LES ENTREPOTS DE DOCUMENTS : GESTION DES VERSIONS

Kaïs KHROUF (*), **Mohamed MBARKI** (**), **Franck RAVAT (**), Chantal SOULE-DUPUY (**), Nathalie VALLES-PARLANGEAU (**)**
khrouf.kais@isecs.rnu.tn, mbarki@irit.fr, ravat@irit.fr, soule@irit.fr, nathalie.valles-parlangeau@univ-tlse1.fr

(*) MIRACL – ISECS, Université de Sfax, Route El Meharza km1.5, BP n° 1030, 3018 Sfax – Tunisie
(**) IRIT – Equipe SIG/D2S2, Université Toulouse I et III, 118 Route de Narbonne, 31062 Toulouse Cedex – France

Mots clefs :
Entrepôts de documents, gestion de versions, modèle générique étendu.
Keywords:
Document warehouses, versioning, extended generic model.
Palabras clave :
Almacenes de documentos, gestión de versiones, modelo genérico extendido.

Résumé

Les entrepôts de documents doivent permettre le stockage de documents hétérogènes, sélectionnés et filtrés, ainsi que leur classification selon des structures logiques génériques. Une telle organisation des entrepôts permet de faciliter l'exploitation des informations documentaires intégrées au travers plusieurs techniques complémentaires, à savoir : la recherche d'information, l'interrogation des données et l'analyse multidimensionnelle. Cependant, les documents intégrés dans l'entrepôt sont très dynamiques et changent de contenu dans le temps. Par conséquent, les entrepôts de documents doivent garder et gérer les différentes versions des documents. Ce papier présente un modèle générique étendu d'entrepôts permettant l'historisation et la gestion des versions de documents.

1 Introduction

Actuellement, si l'intuition reste fondamentale dans le processus de prise de décision, elle ne suffit plus, il faut aussi pouvoir prendre ce que les anglophones appellent « la décision informée » (informed décision). A cette fin, les entreprises disposent aujourd'hui d'importants volumes d'informations qui sont généralement hétérogènes : données structurées (bases de données, fichiers…), données semi-structurées et multimédia (XML, HTML…).

Vu cette hétérogénéité des données, il est nécessaire de les contrôler, les homogénéiser, les organiser, les intégrer et enfin les stocker pour donner à leur utilisateur une vue intégrée et orientée métier ; ces informations factuelles et aussi textuelles constituent un facteur de savoir faire et de connaissances. Ainsi, face à l'augmentation considérable du nombre de documents gérés par les entreprises, le besoin d'outils pour traiter automatiquement les informations documentaires s'est rapidement fait sentir par les entreprises.

Dans ce contexte, un nouveau concept est apparu permettant de gérer cette masse importante d'informations, à savoir : l'entrepôt de documents. C'est « *une source d'informations orientées-sujets, filtrées, intégrées, historisées et organisées comme support d'un processus de recherche, d'interrogation ou d'analyse.* » [7]. D'après cette définition, les données d'un entrepôt doivent être organisées par sujets, jugées pertinentes, intégrées en provenance de multiples sources et historisées, c'est à dire garder leur évolution dans le temps, ainsi que leurs différentes versions ; ce dernier point fera l'objet de ce papier.

Ce papier se décompose comme suit. La section 2 décrit les travaux existants pour la gestion des versions des documents. La section 3 présente le contexte de nos travaux. Plus précisément, elle décrit le modèle générique proposé d'entrepôts de documents. Dans la section suivante, nous présentons l'extension que nous proposons à ce modèle pour tenir compte de la gestion des versions de documents. Enfin, nous décrivons les modifications apportées à notre outil DOCWARE (DOCument WAREhouse).

2 Etat de l'art

Un des problèmes liés à Internet est qu'il permet à l'information de circuler et d'évoluer de manière totalement aléatoire. Ces changements rapides et à priori imprédictibles posent donc le problème de leur détection : un utilisateur visitant des documents de manière répétée peut vouloir être informé de la manière dont le document a été modifié depuis sa dernière visite. Il s'agit donc de pouvoir détecter les changements dus à ces évolutions, en comparant les anciennes et les nouvelles versions du document, et en évaluant leur similarité. Dans ce contexte, plusieurs outils et travaux ont été proposés dans la littérature tels que : XML TreeDiff [4], XyDiff [3], X-Diff [15], [10] et [12].

XML TreeDiff [4] réalisé par IBM, est un ensemble de modules permettant la différentiation et la mise à jour des arbres DOM [14], similaires aux outils diff et patch de différentiation et de mise à jour des fichiers de données classiques. Cependant, au lieu de différencier les représentations des fichiers des documents XML, XML TreeDiff utilise leurs DOM.

XyDiff [3] est un composant de Xylème [1] permettant de gérer les différentes versions d'un même document. Tout élément modifié est alors représenté comme un fichier XML, indexé et stocké dans un entrepôt de données. Ces fichiers sont ensuite utilisés pour reconstruire une version antérieure d'un document. XyDiff utilise la structure arborescente des documents XML afin de détecter les mouvements et les changements qui interviennent sur un document.

X-Diff [15] est un algorithme permettant d'intégrer les caractéristiques principales de structures XML avec les techniques standards de comparaison d'arbres afin de calculer les différences entre deux versions d'un document XML. La principale caractéristique de cet algorithme est que les documents XML sont représentés sous formes de structures arborescentes non-ordonnées, contrairement aux travaux de XML TreeDiff et de XyDiff.

[10] représente un document sous la forme d'un arbre d'éléments (ensemble de fragments indépendants les uns des autres). L'évolution d'un document concerne l'évolution du document dans sa globalité, ainsi que les fragments qui le composent. Les auteurs distinguent ainsi deux types de versions : une version de document et une version de fragment. En fait, la modification de certains fragments du document entraîne la création de nouvelles versions de fragments, ainsi qu'une nouvelle version du document.

[12] propose une approche pour l'extraction des règles à partir des changements que subissent les versions des documents XML dynamiques. Plus précisément, les auteurs proposent un algorithme qui étudie le comportement des versions de documents XML dans le temps et détermine ainsi des règles d'apprentissage afin de prévoir les changements des documents dans le futur.

Dans le cadre de nos travaux, nous nous sommes intéressés non seulement à la gestion des versions des documents (suivre et détecter l'évolution des changements d'un document dans le temps), mais aussi à la gestion des versions des collections de documents (ensembles de documents regroupés sous formes de classes). Nous distinguons ainsi deux types de gestion de versions, à savoir : gestion des versions de documents et celle des versions de structures génériques. Avant de détailler notre approche, nous présentons dans ce qui suit le contexte de nos travaux.

3 Contexte de nos travaux

Les entrepôts de documents doivent constituer une source d'informations synthétique et homogène [2]. A cette fin, nous avons proposé un modèle générique permettant d'intégrer des documents issus de sources disséminées et hétérogènes et facilitant leur exploitation [7]. La figure 1 présente le modèle générique proposé.

Ce modèle générique comporte les composants suivants :

- la structure logique générique : est caractérisée par les trois méta-classes « Str_Gen » pour les structures logiques génériques, « Elts_Gen » pour les éléments génériques, et « Atts_Gen » pour les attributs génériques. C'est une structure dont relève toute une classe de documents. Elle est définie par un ensemble d'éléments génériques pouvant être composés d'autres éléments génériques et/ou décrits par des attributs génériques.
- la structure logique spécifique : est caractérisée par les classes « Documents », « Déclarations », « Elts_Spec » pour les éléments spécifiques et « Atts_Spec » pour les attributs spécifiques. Elle correspond à un et un seul document, elle est définie par un ensemble d'éléments spécifiques pouvant englober des attributs spécifiques.
- Le contenu : est caractérisé par les autres classes « Informations » (contenu textuel d'un élément spécifique), « Mots-clés » (issus de la phase d'indexation) et « Index ». C'est la description des informations documentaires associées aux éléments de la structure logique spécifique. Il s'agit d'extraire les termes descriptifs en se basant sur des techniques issues de l'indexation automatique de texte [13].

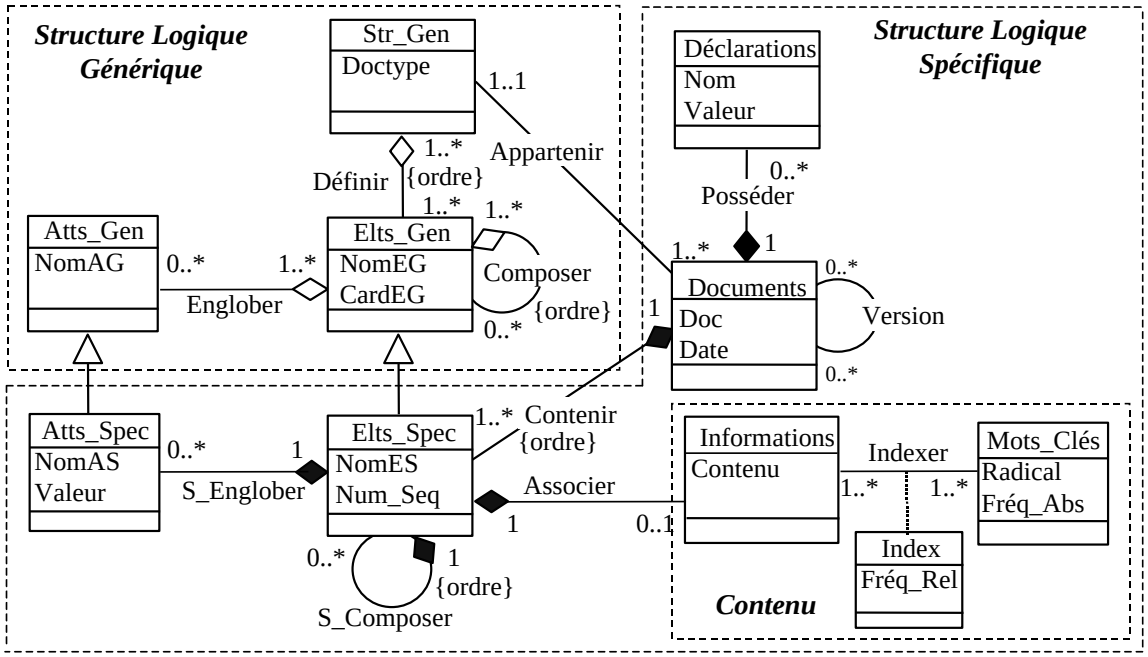


Figure 1 : Modèle générique d'entrepôts de documents

Exemple :

```
<Livre>
<Titre>SGBD</Titre>
<Auteur>J.Dupont</Auteur>
<Chapitre>Introduction
générale…</Chapitre>
<Chapitre>Le langage sgbd
est…</Chapitre>
</Livre>
```

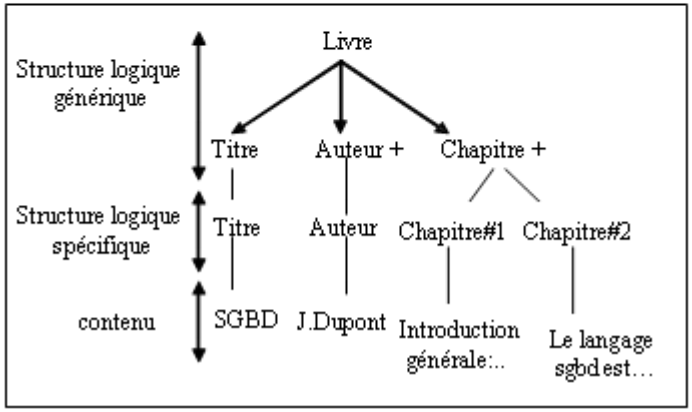


Figure 2 : Composants d'un document intégré dans l'entrepôt

Cette décomposition en structures logiques génériques et spécifiques permet de regrouper, sous forme de classes, les documents ayant des structures similaires sans pour autant perdre leurs caractéristiques spécifiques [9]. Chaque document intégré dans l'entrepôt doit posséder une structure spécifique et il sera rattaché à une structure générique. Ceci permet de préserver la généricité de notre modèle.

L'intérêt d'un tel modèle générique réside dans :

- l'intégration de tout type de documents caractérisés par une absence totale ou partielle de structure [9].
- le regroupement des documents selon des structures communes identiques ou approchantes. Dans [6], nous avons défini une approche basée sur le calcul de similarité (ressemblance) d'arborescences hétérogènes d'éléments ordonnés et étiquetés.
- l'application des techniques d'analyse multidimensionnelle aux informations documentaires. Ces techniques consistent à considérer un sujet analysé comme un point dans un espace à plusieurs dimensions au travers la construction des magasins de documents. Ces magasins permettent l'analyse et la visualisation des informations de l'entrepôt sous forme de tables multidimensionnelles [8].

Remarque : un outil appelé DOCWARE (DOCument WAREhouse) [8] a été réalisé permettant l'intégration des documents dans l'entrepôt tout en les regroupant selon des structures communes identiques ou approchantes. Cet outil permet aussi d'analyser les documents intégrés d'une manière multidimensionnelle.

Les entrepôts de documents doivent permettre aussi la persistance des documents jugés pertinents. Les informations issues du Web par exemple sont très volatiles, c'est à dire qu'elles évoluent rapidement et peuvent disparaître, être modifiées ou encore changer de localisation fréquemment. D'après le modèle générique proposé, un document possède une ou plusieurs versions (l'association réflexive « Version »). Mais, chaque version sera considérée et stockée comme étant un nouveau document. Dans le cadre de ce papier, nous nous somme intéressés à l'intégration et à la gestion des versions de documents au niveau des entrepôts.

4 Gestion des versions

4.1 Problématique

Les documents intégrés dans l'entrepôt peuvent subir plusieurs changements au cours du temps. Ces changements sont de natures différentes (ajout de nouvelles informations, modification, suppression de certaines données ou les trois à la fois). Chacune de ces actions donne naissance à une nouvelle version. Au niveau des entrepôts de documents, il serait intéressant non seulement de garder la version actualisée d'un document mais aussi ses différentes versions antérieures, de manière à mettre en évidence uniquement les parties modifiées entre deux versions successives.

Le modèle générique proposé jusqu'à maintenant traite chaque document à intégrer indépendamment des documents de l'entrepôt même s'il s'agit d'une nouvelle version. En effet, le contenu d'une nouvelle version sera inséré entièrement dans l'entrepôt même s'il est très similaire au contenu de la version précédente. De plus, les changements que peuvent subir un document peuvent concerner soit son contenu, soit sa structure, soit les deux à la fois.

Pour pallier ces insuffisances, nous proposons dans la section suivante une extension du modèle générique permettant d'une part d'intégrer les liens de partages [5] entre les éléments non modifiés des différentes versions et d'autre part de gérer les changements de contenu et les changements structurels [11].

- Changement de contenu : lorsque l'ancienne et la nouvelle version ont la même structure logique et seulement le contenu qui change.
- Changement structurel : lorsque la structure logique de la nouvelle version du document est différente de celle de l'ancienne version.

Exemple : Soient les trois versions suivantes du « Doc.xml ».

Doc.xml(Version 1)

```
<?XML Version="1.0">
<Doctype Article[…]>
<Article>
<Titre>XML</Titre>
<Editeur>Roger</Editeur>
<Contenu>les
Balises…</Contenu>
</Article>
```

Doc.xml(Version 2)

```
<?XML Version="1.0">
<Doctype Article[…]>
<Article>
<Titre>XML</Titre>
<Editeur>Roger</Editeur>
<Contenu>le langage xml…
</Contenu>
</Article>
```

Doc.xml(Version 3)

```
<?XML Version="1.0">
<Doctype Livre[…]>
<Livre>
<Titre>XML</Titre>
<Chapitre>Introduction
à…</Chapitre>
<Résumé>Les
Techniques… </Résumé>
</Livre>
```

Nous remarquons que les deux versions Version1 et Version 2 de « Doc.xml » ont la même structure logique spécifique, seul le contenu de la balise Contenu a été modifié (Changement de contenu). Alors que la version 3 a une structure logique différente des deux premières versions (Changement structurel). Dans notre exemple, les parties non modifiées seront partagées entre les versions du document « Doc.xml ».

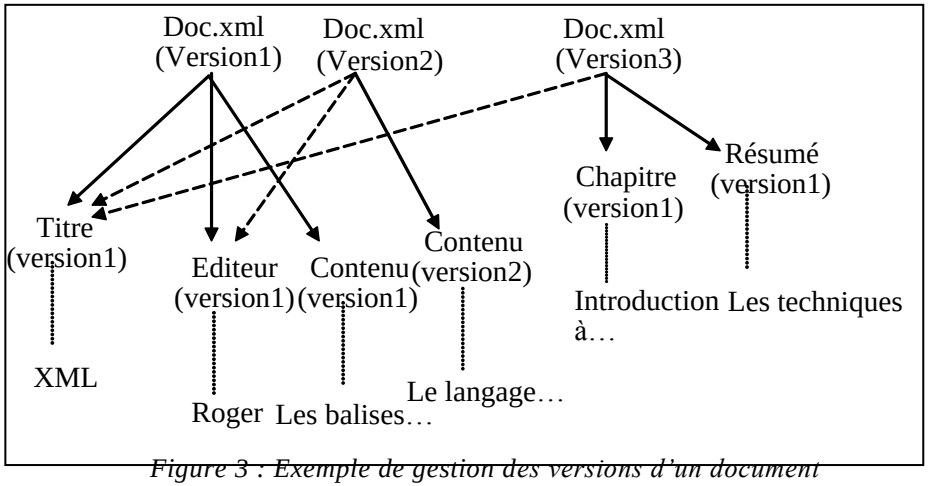


Figure 3 : Exemple de gestion des versions d'un document

4.2 Modèle générique étendu

Deux versions d'un document peuvent avoir des parties communes et des parties différentes. Les parties communes correspondent aux éléments spécifiques qui ne sont pas modifiées et les autres parties correspondent aux nouveaux éléments ou aux éléments spécifiques modifiées d'où la nécessité de la classe « Version_ES » (Versions des Eléments Spécifiques).

Deux versions d'une structure générique peuvent avoir des parties communes et des parties différentes c'est-à-dire des éléments génériques communs et des autres différents d'où l'importance de la classe « Version_EG » (Versions des Eléments Génériques). La figure 4 présente l'extension proposée au modèle générique afin de modéliser la gestion de versions. Les éléments ajoutés sont mis en évidence par la couleur grise.

Les classes, que nous avons ajoutées dans le nouveau diagramme, sont les suivantes :

- la classe « Version_SG » pour les versions de structures génériques : elle permet de garder, dans l'entrepôt, toutes les versions d'une structure générique. Une version de structure générique est générée si au moins un de ses éléments génériques a été modifié.
- la classe « Version_EG » pour les versions des éléments génériques : elle permet de marquer les différentes versions d'un élément générique. Une version d'élément générique peut être composée d'autres versions d'éléments génériques. A une version d'élément générique est rattaché obligatoirement l'élément générique correspondant (Lien d'association Rattacher). Un élément générique change de version lorsque ses descendants ou ses attributs génériques changent.
- la classe «Version_Doc » pour les versions de documents : elle permet de garder la trace de toutes les versions d'un document. Ainsi, pour un document donné, nous pouvons associer une ou plusieurs versions.

- la classe « Version_ES » pour les versions des éléments spécifiques : elle assure le suivi de toutes les versions d'un élément spécifique. Une version d'un élément spécifique peut appartenir à un ou plusieurs versions d'un document. Un élément spécifique change de version si son contenu, ses attributs spécifiques ou ses descendants changent.

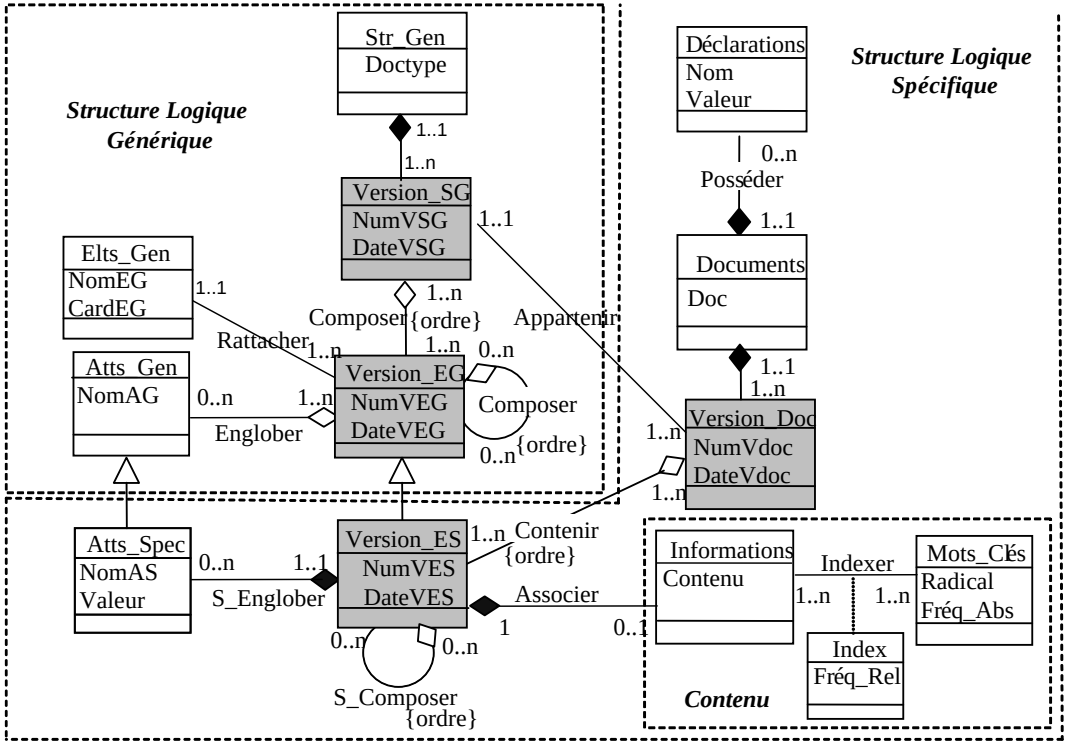


Figure 4 : Modèle générique étendu d'entrepôts de documents

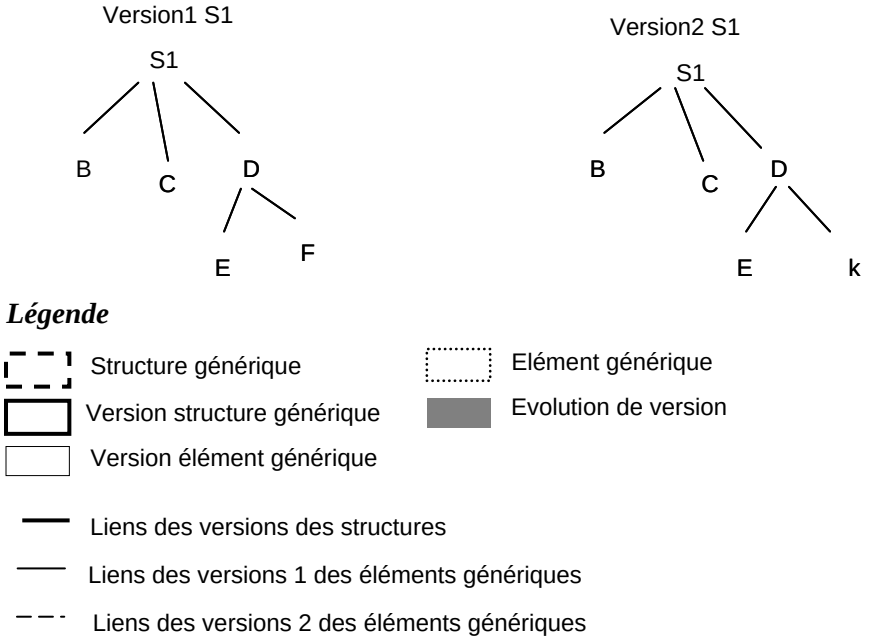
Remarques :

- Les différentes classes que nous avons ajoutées comportent deux attributs : un numéro séquentiel de version et sa date de création dans l'entrepôt.
- Dans notre approche, nous utilisons les liens de partage des versions c'est-à-dire que les versions d'éléments génériques ou spécifiques non modifiées seront partagées entre l'ancienne et la nouvelle version de la structure générique ou du document.
- Les liens d'héritage entre « VersionEG » et « VersionES » et entre « Atts_Gen » et « Atts_spec » sont utilisés afin de garantir une conformité des structures logiques spécifiques par rapport aux structures logiques génériques.

Dans ce qui suit, nous présentons deux exemples d'instanciation du modèle générique étendu. Le premier traite le cas de changement structurel et le deuxième traite celui du changement de contenu.

Premier Exemple : « Changement structurel »

Soit les deux versions suivantes de la structure générique S1 : La première version se compose des éléments génériques B, C et D. Ce dernier élément se compose des éléments E et F. La nouvelle version de S1 est similaire à la première version, la seule modification consistait à remplacer l'élément générique F par l'élément générique K.



L'instanciation de ces deux versions au niveau de l'entrepôt donne le diagramme d'objets simplifié suivant.

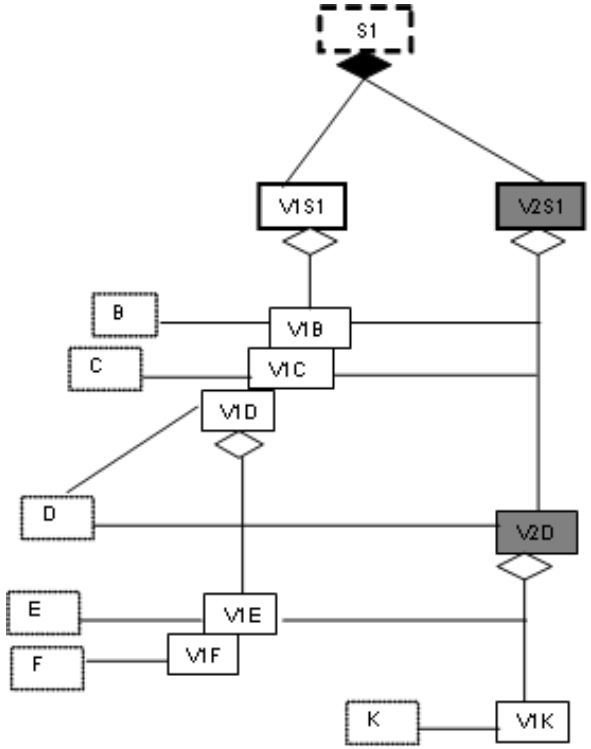


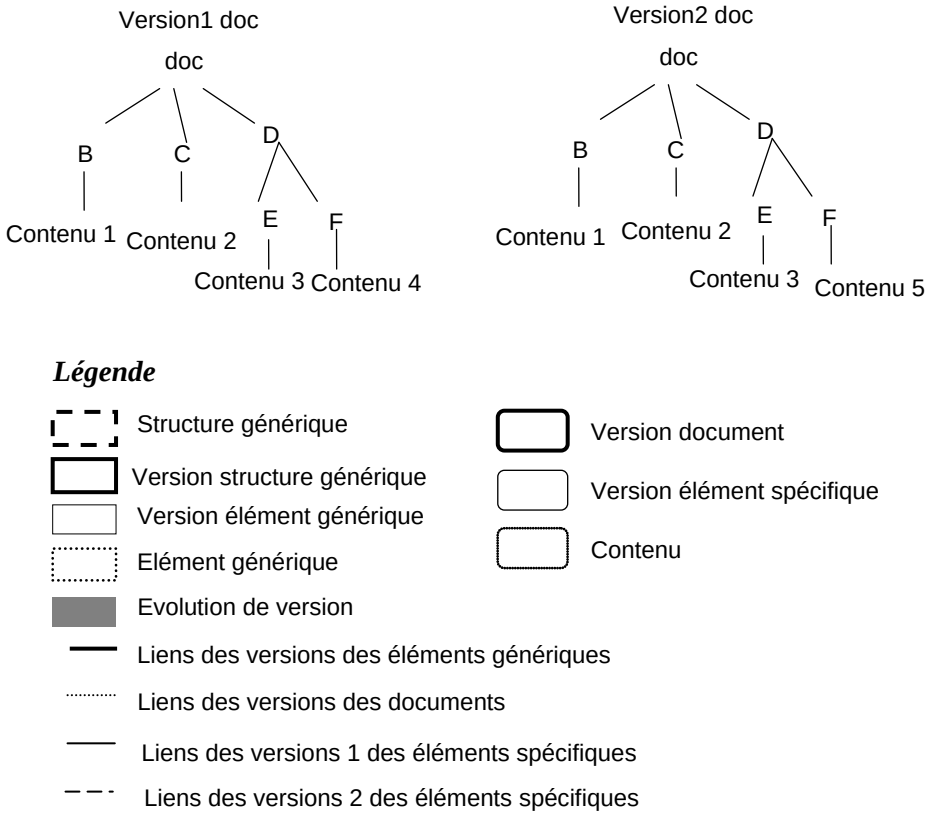
Figure 5 : Changement structurel - Exemple d'un diagramme d'objets

La structure S1 se compose de deux versions. La première se compose d'une première version de B, une première version de C et une première version de D. Cette dernière se compose d'une première version de E et d'une première version de F. Cependant, la deuxième version de S1 se compose de la première version de B et de C et d'une deuxième version de l'élément générique D (modification de ses descendants). Cette nouvelle version de D se compose de la première version de E et d'une première version de K.

Pour conclure, les deux versions de la structure générique S1 partagent les trois versions V1B, V1C et V1E des éléments génériques respectifs B, C et D. Alors que, la version de l'élément générique D change dans la deuxième version car ses descendants changent.

Deuxième Exemple : « Changement de contenu »

Soient les deux versions de document « Doc.xml ». Ces deux versions ont la même version de structure générique « Version1 S1 », seulement le contenu de l'élément spécifique F change au niveau de la deuxième version de « Doc.xml ».



L'instanciation de ces deux versions au niveau de l'entrepôt donne le diagramme d'objets simplifié suivant.

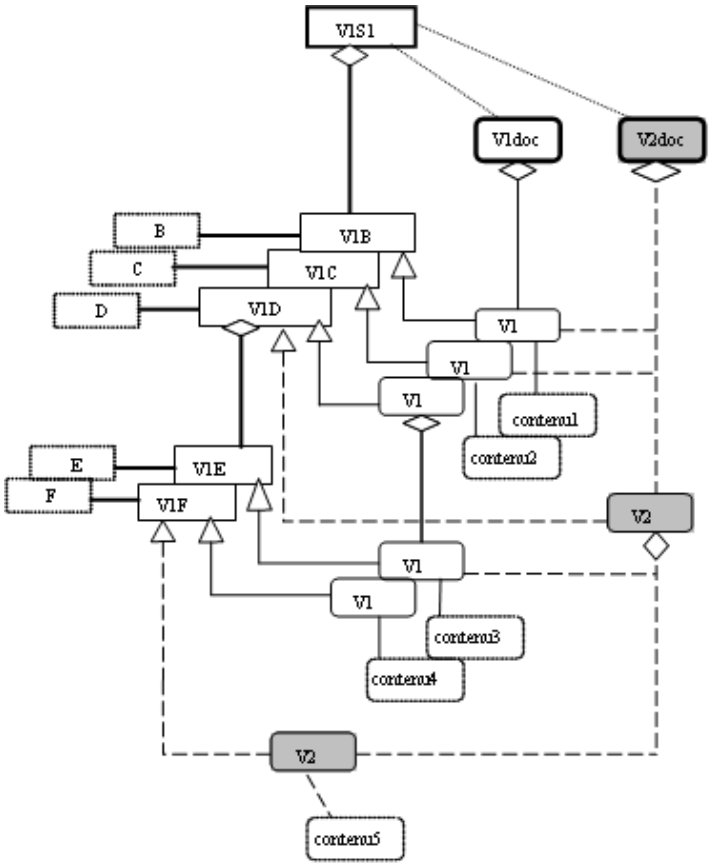


Figure 6 : Changement de contenu - Exemple d'un diagramme d'objets

Tous les éléments spécifiques de la première version de « Doc.xml » seront rattachés à la première version des éléments génériques correspondants. Cependant, la deuxième version de « Doc.xml » se compose de la première version des éléments spécifiques qui n'ont pas changé de contenu, à savoir : B, C et E. Le contenu de l'élément spécifique de F a changé, ce qui va entraîner une nouvelle version de F et une nouvelle version de son élément père D.

Pour conclure, les deux versions du document « doc.xml » partagent les versions des éléments spécifiques dont le contenu ne change pas. Le changement de la version d'un élément spécifique entraîne le changement de tous ses antécédents.

5 Implantation

Nous avons apporté les améliorations et les modifications, proposées dans le cadre de ce papier, sur l'outil DOCWARE (DOCument WAREhouse) que nous avons réalisé. Cet outil permet ainsi de gérer les différentes versions des documents [11]. La fenêtre principale de notre outil comporte trois menus : menu « Affichage », menu « Alimentation » et menu « Exploitation ». Le menu « Affichage » permet d'afficher soit un document avec ses différentes versions, soit l'arborescence d'une structure générique avec ses différentes versions.

Affichage d'un document : La figure 7 présente la version 3 du document « document2.xml ».

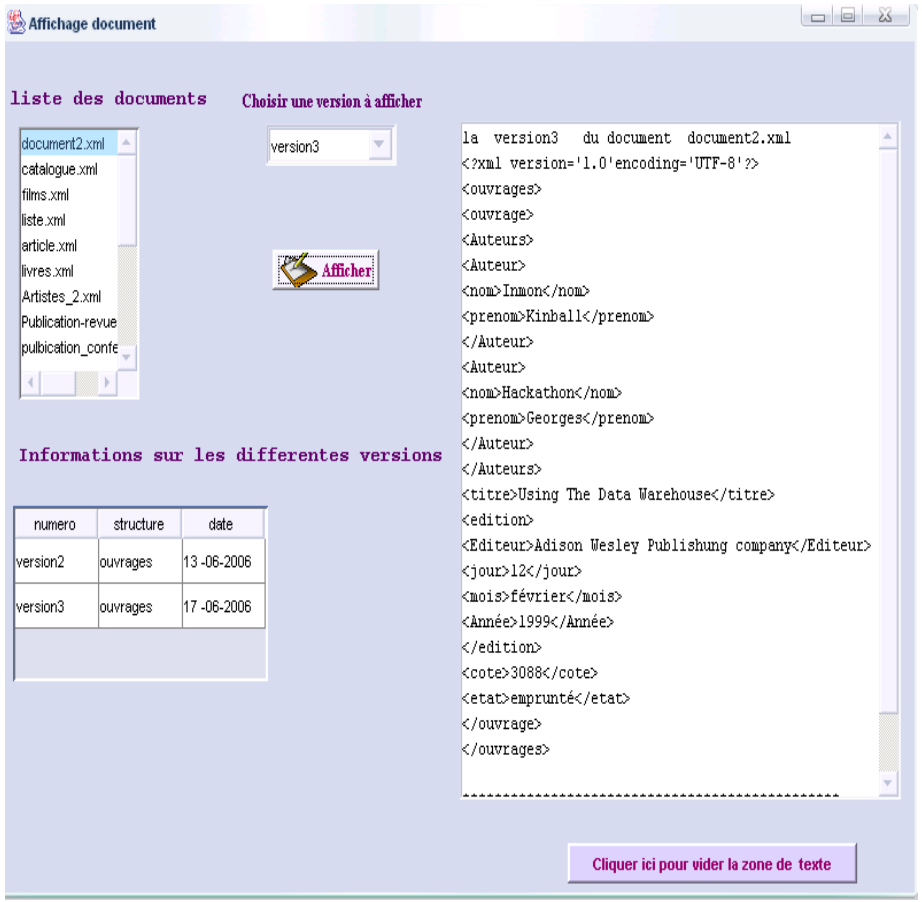


Figure 7 : Affichage d'un document et de ses versions

Affichage d'une structure générique : La figure 8 présente la version 2 de la structure générique « ouvrages ».

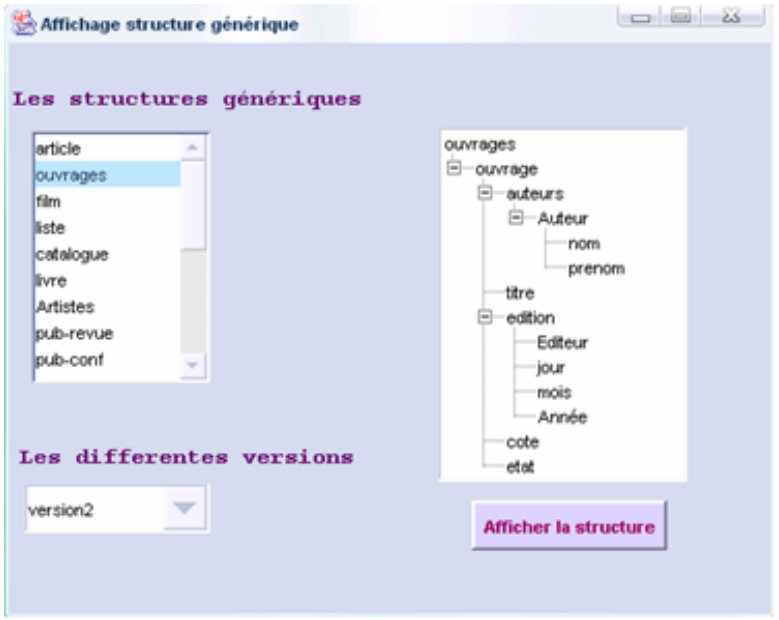


Figure 8 : Affichage d'une structure générique et de ses versions

Le menu « Alimentation » permet :

- L'ajout d'un document dans l'entrepôt tout en tenant compte de l'aspect version.
- La suppression d'une version d'un document.
- La suppression d'un document de l'entrepôt, ce qui entraîne la suppression de toutes ses versions.
- La suppression d'une version de structure générique entraînant ainsi la suppression de toutes les versions de documents appartenant à cette version de structure générique.
- La suppression d'une structure générique impliquant la suppression de toutes les versions de cette structure générique ainsi que leurs différentes versions de documents.

Le menu « Exploitation » permet l'analyse multidimensionnelle des informations documentaires de l'entrepôt. Les travaux sont en cours pour tenir compte de l'aspect version dans le processus d'analyse. En effet, l'utilisateur peut effectuer des analyses soit sur des différentes versions d'un ou de plusieurs documents, soit sur les versions actualisées des documents (les dernières).

6 Conclusion

Le concept d'entrepôts de documents doit permettre la gestion et l'exploitation aisée d'une mémoire documentaire constituée à partir d'informations provenant de sources distribuées et hétérogènes, filtrées, synthétisées, en vue d'être partagées et analysées. L'historisation des données et plus précisément la gestion des versions de documents des entrepôts constitue une étape indispensable pour la phase d'exploitation et celle d'analyse.

Dans ce papier, nous avons proposé une extension de notre modèle générique afin de gérer les changements structurels (lorsque la structure change d'une version à une autre) et les changements de contenu (lorsque le contenu change d'une version à une autre) que peuvent subir des documents ou une classe de documents. Dans notre approche, nous avons utilisé les liens de partage ; les versions d'éléments génériques ou spécifiques non modifiées seront partagées entre les différentes versions de structures génériques ou des documents.

Les perspectives que nous envisageons pour compléter cette étude concernent la visualisation en 3D des modifications entre les versions d'un document ou d'une classe de documents, ainsi que les techniques d'extraction de connaissances par rapport aux changements que subissent les données.

Remerciements

Nous tenons à exprimer nos vifs remerciements aux mademoiselles Amel LASSOUAD et Awatef RACHDI pour leur aide dans la réalisation de ce travail et plus précisément, l'intégration de la gestion de versions dans l'outil DOCWARE.

7 Bibliographie

[1] ABITEBOUL S., CLUET S., FERRAN G., ROUSSET M.C., *The Xyleme Project*, Computer Networks, 39(3): 225-238, 2002.

[2] BALMISSE G., *Gestion des connaissances : Outils et applications du knowledge management*, Edition Vuibert, 2002.

[3] COBENA G., *Gestion des changements pour les données semi-structurées du Web*, Thèse de doctorat, Juin 2003.

[4] CURBERA D., EPSTEIN A., *Fast Difference and Update of XML Documents*, XTech, San Jose, California, USA, March 1999.

[5] HASSEN S., *Entrepôt de documents : Aspect temporel*, Mémoire de DEA, Université Paul Sabatier, Toulouse, 2004.

[6] KHROUF K., RAVAT F., SOULE-DUPUY C., *Comparaison et fusion de structures logiques de documents semi-structurés*, Ingénierie des Systèmes d'Information (ISI), Edition Hermès, Vol. 8, N°5-6/2003, p. 127-151, 2003.

[7] KHROUF K., SOULÉ-DUPUY C., *A Textual Warehouse Approach: a Web Data Repository*, Chapter VII, Intelligent Agents for Data Mining and Information Retrieval, Idea Group Publishing, p. 101-124, 2004.

[8] KHROUF K., SOULE-DUPUY C., *DOCWARE : Vers l'entreposage et l'analyse multidimensionnelle de documents*, Conférence en Recherche d'Information et Applications (CORIA), p. 405-420, Grenoble, France, Mars 2005.

[9] KHROUF K., MBARKI M, RAVAT F., SOULE-DUPUY C., VALLES-PARLANGEAU N., *Entreposage de documents multimédia : modélisation basée sur le contenu et instanciation automatique*, Centre de Hautes Etudes Internationales d'Informatique Documentaire (C.I.D.), p. 1-24. 2006.

[10] NICOLLE C., AMGHAR Y., PINON J.M., *Gestion simultanée des liens des versions dans le contexte des documents structurés*, Colloque International sur le Document Electronique (CIDE), 2000.

[11] RACHDI A., LASSOUAD A., *Les entrepôts de documents : Gestion des versions*, Mémoire de Projet de Fin d'Etudes de Maîtrise, ISIMS, Université de Sfax, 2006.

[12] RUSU L.I., RAHAYU J.W., TANIAR D., *Mining Changes from Versions of Dynamic XML Documents*, Workshop on Knowledge Discovery in XML Documents (KDXD), p. 3-12, 2006.

[13] SOULE-DUPUY C., *Bases d'informations textuelles : Des modèles aux applications*, Mémoire d'HDR, Université Paul Sabatier, Toulouse III, Décembre 2001.

[14] W3C, *Document Object Model (DOM) Level 3 Core Specification Version 1.0*, W3C Recommendation, 7 April 2004.

[15] WANG Y., DEWITT D.J., CAI J.Y., *X-Diff: An Effective Change Detection Algorithm for XML Documents*, International Conference on Data Engineering (ICDE), p. 519-530, Bangalore, India, March 2003.