pour parcourir les systèmes de fichier de type UNIX, ou celle utilisée pour les URL sur Internet. Ainsi une expression XPath ressemble à ceci :

```
/clients/client[@id ="123"]/nom
```

Ce genre d'expression se lit très facilement lorsque l'on a l'habitude des systèmes de fichiers. Dans ce cas, on voit aisément que l'on veut récupérer l'élément nom, de l'élément client d'identifiant 123, de l'éléments clients.

Le langage XQuery a été conçu pour permettre des requêtes précises et facilement compréhensibles. Il permet de formuler des requêtes basées sur des expressions FLWR (qu'on prononce "*flower*"). Elles sont comparables au *select-from-where* de SQL. XQuery est en fait un langage fonctionnel dont les requêtes sont des expressions de différents types telles que des expressions de chemin (basées sur XPath), des expression FLWR, des expression conditionnelles ou encore des fonctions.

Les expressions FLWR comportent quatre clauses d'où elles tirent leur nom :

- for : permet l'itération sur un ensemble de noeuds.
- let : permet d'assigner des valeurs à des variables.
- where : permet de filtrer les noeuds rassemblé par les clause for et let via des prédicats.
- return : permet de présenter les résultats sous la forme désirée.

Comme déjà mentionné ci-dessus une expression FLWR peut être comparée à une expression select-from-where, si ce n'est que leur fonctionnement respectif est

inversé. Avec SQL le résultat souhaité se construit via la clause select tandis qu'avec une expression, le résultat est construit via la clause return.

A titre d'illustration, observons la requête FLWR suivante qui liste les noms de tous les clients ayant acheté au moins deux produits :

```
<results>
   for $client in //client
   let $produits := //commande[client=$client/@id]//article
   where count ($produits) >= 2
   return <nom>$client/nom/text()</nom>
</results >
```

FIG. 11.11 – *Exemple d'une requête FLWR*

La clause for permet d'itérer sur tous les clients du fichier. Pour chaque client, la clause let va sélectionner la liste de tous les produits achetés par ce client. Ici a lieu une jointure, grâce à un prédicat sur les commandes. En effet, on ne sélectionne que les commandes relatives au client courant, retenu par la clause for. Ensuite, la clause where filtre les résultats, en appliquant la fonction d'agrégation count() sur les résultats de la clause let. Pour finir, on construit le résultat à l'aide de la clause return.

*Modification et suppression*

Les base de données *XML* natives disposent d'une variété de stratégies pour la modification et la suppression de documents, que ce soit par simple remplacement ou suppression du fichier existant, par modification via un arbre DOM [3], ou encore via un langage spécifiant la manière de modifier un document. Les deux standards en la matière sont :

- XUpdate : provenant de XML :DB Initiative. C'est un langage basé *XML* utilisant XPath pour identifier des ensembles de noeuds. Une fois les noeuds identifiés, il permet d'insérer ou de supprimer ces noeuds, ou encore d'insérer de nouveaux noeuds avant ou après les noeuds identifiés.
- Un ensemble d'extensions XQuery. Ces extensions sont proposées par le W3C

*Transactions, verrouillage et concurrence*

Virtuellement toutes les base de données *XML* natives permettent d'effectuer des transactions. Malheureusement le verrouillage s'effectue souvent au niveau du document plutôt qu'au niveau des noeuds. Ceci entraîne un niveau de concurrence

[3] Document Object Modele : API de manipulation de données XML sous forme d'arbre

pour les accès multi-utilisateurs assez faible. Il est donc important de bien choisir les éléments constituant les documents, afin d'obtenir un niveau de granularité suffisant.

Le problème vient du fait que le verrouillage d'une noeud implique aussi le verrouillage de ses noeuds ancêtres. Si on verrouille tout ses ancêtre on va finir par verrouiller le noeud racine, entraînant le verrouillage de tous le document. Cependant ce verrouillage complet est nécessaire, en effet si une transaction doit modifier un noeud feuille, il faut pendant ce temps empêcher qu'une autre transaction puisse supprimer un des parents du noeud feuille, supprimant ainsi le noeud feuille lui-même.

Une solution partielle à ce problème a été apportée par Stijn Dekeyser. Même s'il ne solutionne pas entièrement le problème du verrouillage des ancêtres, il rend les verrous plus flexible en les annotant via la requête. Ceci permet aux autres transactions de déterminer si elles entrent en conflit avec la transaction possédant le verrou.

*Application Programming Interfaces (APIs)*

La plupart des bases de données *XML* natives proposent des APIs de programmation. Elles se présentent généralement sous forme d'interface ODBC. Elles fournissent les moyens de se connecter à la BD, d'explorer les métadonnées, d'exécuter des requêtes et de récupérer les résultats. Ceux-ci sont souvent sous forme de chaînes de caractères XML, d'arbre DOM, d'un SAX Parser ou XML Reader sur le document retourné. Si les requêtes sont capables de retourner plusieurs documents, alors l'API fournit des moyens de parcourir l'ensemble des résultat. Malheureusement la plupart de ces API sont propriétaires. Citons-en quand même deux qui ne le sont pas : XML :DB API de XML :DB.org et JSR 225.

Signalons aussi qu'un grand nombre de BD *XML* natives offrent la possibilité d'exécuter des requêtes et de retourner les résultats en HTTP.

*L'aller-retour des documents*

Une des caractéristiques majeures des BD *XML* natives réside dans le fait qu'elles permettent aux documents de faire des "allers-retours". Ce qui signifie qu'une fois qu'on a stocké un document dans la BD, celle-ci peut nous rendre exactement le même document. Ceci est très important pour les applications basées document, notamment quand le document a une valeur légale ou encore pour les documents médicaux. Toutes les bases de données *XML* natives permettent ces allers-retours au niveau des éléments, des attributs, des PCDATA et de l'ordre du document.

*Index*

Toutes les base de données *XML* natives supportent les index comment moyen d'améliorer la vitesse des requêtes. Il existe trois type d'index :

- Les index de valeur qui indexent le texte et les valeurs des attributs.
- Les index de structures indexant les positions des éléments et des attributs.
- Les index full-text indexant individuellement les mots du document ainsi que les valeurs des attributs.

L'indexation faisant l'objet d'un chapitre suivant, on ne va pas s'étendre sur le sujet maintenant.

## 11.3 Le choix des bases de données XML natives

Après avoir passé en revue les différents types de bases de données permettant de stocker des fichier *XML*, nous allons maintenant passer au choix du type de base de données qui convient le mieux au stockage des descriptions *MPEG-7*. Les bases de données qui nous semblent les plus appropriées sont les bases de données *XML* natives. Dans cette section nous allons présenter les arguments qui nous ont permis de poser notre choix.

Les bases de données relationnelles sont déjà utilisées depuis très longtemps, et leur fiabilité n'est plus à démontrer. Elles semblent donc pouvoir constituer une bonne solution pour stocker les descriptions *XML*. Cependant, elles sont malheureusement conceptuellement inadaptées à la gestion des fichier *XML*. En effet, un document *XML* a une structure d'arbre et n'est généralement pas aussi bien structuré qu'une table dans un système relationnel.

Lorsque l'on veut stocker un fichier *XML* dans une base de données étendue *XML*, la première solution consiste à stocker le fichier entier dans la base de données. On peut soit le stocker dans une cellule d'une table en tant que chaîne de caractère (character large object : CLOB), soit éclater sa structure en différentes tables.

Avec la première technique, il n'est plus possible d'interpréter directement le contenu du fichier *XML* via les requêtes SQL. Or le but de l'application est justement de retrouver les descripteurs qui correspondent au mieux aux descripteurs de références, parmi tous les fichier *MPEG-7*de la base de données.

Alors qu'avec la deuxième technique, l'éclatement de la structure du fichier, le fichier est stocké dans un ensemble de tables spécialement conçues à cette effet.

Avec cette technique le contenu *XML* est accessible via les requêtes SQL. Cependant tous les documents *XML*, même ceux dont les contenus n'ont aucun lien entre eux, sont tous stockés dans les quatre mêmes tables. Donc les fichiers *XML* contenant des descripteurs d'images, de vidéo, de son, ..., se retrouveront dans les mêmes tables. De plus l'ensemble des valeurs du fichier étant stockée dans la même colonne d'une table, elles sont toutes de type chaîne de caractères. Or les descripteurs *MPEG-7* sont exprimés sous forme d'entier, de réel, de tableau, ... , mais une fois stockés dans la base de données, ils ne pourront plus qu'être traités qu'en tant que string. Cela est bien entendus impensable pour une application basée sur *MPEG-7*.

La deuxième solution permettant de stocker des fichiers *XML* dans une des base de données étendue *XML*consiste à éclater les informations contenues dans le fichier en différentes tables, c'est la méthode de mapping. Cette technique apporte différents avantages : elle permet la gestion des données provenant du fichier *XML* dans la base de données, elle permet aux requêtes SQL d'accéder directement aux données stockées.

Une fois stockées dans la base de données via la méthode de shredding (éclatement des informations), les données du fichier *XML* peuvent facilement être récupérées via des requêtes SQL. Une fois les données récupérées, il n'y a plus qu'à les réassembler dans un nouveau fichier *XML* avec la structure désirée, c'est l'étape de publishing.

Il faut cependant bien se rendre compte que cette gestion est efficace si les fichiers *XML* d'entrées et de sorties sont de taille modeste. Lorsque la taille des fichiers d'entrées devient conséquente, les données sont réparties dans un plus grand nombre de tables. De l'autre coté, quand les fichiers de sortie sont de grande taille, cela signifie qu'il y a beaucoup de données à récupérer dans la base de données et à réassembler. S'il y a beaucoup de données à récupérer, les requêtes SQL deviennent complexes à rédiger. De plus, une fois les données récupérées, il faut encore transformer l'ensemble des résultats en format *XML*, ce qui n'est pas toujours aisé et entraîne des pertes de performances. Vu le nombre de descripteurs proposés par le standard *MPEG-7*, la taille des fichiers *XML* manipulés est importante. Une application toute simple utilisant trois ou quatre descripteurs manipule déjà des fichiers de plusieurs dizaines de kilos octets.

Même avec une liste exhaustive des descripteurs qu'une application utilise, la conception d'un schéma relationnel pose toujours de gros problèmes à cause du nombre élevé de descripteurs. Prenons l'exemple d'une application de gestion de photos. Cette application va devoir gérer des types de photo très différents : gros plans, photos de groupe, photos de paysage, photos de bâtiments, d'objets, ... , et chaque type de photos dispose de ses propres descripteurs. Pour stocker ceux-ci soit on définit une table pour chaque type de photo et dans ce cas on va avoir un nombre considérable de table, soit on utilise un table gigantesque pour stocker tous les des-

cripteurs, mais de ce cas chaque colonne se retrouvera avec énormément de valeurs de nulles. En fait il y aura une valeur nulle chaque fois qu'un descripteur est inutile pour décrire la photo.

De plus, la mise en lambeau des documents XML conduit à un autre désavantage. Une fois le document stocké dans la base de données, il est très difficile d'en retrouver l'ordre et la hiérarchie exact. Malheureusement pour certains types de documents, documents à caractère légal ou médical par exemple, il est obligatoire de les retrouver intacts. C'est aussi le cas pour les fichier *MPEG-7*, dans de nombreux cas l'ordre des descripteurs est important et si leur ordre est modifié, ils ne veulent plus rien dire.

Un dernier désavantage des bases relationnelles est qu'elles imposent aux fichiers *XML* d'être très structurés. De plus tous les fichiers doivent avoir la même structure, contrairement aux bases de données *XML* natives. Ceci peut entraîner des pertes de données lors du passage du fichier à la base de données. Par exemple si un fichier contient un élément non attendu, cet élément sera perdu. Sur toute la chaîne de production d'un document audiovisuel, la quantité de descripteurs utilisés, fait qu'il est quasi impossible de les recenser tous. Or l'utilisation de bases de données *XML* natives permet de stocker des fichiers semi-structurés et même des fichiers sans schéma, évitant ainsi la perte d'information.

En ce qui concerne les bases de données orientées objets, à l'heure actuelle, il est difficile de prononcer un verdict concernant leur capacité à stocker des fichier *XML*. En effet, pour l'instant, il n'y a pas encore beaucoup de SGBDOO permettant de stocker des fichiers *XML*. De prime abord le modèle orienté objet paraît être une bonne solution car il est assez proche de la structure des fichiers *XML*. Cependant, étant plus jeunes que leurs homologues relationnelles, les bases de données orientés objet n'offrent pas encore vraiment toutes les garanties que l'on peut attendre d'une base de données. Il manque notamment encore un langage commun et standardisé de requête. Mais il est clair que ce type de base de données va encore se développer. Il serait donc judicieux de garder un oeil sur leur développement futur.

Au vu des problèmes que les bases de données relationnelles ont à stocker des fichiers *XML* et de la jeunesse des bases de données orientées objet, notre choix pour stocker les descriptions *MPEG-7*se porte sur les bases de données *XML* natives. Conçues pour le XML, les bases natives exploitent pleinement sa richesse. Elles stockent le document dans son intégralité qui peut ainsi servir de preuve légale.

Dans un système relationnel, les données doivent s'adapter au modèle relationnel assez rigide. A l'inverse les BD *XML* natives sont plus flexibles. De plus elles supportent plus facilement les modifications de schéma que leur homologues relationnelles, elle permettent même de manipuler des données qui n'ont aucun schéma particulier.

Un autre avantage des BD *XML* natives, c'est qu'elles gèrent des données semi-structurées, c'est-à-dire que la structure des documents stockés ne doit pas être rigoureusement identique d'un document à l'autre. Cette propriété est très utile pour stocker les descriptions *MPEG-7*.

Presque toutes les bases de données *XML* natives supportent un ou plusieurs langages de requêtes. Notamment le XQuery, langage conseillé par le W3C.

Pour une explication plus détaillée sur les bases de données *XML* natives et pourquoi elles sont appropriées au stockage des document *XML*, nous vous invitons à consulter le site de Ronald Bourret consacré à ce sujet. Ce site est se trouve sur l'adresse suivante : http://www.rpbourret.com/index.htm

## 11.4 Références

Afin de réaliser ce chapitre nous nous sommes inspirés des documents suivants : [BL05], [Bou02], [Mod06], [BMRC], [SBC+04], [Ver06], [Cat05], [WK03].

CHAPITRE 12

# L'indexation et la recherche de document

Afin de permettre une recherche rapide et pertinente de documents multimédia, il est nécessaire que nos deux systèmes disposent d'un moteur d'indexation et de recherche. Afin de mieux comprendre leurs rôles, essayons tout d'abord de comprendre comment un moteur de recherche de page Web référence et permet de retrouver une page Web. Le référencement et la recherche du site **www.le-mpeg7.be** peuvent être décrits par quatre processus illustrés à la figure 12.1. Les flèches en pointillé représentent les actions effectuées par le moteur d'indexation et le moteur de recherche.

1. Il est nécessaire tout d'abord de déclarer le site **www.le-mpeg7.be** sur le site du moteur de recherche.
2. Dès que le site est enregistré, un robot (appelé aussi spider) va récupérer les informations qu'il considère comme significatives sur les différentes pages du site.
3. Ensuite le moteur d'indexation va analyser les informations, les classer, les faire correspondre au site qu'elles décrivent (ici **www.le-mpeg7.be**) et enfin les enregistrer dans un dictionnaire inversé, c'est-à-dire un dictionnaire indexé par les définitions.
4. Une requête par mot-clef `MPEG7` est analysée par le moteur de recherche. Ce dernier recherche ensuite dans le dictionnaire inversé les différents sites indexés par le mot `MPEG7` et les classe par ordre de pertinence suivant des algorithmes bien précis.

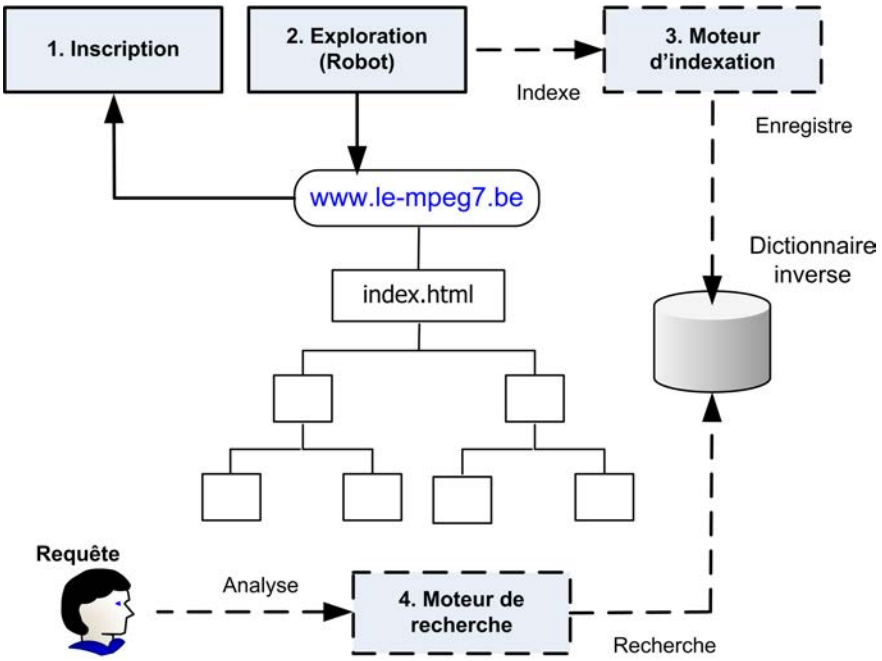


FIG. 12.1 – *Fonctionnement d'un moteur de recherche de page Web*

Un moteur d'indexation et de recherche remplit donc les fonctions suivantes : l'indexation de l'information, le stockage des indexes dans un dictionnaire, l'analyse des requêtes des utilisateurs et la recherche dans le dictionnaire. Nos deux architectures comprennent chacune un moteur d'indexation et un moteur de recherche, ces derniers rempliront les mêmes fonctions.

Essayons maintenant d'imaginer un outil de recherche de document multimédia sans moteur d'indexation et de recherche. Dans ce cas, un utilisateur désirant rechercher des photos de "Panda" devrait parcourir l'ensemble des photos stockées dans le système jusqu'à ce qu'il trouve ce qu'il désire. Si la base de données de photo est conséquente, la recherche pourrait durer plusieurs jours. On comprend donc assez aisément l'utilité d'un moteur de recherche et d'indexation. Avec un tel système, il suffit à l'utilisateur de questionner le système à l'aide d'une requête utilisant le mot-clef "Panda". Pour ce faire, lors du stockage du document multimédia, le moteur d'indexation parcourt la description du document et la stocke dans un format optimisé pour la recherche.

Nous allons maintenant analyser les deux moteurs de recherche et les deux moteurs d'indexation que nous avons intégrés dans nos solutions. Il s'agit pour la solution "Lucene" de Lucene et LIRE. Pour la solution "full-*XML*", il s'agit du moteur de requête XQuery et du moteur d'indexation de la base de donnée *XML* Native.

# 12.1 Lucene

Lucene est une interface de programmation (API) permettant d'indexer et de rechercher du texte. C'est un projet open source de la fondation Apache mis à disposition sous Licence Apache.

## 12.1.1 Format de stockage des indexes

La qualité d'un moteur de recherche tient principalement dans le temps de réponse d'une recherche. Plus les résultats s'affichent rapidement, meilleure est la qualité du moteur. Une des conditions essentielles pour avoir des réponses rapides est d'avoir un bon format de stockage des index.

Comme l'illustre la figure 12.2, Lucene repose sur 4 éléments clés :

– l'index contenant une série de documents ;
– le document représenté par une suite de champs ;
– le champ qui est un nom associé à une séquence de termes ;
– le terme qui est une chaîne de caractères.

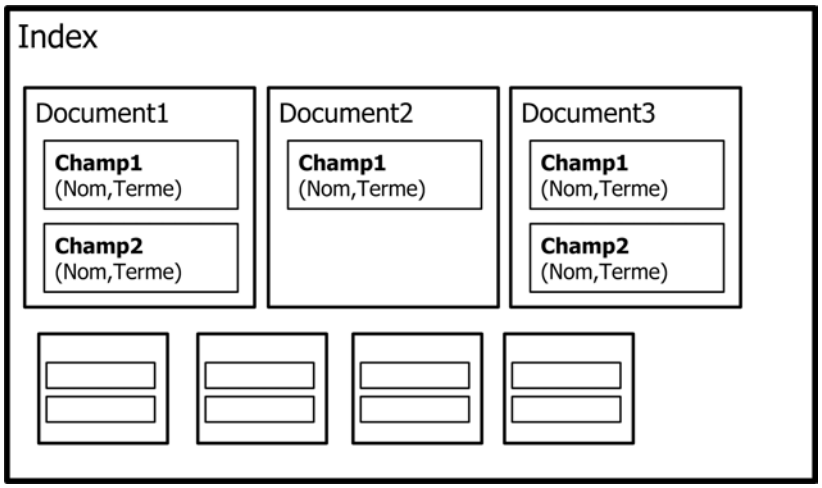


FIG. 12.2 – *Les éléments clés de Lucene*

Un champ est donc représenté comme un couple `(Nom,Terme)`, où `Nom` est une chaîne de caractères représentant le nom du champ et `Terme` est une chaîne de caractères du champ. Cela signifie que deux termes similaires se trouvant dans des champs différents seront considérés comme différents (figure 12.3).

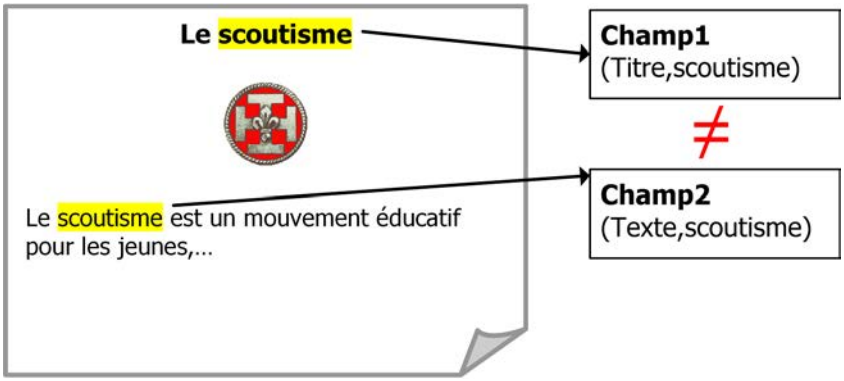


FIG. 12.3 – *Champs différents comportant le même terme*

L'index de Lucene fait partie de la famille des index inversés. En effet, Lucene liste, pour un terme donné, les documents qui contiennent ce mot. "inversé" signifie que cette relation va en sens inverse de la relation naturelle : un document liste des termes.

Dans Lucene, les champs peuvent être stockés. Dans ce cas, leur texte (séquence de termes) est stocké littéralement dans l'index et ce de manière non inversée. Notons qu'un champ peut être à la fois indexé et stocké.

Pour être indexé, Un texte peut aussi être tokenizé, c'est-à-dire découpé en liste de mots dans laquelle on a éventuellement retiré les mots n'ayant pas de réelle signification (déterminants, signes de ponctuation, ...).

Un index Lucene peut être décomposé en sous-index appelés **segments**. Bien qu'il soit un sous-ensemble d'un index, un segment est lui-même un index indépendant sur lequel il est possible d'effectuer une recherche. L'évolution des index se fait donc par rajout de nouveaux segments pour les documents ajoutés dernièrement ou par fusion de segments existants.

Un segment est composé par les éléments suivant :

- Le nom des champs utilisés dans l'index (*Field Names*).
- La liste des champs stockés (c'est-à dire, pour chaque document, les couples (Nom,Terme) stockés). Les champs stockés sont utilisés pour stocker de l'information supplémentaire sur un document (le titre, l'emplacement physique, ...). L'ensemble des champs stockés d'un document représente l'information qui est renvoyée lorsque le moteur de recherche retrouve le document (*Stored Field Values*).
- Le dictionnaire des termes (*Term Dictionnary*) contenant l'ensemble des termes des champs indexés de chaque document. Le dictionnaire indique aussi le nombre de documents associés à chaque terme ainsi que la position où ils se

trouvent sur chaque document.
– Les documents effacés (*Deleted Documents*) qui constituent un fichier optionnel indiquant les documents supprimés.

Nous utiliserons Lucene dans notre architecture “Lucene” pour indexer les descriptions sémantiques. Une fois le formulaire de description complété par l'utilisateur, le système enregistre au format *XML MPEG-7* les différentes informations descriptives et les transmet à Lucene. En plus des informations descriptives, le système transmet à Lucene l'emplacement de stockage du fichier *MPEG-7*. En effet, les champs représentant les informations descriptives ne seront pas stockés littéralement dans l'index, seul le champ correspondant à l'emplacement du fichier *MPEG-7* le sera. Lors d'une recherche d'une image, le moteur de recherche renverra l'emplacement des fichiers *MPEG-7* correspondant aux documents trouvés dans l'index. Pour afficher les images trouvées et leur description, le système devra encore rechercher et traiter[1] le document *MPEG-7*.

La création du document Lucene se fait de la manière suivante :

1. Un document Lucene est créé. Celui-ci peut être vu comme le formulaire de description de l'application.
2. Pour chaque champ du formulaire de description, on crée un champ Lucene auquel on associe l'information contenue dans le champ du formulaire.
3. L'emplacement du fichier *MPEG-7* correspondant au document Lucene est stocké littéralement dans l'index.
4. Le document est ajouté à l'index.

[1]Il existe des outils tels que le XSLT (Extensible Style Language Transformations) permettant la transformation d'un fichier *XML* en un autre format (PDF, RTF, HTML, ...).

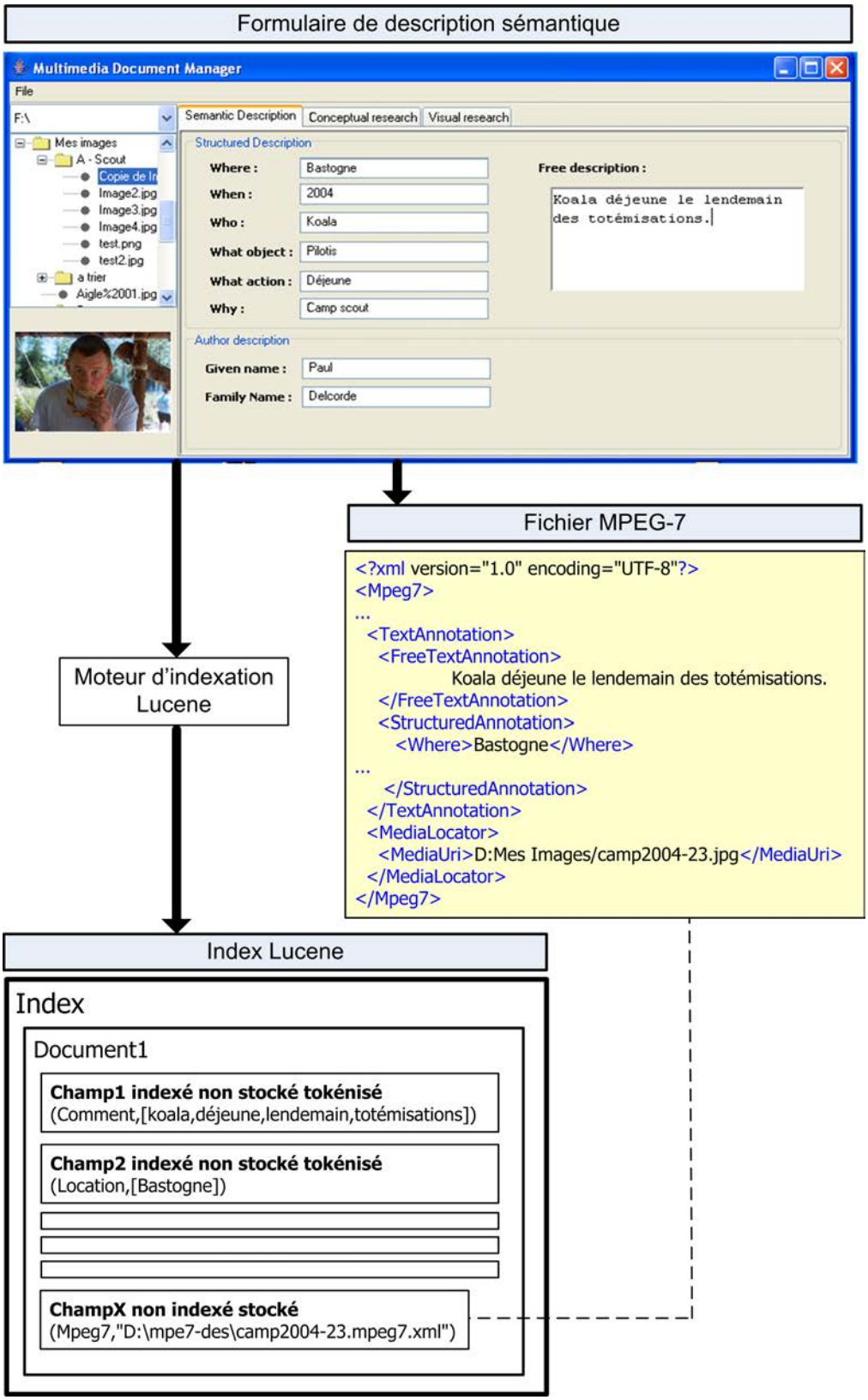


FIG. 12.4 – *Création d'un index Lucene*

### 12.1.2 La recherche

Avant d'effectuer une recherche, il est nécessaire de savoir au préalable ce que l'on désire trouver, c'est ce qui est spécifié dans la requête. L'API de Lucene permet à l'utilisateur de créer ses propres requêtes. Toutefois, Lucene propose aussi un langage de requêtes relativement riche et une interpréteur de requêtes (`Query Parser`). Dans cette partie, nous ne verrons qu'un simple aperçu des possibilités offerte par Lucene en termes de recherche. Pour de plus amples informations, nous vous conseillons de vous rendre sur le site de Lucene[2].

Une requête Lucene est décomposée en termes et opérateurs. Les termes sont de deux types :

– les termes simples qui sont des mots uniques tels que "mpeg7" ou "standard" ;
– les phrases qui sont des groupes de mots comme "mpeg7 est un standard".

Afin de créer des requêtes plus complexes, Lucene permet de combiner plusieurs termes ensemble à l'aide d'opérateurs booléens.

Une recherche peut se faire soit sur un champ par défaut soit sur un champ spécifique. Pour spécifier un champ, il suffit d'inscrire le nom du champ suivi du symbole " :" et du terme à rechercher. Prenons comme exemple (illustré à la figure 12.5) un index Lucene contenant un champ par défaut `auteur`, un champ `citation` et un champ stocké `url`.

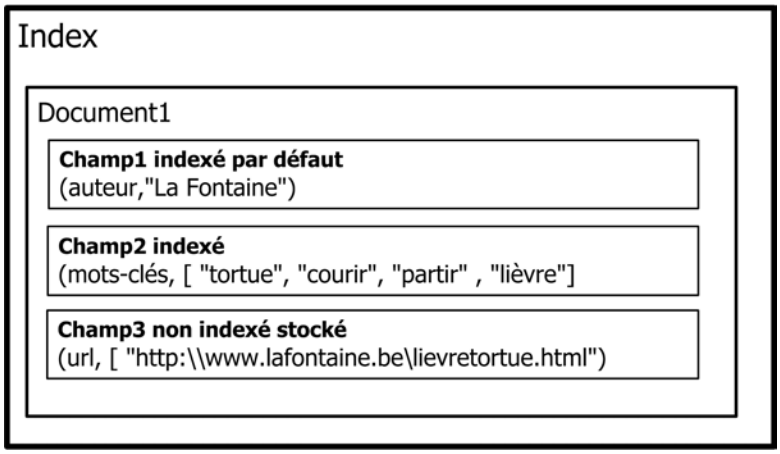


FIG. 12.5 – *Création d'un index Lucene*

Pour retrouver l'adresse du site internet contenant le poème de Jean de La Fontaine "*Le Lièvre et la Tortue*", un utilisateur peut utiliser les requêtes illustrées à la figure 12.6.

[2]**http ://lucene.apache.org/**

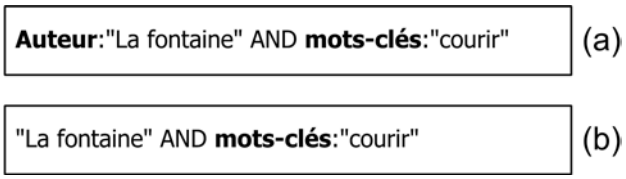


FIG. 12.6 – *Exemple de requêtes Lucene valide*

Les requêtes (A) et (B) sont identiques. En effet, vu que le champ *auteur* est un champ par défaut, il n'est pas nécessaire de l'indiquer.

Lucene comporte un large éventail d'options de recherche à l'aide de modificateurs de termes. On peut citer entre autres :

– la recherche par joker ;
– la recherche floue ;
– la recherche de proximité ;
– la recherche d'intervalle.

Comme nous l'avons déjà mentionné plus haut, Lucene permet la combinaison de termes à l'aide de connecteurs logiques. Les opérateurs supportés sont **AND**, **OR**, **NOT**, "**+**" et "**-**".

La figure 12.7 illustre le fonctionnement d'une recherche sémantique d'une image avec Lucene.

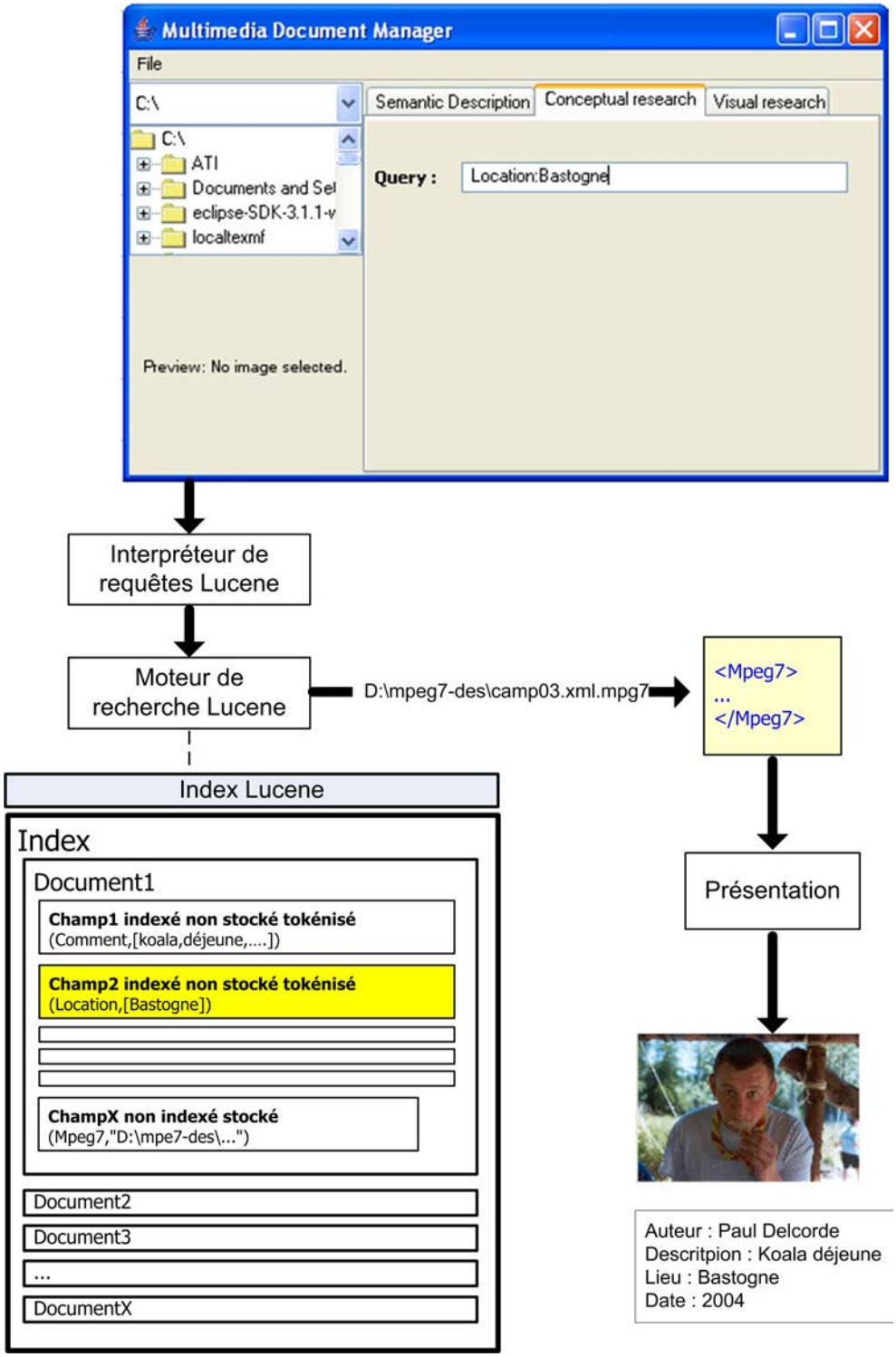


FIG. 12.7 – *Recherche sémantique d'une image avec Lucene*

## 12.2 LIRE

LIRE (Lucene Image REtrieval) est une API, construite autour de Lucene, permettant l'indexation et la recherche d'image en se basant sur des descripteurs visuels. LIRE possède un outil d'analyse d'images et permet la production des descripteurs *MPEG-7* visuels suivant : ScalableColor, ColorLayout et EdgeHistogram[3]

Lors de l'indexation d'une image, LIRE calcule en premier lieu les différents descripteurs visuels puis les enregistre dans un index de type Lucene sous forme de champs stockés. Dans notre architecture "Lucene", la création des descripteurs visuels se fait automatiquement lorsque l'utilisateur choisit son image à indexer. Une fois la description sémantique validée par l'utilisateur, le système stocke la description sémantique dans un fichier *MPEG-7*, envoie les informations sémantiques à Lucene pour être indexées et transmet l'image à LIRE pour réaliser l'indexation visuelle.

LIRE utilisera un index différent de celui de Lucene. Nous ajouterons à l'index de LIRE un champ stocké `mpeg7` indiquant l'emplacement des fichiers *MPEG-7* correspondant aux images indexées.

Pour la recherche d'une image par similarité[4], LIRE calcule tout d'abord les descripteurs visuels de l'image de recherche puis effectue un algorithme de similarité avec les descriptions visuelles des images se trouvant dans son index. Une fois l'algorithme effectué, LIRE retourne les coefficients de similarité des images indexées par rapport à l'image de recherche.

## 12.3 L'indexation des bases de données XML natives

### 12.3.1 Introduction

Un des avantages majeurs des bases de données *XML* natives est leur capacité à indexer les documents *XML* qu'elles contiennent. Si les index sont correctement utilisés ils permettent de réduire le temps requis pour exécuter des requêtes. Si la base de données n'est pas indexée, le gestionnaire de base de données n'a pas d'autre choix que de parcourir chaque document un par un, tandis que si la base de données est correctement indexée, le gestionnaire de base de données peut retrouver

[3]Voir Le *MPEG-7* : 3.2 Le *MPEG-7* Visual, B - Les descripteurs de couleur.
[4]Voir La production de descripteurs : 10.2, Les descripteurs visuels

un sous-ensemble de documents correspondant à celui de la recherche, ou du moins réduire considérablement le nombre de documents à vérifier. En général, on indexe les éléments les plus fréquemment interrogés par les requêtes.

Les langages de requêtes XML comme XPath ou XQuery utilisent des expressions de chemin pour parcourir les documents *XML*. Le document *XML* ayant une structure logique d'arbre, un chemin permet d'accéder à un noeud ou à un ensemble de noeuds de l'arbre, correspondant à un élément du document. Par exemple l'expression `entreprise//departement/employe` permet de sélectionner tous les noeuds de type `employe`, ayant comme parent un noeud de type `departement` et comme ancêtre un noeud de type `entreprise`, et uniquement ceux-là. Dans l'expression du chemin, la double slash exprime qu'il doit y avoir un chemin entre un noeud de type `entreprise` vers un noeud de type `departement`, une relation ancêtre-descendant, alors que la simple slash exprime une relation parent-enfant. Mais il est à noter que XPath définit d'autres types de relations entre les noeuds comme frère-suivant et frère précédant.

L'ensemble des noeuds sélectionnés par une expression de chemin peut encore être réduit en utilisant des prédicats. Un prédicat (entre crochets dans un chemin) est une expression qui peut prendre une valeur de vérité, vrai ou faux. Les noeuds ayant une valeur fausse pour le prédicat sont exclus de la sélection. Par exemple `entreprise//departement/employe[contains(adresse,'Namur')]` permet de sélectionner les noeuds de type `employe` dont l'`adresse` contient Namur.

La première expression de chemin (celle sans prédicat) effectue une sélection structurelle, tandis que la deuxième effectue une sélection basée sur des valeurs. Celles-ci peuvent être des noms d'élément ou d'attribut, des valeurs d'élément ou d'attribut. Les sélections structurelles sont quant à elles basées sur les relations entre les noeuds tels que parent-enfant, ancêtre-descendant, etc.

Une expression de chemin du type `entreprise//departement` pose un gros problème. Face à une requête contenant un tel chemin, le gestionnaire de la base de données va partir du noeud de type `entreprise` et va parcourir tous les chemins descendant de ce noeud pour essayer de trouver tous les descendants de type `departement`. Il va donc parcourir des chemins qui ne contiennent pas de noeuds de type `departement`.

Au travers de cet exemple, on se rend compte de l'importance d'indexer les bases de données, surtout pour les grandes collections de documents. Le système d'indexation devra permettre de s'occuper des sélections basées sur les valeurs ainsi que les sélections structurelles. Pour l'indexation des valeurs on peut utiliser un système classique tel que les *B tree*. Par contre pour l'indexation structurelle, les choses sont plus compliquées. En effet le système d'indexation devra être capable d'identifier les relations entre les noeuds. Si l'indexation est correctement effectuée, pour recher-

cher une information il ne faudra quasi plus jamais passer par le document *XML*, mais par l'index.

### 12.3.2 Les type d'indexation

#### *A - L'indexation basée sur les valeurs*

Les gestionnaire de BD utilisent généralement des *B-tree* pour organiser les informations d'indexage. La figure 12.8 montre les trois éléments constitutifs d'un *B-tree* :

- Un noeud racine qui contient des pointeurs vers des noeuds branches.
- Des noeuds branches qui contiennent des pointeurs vers des noeuds feuilles ou branches.
- Des noeuds feuilles qui contiennent des index et des pointeurs horizontaux vers d'autres feuilles.

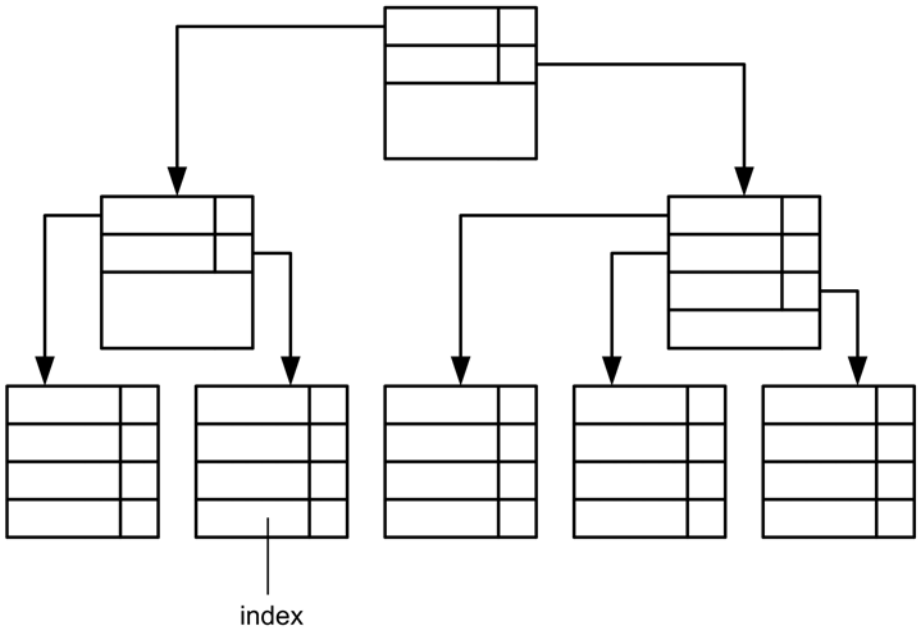


FIG. 12.8 – *Exemple de B-tree*

L'index est l'unité fondamentale de l'arbre d'index. Il contient une valeur clé correspondant à l'élément du document *XML* indexé, ainsi que la localisation de cet élément. Un noeud quant à lui est une page d'index contenant un ensemble d'index.

Quand on crée un index pour une collection vide, le gestionnaire de BD crée le noeud racine qui restera vide jusqu'a ce que la collection se remplisse de documents. Au début le noeud racine joue le même rôle que les noeuds feuilles. Pour chaque document entré dans la collection, le gestionnaire va créer et insérer un index dans le noeud racine.

Quand le noeud racine est rempli d'index, le gestionnaire va répartir les index du noeud dans deux nouveaux noeuds comme suit :

– Il crée deux noeuds feuilles.

– Il réparti la moitié des index dans chacun des noeuds feuilles.

– Il met à jour les pointeurs du noeud racine vers le noeuds feuilles.

Les index contenus dans les noeuds branches et racines correspondent à la plus grande valeur d'index du noeud fils que pointe le pointeur de l'index, comme illustré à la figure 12.9.

Au fur et à mesure que l'on ajoute des documents dans la collection, le gestionnaire va remplir les noeuds feuilles avec les nouveaux index. Dès qu'un noeud feuille est plein, le gestionnaire crée un nouveau noeud feuille, déplace une partie des index dans le nouveau noeud et ajoute un pointeur dans le noeud racine, comme illustré à la figure 12.9.

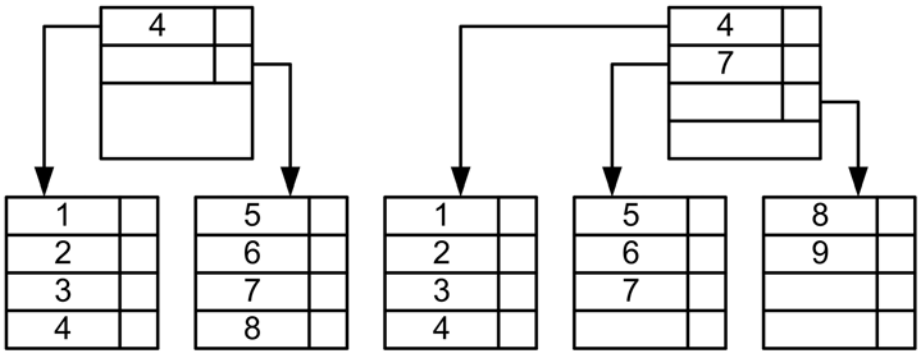


FIG. 12.9 – *Division d'un noeud feuille*

Quand le noeud racine sera rempli (il contiendra quatre feuilles dans notre exemple), et que ses feuilles seront toutes remplies, lorsque le gestionnaire voudra ajouter un nouvel index, s'il crée une nouvelle feuille, il ne pourra pas ajouter un pointeur de plus dans le noeud racine pointant sur cette nouvelle feuille.

Dans ce cas, le gestionnaire va créer deux noeuds branches et les remplir chacun avec la moitié des index du noeud racine et mettre à jour les pointeurs. Au fur et à mesure que la collection va grandir, les noeuds vont être divisés, l'arbre aura de

plus en plus d'étages. Dès lors un simple ajout peut entraîner une restructuration complète de l'arbre. Ceci peut sembler être un défaut de la technique, mais c'est aussi un sérieux avantage. En effet grâce à ce mécanisme, l'arbre est toujours plus ou moins bien équilibré.

### *B - Indexation structurelle*

Pour ce qui concerne l'indexation structurelle, la technique est tout à fait différente. Dans ce cas on utilise un plan de numérotation. De nombreuses recherches ont été effectuées dans ce domaine, aboutissant à différentes techniques. Un plan de numérotation assigne à chaque noeud de l'arbre du document un identifiant. Ces identifiants fournissent un mécanisme pour pouvoir identifier rapidement les relations structurelles entre une paire de noeuds. Pour faire simple, un plan de numérotation doit pouvoir permettre d'effectuer les deux opérations suivantes :

- La décision : c'est une opération permettant de savoir quelle est la relation structurelle entre deux noeuds : frère-précédent, frère-suivant, parent-enfant, ancêtre-enfant.
- Le reconstitution : c'est une opération qui permet, pour un noeud donné, de déterminer quels sont les identifiants de ses noeuds voisins : son père, ses frères, ses enfants, etc.

Afin d'illustrer ce qu'est un plan de numérotation, nous allons présenter le plan de Boëhme et Rahm utilisé par eXist. Ce plan est nommé "numérotation dynamique par niveau" ou DLN. Les identifiants de noeuds, les "nombres dynamiques de niveau", sont basés sur le système de classification décimale de Dewey, utilisé pour classer les livres dans les bibliothèques. Ces identifiants sont représentés par une suite de nombres séparés par des caractères spéciaux et ils sont hiérarchiques. La racine se voit attribuer l'identifiant "1". Ensuite l'identifiant de chaque noeud est composé de l'identifiant de son noeud parent auquel on ajoute un numéro d'ordre. Ce système est illustré à la figure 12.10. Avec ce plan de numérotation, les opérations de décision et de reconstitution deviennent élémentaires.

## 12.3.3 L'indexation avec Berkeley DB XML

Vu que chaque base de données *XML* native dispose plus ou moins de son propre système d'indexation, il est dès lors difficile d'expliquer de façon générale la manière dont sont implémentés les mécanismes d'indexation des BD *XML* natives.

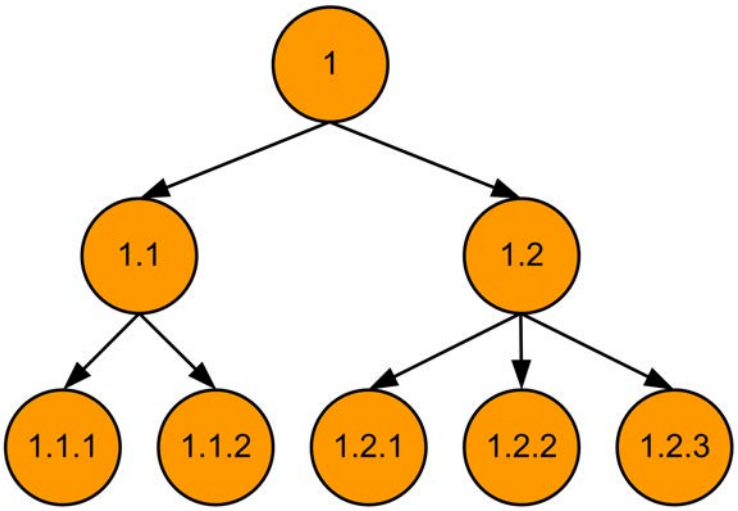


FIG. 12.10 – *Plan de numérotation DLN*

C'est pourquoi nous vous proposons de s'attarder un peu sur les mécanismes d'indexation de la base de données Berkeley DB XML ou BDB XML.

BDB XML est une base de données *XML* native open source supportant le langage de requête XQuery. Elle a été spécialement conçue pour le stockage et la récupération d'une grand nombre de documents *XML*. BDB XML s'utilise au moyen d'une API (disponible pour JAVA et C++) permettant la gestion, la modification et l'exécution de requête. Elle fournit un mécanisme d'indexation flexible et robuste pouvant améliorer grandement les performance des requêtes. L'indexation des containers se fait via l'API qui fournit un ensemble de méthodes permettant la gestion des index.

Les types d'index fournis par BDB XML sont définis par les quatre informations suivantes :

- Uniqueness
- Path Types
- Node Types
- Key Types

*Uniqueness*

Uniqueness spécifie si la valeur de l'index doit être unique ou non au sein du container. Par défaut, les valeurs d'index ne sont pas uniques. Pour qu'elles le soient, il faut le spécifier lors de la déclaration de l'index.

*Path Types*

Le path type (type chemin) permet de retrouver un noeud spécifique des documents de la collection plus rapidement. On peut soit indexer directement le noeud avec le path type `node`, soit indexer une portion de chemin menant au noeud via le path type `edge`.

```
<vendor type="wholesale">
    <name>TriCounty Produce</name>
    <address>309 S. Main Street</address>
    <city>Middle Town</city>
    <state>MN</state>
    <zipcode>55432</zipcode>
    <phonenumber>763 555 5761</phonenumber>
    <salesrep>
        <name>Mort Dufresne</name>
        <phonenumber>763 555 5765</phonenumber>
    </salesrep>
</vendor>
```

Sur base de l'exemple ci-dessus provenant de [Cat05], supposons que l'on veut indexer le noeud `name`. Il y a deux endroits où le noeud `name` apparaît. Si on utilise le path type `node`, le noeud `name` nécessitera deux entrées dans l'index, chacune ayant des valeurs différentes (TriCounty Produce et Mort Dufresne). Dès lors les requêtes basées sur le noeud `name` devront examiner les deux entrées.

C'est pour cela que dans le cas où plusieurs noeud ont le même nom il vaut bien mieux utiliser le path type `edge`. Si on considère les deux chemins `/vendor/name` et `salesrep/name` comme index, il n'y a plus de double entrée dans l'index.

Dès lors on utilisera le type `node` uniquement pour indexer un noeud dont le nom ne revient qu'une seule fois dans les documents et le type `edge` pour un noeud dont le nom est utilisé dans plusieurs contextes différents.

*Node Types*

Via les node types (types noeud) on peut indexer les valeur des éléments, des attributs ou des métadonnées. Cela permet d'accroître les performances des requêtes qui testent les valeurs des éléments, des attributs ou des métadonnées. Les noeuds de métadonnées se trouvent seulement dans les métadonnées du document, il ne peut donc pas utilisé le path type `edge` pour l'indexer.

*Key Types*

Les key types permettent de spécifier le type des tests dont l'indice fera l'objet. Il en existe trois types :

- equality : améliore les performances des tests comparant des noeuds avec une valeur spécifique
- presence : améliore les performances des tests vérifiant l'existence sans égard pour sa valeur.
- substring : améliore les performances des tests vérifiant si la valeur d'un noeud contient une chaîne de caractères

Dans le cadre de notre application, la figure 12.11 donne un exemple de document qu'il faut indexer. Pour indexer ce type de document, il faut utiliser le node type pour indexer le descripteurs sémantiques et le path type pour les descripteurs visuels.

En ce qui concernent les descripteurs sémantiques, leur valeur est comparée avec les mots clés de la recherche. C'est pourquoi il faut utiliser le node type pour les indexer ainsi qu'un key type en fonction du test que l'on veut effectuer sur ces mots clés.

Pour les descripteurs visuels, il est impossible de les comparer directement avec le descripteur de la recherche en utilisant les opérateur classique. Il faut donc utiliser le path type. Comme les descripteurs visuels ont des noms bien distincts, on va utiliser le path type node.

## 12.4 La recherche via XQuery

La recherche d'information dans une base de données *XML* native se fait via des requêtes exprimées en XQuery. Le langage XQuery a été conçu pour permettre des requêtes précises et facilement compréhensibles. Il permet de formuler des requêtes basées sur des expressions FLWR (qu'on prononce "*flower*"). Elles sont comparables au *select-from-where* de SQL. XQuery est en fait un langage fonctionnel dont les requêtes sont des expressions de différents types tels que des expressions de chemin (basées sur XPath), des expressions FLWR, des expressions conditionnelles ou encore des fonctions.

Les expressions FLWR comportent quatre clauses d'où elles tirent leur nom :

- for : permet l'itération sur un ensemble de noeuds.

```
<?xml version="1.0" encoding="UTF-8"?>
<Mpeg7>
 <Creator>
      <GivenName>Paul</GivenName>
      <FamilyName>Delcorde</FamilyName>
 </Creator>

<!-- Descripteurs sémentique -->
 <TextAnnotation>
  <FreeTextAnnotation>
           Koala déjeune le lendemain des totémisations.
  </FreeTextAnnotation>
  <StructuredAnnotation>
    <Where>Bastogne</Where>
    <When>2004</When>
    <Who>Koala </Who>
    <WhatObject>Pilotis</WhatObject>
    <WhatAction>Dejeune</WhatAction>
    <WhatAction>Camp scout</WhatAction>
   </StructuredAnnotation>
 </TextAnnotation>

<!-- Descripteurs visuels -->
 <VisualDescriptors>
  <ColorLayout>
      <YDCCoeff>17</YDCCoeff>
      <CbDCCoeff>16</CbDCCoeff>
      <CrDCCoeff>32</CrDCCoeff>
      <YACCoeff63>14 13 17 ... 15 16 15</YACCoeff63>
      <CbACCoeff63>16 15 18 ... 15 16 15</CbACCoeff63>
      <CrACCoeff63>13 14 14 ... 16 15 16</CrACCoeff63>
   </ColorLayout>
   <ScalableColor numOfBitplanesDiscarded="0" numOfCoeff="256">
      <Coeff>-202 71 27 ... 1 3 1</Coeff>
   </ScalableColor>
   <DominantColorType>
      <SpatialCoherency>0</SpatialCoherency>
      <Value>
        <Percentage>8</Percentage>
        <Index>10 10 10</Index>
      </Value>
      <Value>
        <Percentage>5</Percentage>
        <Index>10 7 5</Index>
      </Value>
      <Value>
        <Percentage>3</Percentage>
        <Index>21 25 26</Index>
      </Value>
      <Value>
        <Percentage>2</Percentage>
        <Index>27 28 26</Index>
      </Value>
   </DominantColorType>
 </VisualDescriptors>
 <MediaLocator>
  <MediaUri>D:Mes Images/camp2004-23.jpg</MediaUri>
 </MediaLocator>
</Mpeg7>
```

FIG. 12.11 – *Exemple de description basée MPEG-7*

– let : permet d'assigner des valeurs à des variables.

– where : permet de filtrer les noeuds rassemblé par les clauses for et let via des prédicats.

– return : permet de présenter les résultats sous la forme désirée.

Comme déjà mentionné ci-dessus une expression FLWR peut être comparée à une expression select-from-where, si ce n'est que leur fonctionnement respectif est inversé. Avec SQL le résultat souhaité se construit via la clause select tandis qu'avec une expression, le résultat est construit via la clause return.

A titre d'illustration, observons la requête FLWR suivante qui liste les noms de tous les clients ayant acheté au moins deux produits :

```
<results>
   for $client in //client
   let $produits := //commande[client=$client/@id]//article
   where count ($produits) >= 2
   return <nom>$client/nom/text()</nom>
</results >
```

FIG. 12.12 – *Exemple d'une requête FLWR*

La clause for permet d'itérer sur tous les clients du fichier. Pour chaque client, la clause let va sélectionner la liste de tous les produits achetés par ce client. Ici a lieu une jointure, grâce à un prédicat sur les commandes. En effet, on ne sélectionne que les commandes relatives au client courant, retenu par la clause for. Ensuite, la clause where filtre les résultats, en appliquant la fonction d'agrégation count() sur les résultats de la clause let. Pour finir, on construit le résultat à l'aide de la clause return.

Le Xquery a pas mal de cordes à son arc. Il dispose notamment d'un quantificateur existentiel : `some` et d'un quantificateur universel : every. Ce dernier n'existe malheureusement pas en SQL.

`some` *var* `in` *expr* `satisfies` *predicate*
`every` *var* `in` *expr* `satsifies` *predicate*

Comme cela a déjà été vu, l'ordre des éléments d'un document *XML* peut avoir de l'importance. C'est pourquoi XQuery fournit des opérateurs d'ordre permettant de savoir si un élément est avant ou après un autre. Ces deux opérateurs sont : « et ».

Il faut faire cependant attention au fait que ces opérateurs d'ordre n'ont rien à voir avec la clause `order by` qui trie les résultats sur base d'un ou pour plusieurs paramètres. Elle a le même comportement que son équivalent SQL.

XQuery étant un langage fonctionnel, il permet de définir et d'utiliser de nouvelles fonctions. Elle s'écrivent comme à la figure 12.13. Une fonction peut en appeler une autre ou s'appeler elle-même. Ces fonctions sont très pratiques dans le cadre de notre application. En effet elle permettent de pouvoir voir directement si deux histogrammes de couleurs sont identiques par exemple.

```
define function nom_fonction ($var1 as type1, $var2 as type2, ...)
as type_retour
{
...
}
```

FIG. 12.13 – *Prototype d'une fonction en XQuery*

Quant au traitement des requêtes XQuery, il y en a deux types : le traitement en flux ou le traitement sur base des index. Le traitement en flux se produit lorsqu'une requête utilise des éléments non indexés. Comme le moteur de requêtes ne dispose d'aucune information sur la structure du fichier, il doit traiter celui-ci comme flux de données, qui ne peut être que lu séquentiellement et qu'une seule fois. Par contre si les collections sont correctement indexées, le moteur de requêtes dispose d'informations sur la structure du fichier. Ces informations lui permette d'accéder directement aux éléments du fichier dont il a besoin.

Malheureusement comme beaucoup de langages, XQuery souffre de quelques limitations. Bien qu'il soit conçu pour traiter des documents *XML*, il ne respecte pas la syntaxe *XML*.

Un autre gros point négatif pour XQuery est qu'il ne dispose pas de fonctions ni d'opérateurs adaptés à la recherche full-text dans un document *XML*. Il n'y a par exemple aucune fonction définie permettant de faire des recherches insensibles à la casse, ni de fonctions avancées de manipulation de chaîne de caractères (trim, split,...). Cela n'empêche que certaines BD *XML* ont implémenté elles-mêmes ces fonctions.

Pour finir XQuery manque aussi d'opérateurs d'ajout et de mise à jour de données. Il ne permet que de consulter les données, ce qui est un très vilain défaut dans le monde des bases de données. Certains moteurs qui utilisent XQuery, proposent des extensions pour pouvoir effectuer ces opérations, mais rien de standardisé pour l'instant. Sinon on peut toujours utiliser le langage XUpdate défini à cette usage.

## 12.5 Références

Afin de réaliser ce chapitre nous nous sommes inspirés des documents suivants : [BR04], [IBM05], [WK03].

# Conclusion

Un premier objectif de ce mémoire était de présenter un état de l'existant des standards de compression. Le but était de se familiariser avec la vidéo numérique qui fait partie intégrante de nos vies. Au travers des différents standards *MPEG-1*, *MPEG-2* et *MPEG-4*, nous avons pu constater les progrès phénoménaux qui ont été accomplis dans ce domaine.

Au vu de l'utilisation massive de la vidéo numérique dans de nombreux domaines, mais aussi au vu du travail de numérisation considérable qui va devoir être accompli pour sauvegarder notre patrimoine audiovisuel qui se meurt en silence, il nous semblait évident, dans le cadre de ce mémoire, de proposer une description des standards de compression vidéo les plus utilisés de par le monde.

Après la présentation de la numérisation, le deuxième objectif de ce mémoire était de présenter la manière de décrire le contenu des documents multimédia. Tout au long du chapitre consacré au *MPEG-7*, nous avons pu découvrir toutes sortes de descripteurs destinés à la description de contenu. Le but de ce chapitre était de montrer l'importance de l'utilisation d'un standard de description dans le monde du multimédia. L'utilisation de ce standard par le plus grand nombre permet d'avoir une interopérabilité forte entres les différents terminaux et applications utilisés de par le monde dans des domaines comme celui de la radiodiffusion, du montage vidéo, de la conservation d'archives, etc.

Au delà de l'utilisation généralisée d'un standard de description, réside le problème de l'échange de documents et de leurs métadonnées servant à la description du contenu. Nous avons découvert grâce à *MXF* un moyen d'échanger des documents audiovisuels tout au long de leur chaîne de production. Le *MPEG-21* quant à lui nous a proposé des solutions pour faire respecter les droits et permissions associés aux oeuvres numériques.

La dernière partie, consacrée à l'indexation de documents via des descripteurs *MPEG-7* et au stockage de ces derniers, avait pour objectif de proposer des solutions aux divers problèmes que l'on peut rencontrer dans l'élaboration d'une application de gestion de banque d'images.

Le chapitre 9 proposait deux types d'applications : une basée sur Lucene et une autre sur l'utilisation des bases de données *XML* Natives. Au chapitre suivant, nous avons exposé les moyens d'extraire différents descripteurs (de façons manuelle et automatique) des images à indexer.

Le chapitre 10 était quant à lui consacré à présenter les différentes solutions possibles pour stocker les descripteurs *MPEG-7*, qui, s'il est besoin de le rappeler, utilisent le langage *XML*. Après avoir jugé chacune des solutions, nous avons motivé notre choix quant à l'adoption des bases de données *XML* natives.

Une fois le choix du type de base de données posé, nous avons consacré le dernier chapitre aux techniques d'indexation des bases de données *XML* Natives ainsi qu'aux techniques d'indexation utilisées par Lucene. Ce dernier chapitre fut aussi l'occasion d'expliquer comment récupérer les données stockées via Lucene ainsi que celles stockées dans les bases de données *XML* native via XQuery.

Dès le début du développement du Multimédia Document Manager, il nous a paru évident que cette application se devait de proposer une recherche par mots-clés via des descripteurs sémantiques. Malheureusement toutes les images ne peuvent pas se décrire uniquement via des mots-clés, nous avons donc ajouté des fonctionnalités permettant la recherche d'images via une autre image de référence. Cette recherche est basée sur l'utilisation des descripteurs visuels. Grâce à eux, notre application est capable de retrouver des images similaires à une image de référence, confirmant par la même occasion l'utilité de *MPEG-7*. Nous avons également appris que même s'il y avait toujours besoin d'une intervention humaine pour encoder les descripteurs sémantiques, on pouvait utiliser des algorithmes pour extraire automatiquement des descriptions visuelles.

A la fin de cette dernière partie, il nous semble qu'une bonne application de gestion d'images devrait utiliser les descripteurs *MPEG-7*et une base de données *XML*native pour stocker ceux-ci. L'application devra permettre l'extraction de caractéristiques de bas niveau automatiquement et permettre à l'utilisateur de pouvoir décrire ses images via une série de mots-clés. Cela permettrait d'introduire les caractéristiques de haut niveau. Vu que l'application va utiliser les descripteurs visuels de *MPEG-7* pour extraire les caractéristiques de bas niveau, il serait judicieux que l'application puisse proposer des méthodes de recherche via ces descripteurs. L'utilisateur pourrait alors donner une image ou un croquis comme élément de référence pour sa recherche. Une autre méthode de recherche, qui serait très efficace, consisterait à allier la recherche par mots-clés à la recherche par descripteur visuel. Ceci permettrait d'éliminer certains problèmes liés à la recherche de mots-clés. Le principal problème de la recherche par mots-clés vient de la subjectivité de la personne qui introduit ces mots-clés.

Pour une gestion plus efficace de la recherche par mots-clés, l'application pourrait se doter d'un thésaurus. Ceci permet de palier aux problèmes de subjectivité du langage naturel (ambiguïté, synonymie, homonymie, ...) en proposant par exemple uniquement les mots du thésaurus pour pouvoir décrire les images.

Même si notre application se limite à la gestion des images, elle nous a permis de soulever une partie des problèmes que l'on rencontrait lors du développement d'une application plus générale permettant de gérer des documents audiovisuels. Elle nous a montré notamment l'importance de l'utilisation des descripteurs *MPEG-7* et des bases de données *XML* natives dans ce domaine.

# Bibliographie


[AD05] M. AMIELH et S. DEVILLERS : Text of ISO/IEC 21000-17 FCD MPEG-21 FID (Fragment Identification of MPEG Resources). Rapport technique, MPEG, 2005.

[All98] Jean-François ALLOUIS : MPEG7 et la révolution numérique de l'audio-visuel. Rapport technique, Ministère de la culture et de la communication (France), 1998.

[Bac00] Bruno BACHIMONT : L'indexation automatique — Enjeux, possibilités et limites. Site Internet, 2000. Accessib via http://www.culture.gouv.fr/culture/dglf/rifal/indexation.htm (Dernière visite 5/8/2006).

[Bac01] Bruno BACHIMONT : Indexation et gestion de contenus audiovisuels et multimédia — Un point de vue issu de l'INA. Rapport technique, Institut National de l'Audiovisuel, 2001.

[Bel05] A. BELAÏD : Les techniques de compression MPEG. Cours en ligne, 2005. Accessible via http://www.loria.fr/~abelaid/Enseignement/sid/cours3-video-mpeg.pdf (Dernière visite 17/01/2006).

[Ber04] J.P. BERNOUX : Indexation de contenus audio-visuels. Rapport technique, TDF, 2004.

[BL05] Rajesh R. BORDAWEKAR et Christian A. LANG : Analytical Processing of XML Documents : Opportunities and Challenges. Rapport technique, IBM T. J. Watson Research Center, 2005.

[BMRC] Berkeley Multimédia Research Center (BMRC) : Berkeley MPEG Tools. Site Internet. Accessible via

http://bmrc.berkeley.edu/frame/research/mpeg/toc.html (Dernière visite 17/01/2006).

[Bou02] Ronald BOURRET : XML and Database. Site Internet, 2002. Accessible via http://www.rpbourret.com (Dernière visite 25/08/2006).

[BR04] Timo BÖHME et Erhard RAHM : Supporting Efficient Streaming and Insertion of XML Data in RDBMS. Rapport technique, University of Leipzig, Germany, 2004.

[Cat05] Sleepycat SOFTWARE : *Getting Started with Berkeley DB XML for Java*. Sleepycat Software, http://www.sleepycat.com/products/bdbxml.html, 2005. (Dernière visite 25/08/2006).

[Cha05] Gilles CHAGNON : Cours de XML - Initiation aux Schema XML, 2005. Accessible via http://gilles.chagnon.free.fr/cours/xml/schema.html (Dernière visite 20/07/2006).

[Con04] Cyril CONCOLATO : Introduction à la norme MPEG-21. Rapport technique, Ecole nationale supérieur des télécommunications, 2004.

[Cor98] Jean-Michel CORNU : Guide de l'information numérique — Comment traiter les données lisibles par machine et les documents numériques. Rapport technique, DLM-Forum, 1998.

[Cou99] P. COURTELLEMON : Cours de Multimédia. Cours en ligne, 1999. Accessible via http://perso.univ-lr.fr/pcourtel/espadon/site_web (Dernière visite 24/01/2006).

[Cre01] Sébastien CRESPIN : La compression des images animées. Site Internet, 2001. Accessible via http://screspin.free.fr/mpeg/ (Dernière visite 17/01/2006).

[Dan05] Jean DANIEL : Le Signal Vidéo Numérique. Site Internet, 2005. Accessible via http://pages.videotron.ca/danjean/ (Dernière visite 17/01/2006).

[Dev02] Bruce DEVLIN : MXF— the Material eXchange Format. Rapport technique, Snell & Wilcox, 2002.

[Dev04] Bruce DEVLIN : MXF Overview. Rapport technique, Snell & Wilcox, 2004.

[Dev06] Sylvain DEVILLERS : MPEG-21 Systems — File Format & DI Streaming. Rapport technique, France Télécom R&D, 2006.

[DJ05] Gerrard DRURY et Thomas-Kerr JOSEPH : Text of ISO/IEC CD 21000-18. Rapport technique, MPEG, 2005.

[Don04] Anthony DON : Indexation Vidéo et Graphes — Interfaces de visualisation des contenus vidéo indexés basées sur les graphes. Rapport technique, LaBRI, Université de Bordeaux I, 2004.

[EPM05] Ecole Polytechnique de MONTRÉAL : Les descripteurs visuels MPEG7, 2005. Disponible via (Dernière visite 14/07/2006).

[EVV00] Eric van der VLIST : Introduction aux schémas W3C XML Schema., 2000. Accessible via http://xmlfr.org/documentations/tutoriels/001218-0001 (Dernère visite 26/07/2006).

[Gas05] Philippe GASSER : MPEG-1 et MPEG-2. Rapport technique, MSH Paris nord - Plate forme Arts, Sciences, Technologies, 2005. Accessible via http://plate-forme-ast.mshparisnord.org/MPEG-1-et-MPEG-2 (Dernière visite 24/01/2006).

[GG96] Jacques-Alexandre GERBER et Sébastien GIGNOUX : La compression vidéo MPEG. Ecole des Mines de Nancy / Exposé de cours en ligne, 1996. Accessible via http://www.mines.inpl-nancy.fr/~tisseran/I33_Reseaux/compression.video/mpeg1.htm (Dernière visite 17/01/2006).

[GL04] Roland GWINNER et Sébastien LALAURETTE : Le standard MPEG-7. Rapport technique, 2004.

[Goo01] Phil GOOD : MPEG et DVB. Site Internet, 2001. Accessible via http://iphilgood.chez-alice.fr/ (Dernière visite 24/01/2006).

[HAL03] HAL : Les normes MPEG — Mpeg1, Mpeg2, Mpeg4, Mpeg7 et Mpeg21. Sujet en ligne sur le forum Hardware.fr, 2003. Accessible via http://forum.hardware.fr/hardwarefr/VideoSon/TraitementVideo/liste_sujet-1.htm (Dernière visite 17/01/2006).

[Hoo05] Emmanuel HOOG : Une mémoire audiovisuelle qui s'estompe en silence, 2005.

[IBM05] IBM : Structure of B-Tree Index Pages. Site Internet, 2005. Accesible via http://publib.boulder.ibm.com/infocenter/idshelp/v10/index.jsp?topic=/com.ibm.adref.doc/adref235.htm (Dernière visite 27/08/2006).

[ISO02a] INTERNATIONAL ORGANISATION FOR STANDARDISATION (ISO) : Current Vision on Event Reporting in MPEG 21. Rapport technique, MPEG : Requirements Group, 2002.

[ISO02b] INTERNATIONAL ORGANISATION FOR STANDARDISATION (ISO) : MPEG-4 Overview - (Version 10), 2002. Accessible via http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm (Dernière visite 22/07/2006).

[ISO03] INTERNATIONAL ORGANISATION FOR STANDARDISATION (ISO) : Requirements for ISO/IEC TR21000-12 Multimedia Test Bed for Resource Delivery. Exigence iso/iec, MPEG, 2003.

[ISO04a] INTERNATIONAL ORGANISATION FOR STANDARDISATION (ISO) : MPEG-7 Overview - (V.21 – Jeju Version), 2004. Accessible via

http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm (Dernière visite 05/08/2006).

[ISO04b] INTERNATIONAL ORGANISATION FOR STANDARDISATION (ISO) : Multimedia framework (MPEG-21) — Part 1 — Vision, Technologies and Strategy. Standard ISO/IEC 21000-1 :2004(E), MPEG, 2004.

[ISO04c] INTERNATIONAL ORGANISATION FOR STANDARDISATION (ISO) : The reference conformance test model. Rapport technique, MPEG, 2004.

[ISO05a] INTERNATIONAL ORGANISATION FOR STANDARDISATION (ISO) : Multimedia framework (MPEG-21) — Part 14 — Conformance Testing. Rapport technique, MPEG, 2005.

[ISO05b] INTERNATIONAL ORGANISATION FOR STANDARDISATION (ISO) : Multimedia framework (MPEG-21) — Part 16 — Binary Format. Standard ISO/IEC 21000-16 :2005(E), MPEG, 2005.

[ISO05c] INTERNATIONAL ORGANISATION FOR STANDARDISATION (ISO) : Multimedia framework (MPEG-21) — Part 2 — Digital Item Declaration. Standard ISO/IEC 21000-2 :2005(E), MPEG, 2005.

[Ive03] John IVE : The Material eXchange Format and the Workflow Revolution. Rapport technique, Professional MPEG Forum, 2003.

[Jeo06] Sangoh JEONG : Histogram-Based Color Image Retrieval. Rapport technique, Stanford Center for Image Systems Engineering, 2006.

[KM04] Wong KA MAN : Content Based Image Retrieval Using MPEG-7. Rapport technique, CITY UNIVERSITY OF HONG KONG, 2004.

[KNL04] Frederik DE KEUKELAERE, Wesley DE NEVE, Peter LAMBERT, Boris ROGGE et Rik Van de WALLE : MPEG-21 DIGITAL ITEM PROCESSING ARCHITECTURE. Rapport technique, Ghent University : Department of Electronics and Information Systems, 2004.

[Lin06] James LING : Lucene Tutorial, 2006. Disponible via http://www.ittopics.com (Dernière visite 20/08/2006).

[LUC06] The Apache Software FOUNDATION : Apache Lucene, 2006. Accessible via http://lucene.apache.org/java/docs/index.html (Dernière visite 24/08/2006).

[Lut02] Benoit LUTTRINGER : Présentation de MPEG4, 2002. Accessible via http://membres.lycos.fr/benoitluttringer/Annexe_Mpeg4.html (Dernière visite 19/07/2006).

[Lux06] Mathias LUX : LIRE, 2006. Accessible via http://www.semanticmetadata.net/lire/ (Dernière visite 24/08/2006).

[Mod06] Vinay MODI : Color Descriptors from Compressed Images. Rapport technique, University of Edinburgh, School of Informatics Homepage, 2006.

[MOV01] B. S. MANJUNATH, Jens-Rainer OHM et Vinod V. VASUDEVAN : Color and Texture Descriptors. Rapport technique, IEEE, 2001.

[MPG] MPEG : The MPEG Home Page. Site Internet. Accessible via http://www.chiariglione.org/mpeg (Dernière visite 17/01/2006).

[MPGF98]Hélène WATERS : The Professional-MPEG Forum. Site Internet, 1998. Accessibble via http://www.pro-mpeg.org/index.html.

[MXF] MXF : MXF — An MXF Primer. Site Internet. Accessible via http://www.mxf.info.

[OAM02] Timo OJALA, Markus AITTOLA et Esa MATINMIKKO : Empirical Evaluation of MPEG-7 XM Color Descriptors in Content-Based Retrieval of Semantic Image Categories. Rapport technique, University of Oulu, Finland, 2002.

[PCD03] Gabriel PEYRÉ, Ismael CASTILLO et Charles DOSSAL : La compression d'images par ondelettes, 2003. Accessible via http://www.cmap.polytechnique.fr/~peyre/java/image/.

[SBC+04] Bart STEEGMANS, Ronald BOURRET, Owen CLINE, Olivier GUYENNET, Shrinivas KULKARNI, Stephen PRIESTLEY, Valeriy SYLENKO et Ueli WAHLI : *XML for DB2 Information Integration*. IBM, 2004. Version électronique accessible via : http://www.redbooks.ibm.com/.

[Sir04] Xavier SIRVEN : Authenticité et accessibilité des archives électroniques — MUSTICA, Le cas de la création musicale numérique. Rapport technique, Université de technologie de Compiègne, 2004.

[Tor06] Fabien TORRE : Cours sur l'écriture de XML Schémas, 2006. Accessible via http://www.grappa.univ-lille3.fr/~torre/guide.php?id=coursxmlschema (Dernière visite 18/07/2006).

[UL00] Université de LYON 2 : Le traitement des images, 2000. Accessible via http://theses.univ-lyon2.fr/didacticiel/unite2/module4.html (Dernière viste 15/08/2006).

[Var03] Jean VARRA : Le stockage de la vidéo et de l 'audio. Rapport technique, Institut Nationel de l'Audiovisuel (INA), 2003.

[Ver06] Boris VERHAEGEN : *Requêtes OLAP sur une base de données XML native*. Thèse de doctorat, Université libre de Bruxelles, 2006.

[Vin00] Laurent VINET : Les archives télévisuelles à l'heure du numérique — L'indexation automatique pour la documentation audiovisuelle, 2000.

[Wan06] Lidan WANG : Clustering WWW Image Search Results Using Color Histogram and Textual Information. Rapport technique, CS838 Term Project Report, 2006.

[Wik06] WIKIPÉDIA : Compression vidéo. Article de Wikipédia (encyclopédie en ligne), 2006. Accessible via http://fr.wikipedia.org/wiki/Compression_MPEG (Dernière visite 17/01/2006).

[WK03] Utz WESTERMANN et Wolfgang KLAS : An Analysis of XML Database Solutions for the Management of MPEG-7 Media Descriptions. Rapport technique, University of Vienna, 2003.

[WRG99] P. H. WESTERINK, R. RAJAGOPALAN et C. A. GONZALES : Two-pass MPEG-2 variable-bit-rate encoding. Rapport technique, IBM Journal of research and development, Juin 1999. Accessible via http://www.research.ibm.com/journal/rd/434/westerink.html (Dernière visite 20/07/2006).

[XML06a] World Wide Web Consortium (W3C) : XML Schema 1.1 Part 1 : Structures, 2006. Accessible via http://www.w3.org/TR/xmlschema11-1/ (Dernière visite 18/07/2006).

[XML06b] World Wide Web Consortium (W3C) : XML Schema 1.1 Part 2 : Datatypes, 2006. Accessible via http://www.w3.org/TR/xmlschema11-2/ (Dernière visite 18/07/2006).