Le langage SGML et ses applications en terminologie

Jean Fontaine (Université de Montréal)

L'avènement des micro-ordinateurs a permis la démocratisation de l'informatique et la multiplication de systèmes de bases de données pour grand public. Les terminologues ont su tirer profit de ces outils informatiques. Les données terminologiques se présentant normalement comme une série de fiches ou d'articles plutôt que comme du texte continu, un système de base de données (SDBD) paraît l'outil naturel pour remplacer le fichier manuel du terminologue. Mais on s'aperçoit rapidement que les fiches terminologiques, par leur nature, ne se conforment pas toujours aisément au moule des SDBD classiques. En effet, la structure d'une fiche terminologique n'est pas simple : elle peut prendre une forme hiérarchisée, certains de ses champs ou de ses éléments peuvent être facultatifs et/ou répétables, l'ordre des champs n'est pas toujours le même, la longueur des données peut varier énormément d'une fiche à l'autre (les contextes, par exemple), plusieurs langues (voire plusieurs alphabets) peuvent être nécessaires, etc.

Tôt ou tard se fait également sentir le besoin d'échanger des données entre systèmes différents. Survient alors le problème de l'incompatibilité des systèmes ou des formats, problème qui nécessite l'écriture de programmes de conversion ad hoc.

Ce problème n'est pas propre à la terminologie. En vue de faciliter les échanges, des efforts ont été entrepris pour uniformiser le codage informatique des données. Citons la norme ISO 646 de l'Organisation internationale de normalisation, adoptée en 1983, portant sur la codification des caractères.

Un nouveau besoin fut d'uniformiser, outre le codage du contenu, le codage de la structure du document à échanger. C'est un des buts visés par la norme ISO 8879, adoptée en 1986, qui définit le langage SGML, ou Standard Generalized Markup Language, un langage puissant pour décrire la structure d'un document à l'ordinateur.

Cette norme, qui fut dès sa publication un best-seller de l'ISO, a suscité un intérêt certain dans le monde de la terminologie comme ailleurs. Des initiatives ont été prises pour définir des formats d'échange qui soient conformes à la norme SGML tout en étant adaptés aux besoins propres des terminologues.

Nous présenterons dans un premier temps le langage SGML, puis nous passerons en revue quelques formats utilisés ou proposés pour l'échange de données terminologiques : MATER et MicroMATER (MM), Nordic Terminological Record Format (NTRF) et TEI-TERM.

1. Standard Generalized Markup Language (SGML)

Le SGML est un langage qui a pour but de formaliser la façon de décrire la structure d'un texte, d'un document, et ce, d'une façon qui soit indépendante de sa présentation visuelle et des systèmes informatiques utilisés pour son traitement. Ce n'est pas la façon de structurer ou de coder le texte qui est normalisée, mais plutôt le langage utilisé pour décrire la façon dont on a codé la structure du document.

On pourrait parler dans ce sens d'un métalangage plutôt que d'un langage. La norme parle d'ailleurs de «syntaxe abstraite» pour désigner l'ensemble de règles générales à suivre pour créer une «syntaxe concrète» que chaque utilisateur ou groupe d'utilisateurs peut définir selon ses besoins spécifiques. La norme comprend aussi une «syntaxe concrète de référence» qui, sans être obligatoire (sauf pour les déclarations générales qui chapeautent un document), peut servir de modèle.

Avec le SGML, il s'agit essentiellement d'ajouter aux données du document des marqueurs, des codes qui décrivent la structure de ces données, après avoir précisé au préalable la forme et le sens que l'on donne à ces marqueurs et les relations structurales qu'ils entretiennent entre eux.

Le document est l'unité de base pour le SGML. Les types de document sont pratiquement illimités : un roman, un manuel technique, un simple mémo, un article de journal, un fichier terminologique, etc. Un document est une structure logique. En ouvrant n'importe quel livre, on voit qu'il est fait d'une suite de caractères présentant une structure hiérarchisée : il comporte des pages préliminaires et des annexes; le corps principal du livre est subdivisé en chapitres, en sous-chapitres, en paragraphes, comprenant des citations, des tableaux, des listes, du texte continu, etc. Le SGML appelle «éléments» et «sous-éléments» les constituants de la structure logique du document. Le document constitue l'élément générique, alors que les simples caractères constituent les éléments terminaux. Chaque type de document se distingue par son contenu, mais ausssi par sa structure et ses types d'éléments. Dans le domaine de la terminologie, la structure d'un article sera différente selon qu'il s'agit de l'article d'un vocabulaire monolingue ou de celui d'un lexique bilingue.

Dans un vocabulaire, des types d'éléments pourraient être l'article, la vedette, la note grammaticale, la définition, etc.

Il est possible d'affecter aux types d'éléments des attributs, c'est-à-dire des catégories prenant un nombre limité de valeurs, ce qui permet de qualifier les éléments avec plus de précision. Par exemple, l'élément «terme» pourrait être affecté d'un attribut «statut», qui pourrait prendre les trois valeurs suivantes : «normalisé», «recommandé», «toléré».

Le SGML permet donc de définir une seule fois la structure d'un certain type de document en définissant les types d'éléments qu'il doit contenir et les relations qu'ils doivent avoir entre eux. La définition d'un type de document comprend : les noms donnés aux types d'éléments autorisés dans un document de ce type, leurs attributs et valeurs d'attribut possibles, ainsi que la structure de ces éléments, c'est-à-dire les sous-éléments qui peuvent ou doivent constituer un élément, et ce, dans quel ordre.

Une fois ainsi défini un type de document, chaque nouveau document créé ou échangé peut être déclaré comme appartenant à ce type de document et être balisé à l'aide de marqueurs correspondant aux types d'éléments définis. Cette «déclaration du type de document» se place en tête du document. Le programme chargé de reconstituer la structure du document reçu vérifiera d'abord dans cette déclaration à quel type le document appartient, puis lira, conformément à ce type de document, les données et les marqueurs qui lui permettront de reconstituer la structure du document.

Les éléments constituent donc la structure logique du document. Mais physiquement, les données peuvent être structurées en entités, c'est-à-dire en suites de caractères que l'on désire, pour des raisons pratiques, considérer comme une unité, indépendamment de leur structure logique. Une entité peut être faite d'un seul caractère ou d'un document entier. L'utilité des entités est la possibilité de remplacer par un code les chaînes de caractères qui reviennent souvent, de diviser un document en parties, de copier des données d'une partie du document à l'autre, de les extraire, de les transférer d'un document à l'autre, d'y référer à l'aide de renvois, etc.

Un document conforme au SGML sera donc représenté comme une suite de caractères organisés physiquement en une structure d'entités et logiquement en une structure d'éléments. Il consiste en caractères de données, qui représentent le contenu du document, et de caractères marqueurs, qui représentent la structure des données et donnent d'autres informations utiles pour leur traitement.

Les marqueurs sont de quatre types principaux : les déclarations de marquage, les marqueurs descriptifs, les références aux entités, les déclarations et les instructions de traitement. Ces types de marqueurs sont obligatoires bien que les caractères utilisés pour les représenter (la syntaxe concrète) soient laissés au choix de l'utilisateur. Les notations données ici sont celles de la syntaxe concrète de référence incluse dans la norme.

1. déclarations de marquage

Ces déclarations, normalement placées en tête de document, servent à définir les types de document, les entités, les éléments, les attributs et valeurs d'attribut, la forme, la signification et la structure des marqueurs utilisés. Elles sont placées entre les symboles <! et >. Voici par exemple deux déclarations d'éléments :

<!ELEMENT entrée (terme, déf, contexte, syn) >
<!ELEMENT contexte (texte, source, page) >

Ces déclarations pourraient par exemple être faites lorsqu'on définit la structure du type de document «vocabulaire». La première signifie : «L'élément *entrée* sera ainsi structuré : il comprendra dans l'ordre les quatre sous-éléments *terme*, *définition*, *contexte* et *synonyme*». À son tour, le sous-élément *contexte* est déclaré comme devant être constitué des éléments *texte*, *source* et *page*.

La déclaration peut comprendre des symboles indicateurs d'occurrence des éléments :

? optionnel (l'élément peut apparaître 0 ou 1 fois)
\+ obligatoire et répétable (1 fois ou plus)
* optionnel et répétable (0 ou 1 fois ou plus)
aucun symbole : obligatoire (1 et 1 seule fois)

Par exemple, la déclaration qui suit spécifierait qu'une entrée doit contenir un et un seul terme avec au moins un contexte, et facultativement des synonymes et un renvoi :

<!ELEMENT entrée (terme, contexte+, syn*, renvoi?) >

Ces indicateurs peuvent affecter tout le groupe contenu entre parenthèses. Ainsi

<!ELEMENT lexique (terman, terfra)+ >

signifierait qu'un lexique est une liste formée d'au moins une paire bilingue (anglais et français) de termes.

Les virgules qui séparent les sous-éléments dans la déclaration sont des connecteurs qui précisent l'ordre de ces éléments. Il existe trois types de connecteurs qui signifient respectivement :

(a , b , c) a et b et c dans cet ordre
(a & b & c) a et b et c dans un ordre quelconque
(a | b | c) a ou b ou c (un seul des éléments peut apparaître).

Voici par exemple comment nous pourrions définir deux listes d'attributs pour l'élément «terme» :

<!ATTLIST terme statut (normalisé | recommandé | toléré)?
forme (acronyme | sigle | abréviation)?>

Pour chacun des deux attributs (statut et forme) de l'élément «terme» une valeur serait ainsi facultative et limitée à une seule des trois permises.

L'usage combiné des parenthèses, des connecteurs et des indicateurs d'occurrences permet donc de définir aisément des structures logiques complexes.

2. marqueurs descriptifs

Les marqueurs descriptifs, ou étiquettes, sont insérés dans le corps du document et servent à délimiter les données qui constituent un élément, à les encadrer à la manière de parenthèses. Normalement ces marqueurs se présentent donc par paires d'étiquettes, l'étiquette de début et l'étiquette de fin.

Soit «a» le nom d'un élément et «.....» son contenu. L'élément sera ainsi marqué : <a>...... Si l'élément possède un atribut «x» de valeur «n», on aura : .....<a>. Voici un exemple de fiche ainsi marquée :

<entrée> <terme statut=recommandé> capteur </terme> <gramm> n.m. </gramm> <domaine> informatique </domaine> <déf> Dispositif qui permet la conversion d'un phénomène donné en une grandeur mesurable. </déf> <syn statut=toléré> senseur <note> calque de l'anglais sensor </note> </syn> </entrée>.

Chaque étiquette de début est donc formée du nom du type d'élément entre les symboles < et >. Chaque étiquette de fin est

formée du nom du type d'élément précédé d'une barre oblique et placé entre les mêmes symboles < et >. De plus, les éléments <terme> et <syn> sont ici affectés d'une valeur pour l'attribut «statut».

On voit que les éléments peuvent s'emboîter les uns dans les autres (par exemple la note se rapporte seulement au synonyme et non au terme principal), et ce, sans limite de niveaux de profondeur.

Il existe trois possibilités de disposition des étiquettes :

1. groupement d'éléments : <a>..... .....
2. emboîtement d'éléments : <a>...............
3. chevauchement d'éléments : <a>...............

On s'aperçoit que, dans le premier cas, si les éléments se suivent linéairement, l'étiquette de fin est redondante, car un élément se termine nécessairement là où débute l'élément suivant. L'étiquette de fin est donc facultative dans les cas où les champs sont uniquement groupés. On pourrait donc écrire, pour simplifier :

<a>..........<c>.....

Dans le cas de l'emboîtement (2), l'étiquette de fin est nécessaire, mais toutes les étiquettes de fin de tous les éléments pourraient être représentées d'une même façon sans distinction car il est implicite que l'étiquette de fin ferme le dernier élément à avoir été ouvert. On pourrait donc simplifier en écrivant :

<a>..........</>.....</>

C'est uniquement dans le cas du chevauchement (3) que les étiquettes de fin d'élément doivent être bien distinctes.

3. références aux entités

Ces marqueurs indiquent l'endroit dans le texte où des entités (définies au long dans la déclaration ou dans un document externe) doivent être restituées. Ils prennent la forme d'une chaîne de caractères précédée du symbole «&» et suivie du point-virgule. Nous pouvons par exemple déclarer au début du texte que la référence «&AFNOR;» représente l'entité que constitue la chaîne de caractères «Association française de normalisation». Chaque fois que, dans le corps du document, sera rencontrée cette référence «&AFNOR;», la chaîne «Association française de normalisation» sera restituée à cet endroit précis.

4. instructions de traitement

Ces instructions, destinées au système de traitement, sont particulières à chaque système et, le plus souvent, à chaque application. Ce

sont des instructions de ce type qui affectent la présentation visuelle du texte (affichage, impression).

Ces procédures peuvent être appariées avec les types d'éléments : par exemple on peut spécifier, au début, que tout ce qui est contenu à l'intérieur des éléments de type «terme» soit imprimé en caractères gras.

Des techniques spéciales sont prévues pour réduire au strict minimum le marquage du texte (par exemple l'omission de marqueurs redondants ou implicites pour l'ordinateur qui lit les données).

Le SGML permet également de définir le jeu de caractères utilisés dans le document. Le jeu de base est celui de la norme ISO 646 (c'est-à-dire le code ASCII à 7 bits), mais il est possible de représenter d'autres jeux de caractères.

Toutes ces possibilités font du SGML un langage puissant et souple pour représenter des structures de données. Les documents dont la structure logique est ainsi codée deviennent facilement convertibles en bases de données puisque les éléments logiques peuvent être distinctement traités. Le SGML simplifie le traitement de documents qui doivent passer par de nombreuses transformations. Il peut également servir de base pour la définition de formats universels d'échange de données.

Nous avons dit en introduction que l'absence de format d'échange entre SDBD entraîne la nécessité d'écrire un programme de conversion (un dans chaque sens) pour chaque paire de SDBD afin de rendre les formats compatibles. Dans le domaine particulier de la terminologie, des efforts ont été faits depuis quelques années pour remédier à ce problème.

2. MATER

Le Comité technique 37 de l'ISO, chargé des questions de principes et de coordination en matière de terminologie, comprend un sous-comité s'occupant des outils informatiques de la terminologie. C'est ce sous-comité qui a mis au point ce qui, en 1987, devint la norme ISO 6156, intitulée *Magnetic tape exchange format for terminological/lexicographical records (MATER)*.

Cette norme se veut un format universel d'échange sur bande magnétique. Le recours à un format d'échange intermédiaire permet pour chaque SDBD de réduire à deux le nombre de programmes de conversion : un programme SDBD --> MATER et un programme MATER --> SDBD. Celui qui envoie des données n'a plus à connaître dans le détail le système ou le format utilisé par chaque destinataire, et vice versa. Les données transmises au moyen de ce format sont précédées d'une zone de caractères

de longueur fixe qui décrit la façon dont elles sont structurées. Un fichier MATER comprend un label d'enregistrement qui contient l'information concernant tout le fichier, suivi d'un nombre d'enregistrements de longueur variable, puis d'un marqueur de fin de fichier. Chaque enregistrement est constitué de champs, lesquels comprennent un nom de champ et des données.

Comme le laisse deviner le support privilégié, ce format est conçu en fonction des gros systèmes plutôt que des micro-ordinateurs. Une norme d'échange de données sur bande magnétique ne rencontre pas les besoins des SDBD modernes, en particulier dans un environnement de micro-ordinateurs, lesquels ont permis le développement de fichiers terminologiques individualisés.

Les spécifications du format MATER, bien que facilement lisibles pour l'ordinateur, deviennent lourdes et fastidieuses pour l'utilisateur d'un micro-ordinateur. Elles comprennent de nombreux pointeurs binaires et indicateurs de longueur en caractères et rendent difficile l'édition d'un fichier MATER à partir d'un micro-ordinateur (à l'aide d'un traitement de texte, par exemple).

C'est pourquoi se sont développés de nouveaux formats d'échange de données terminologiques mieux adaptés à la micro-informatique (signalons que l'ISO compte adopter une nouvelle version du MATER qui tienne compte des récents développements dans le domaine).

3. MicroMATER

Le MicroMATER, comme son nom le laisse entendre, conserve l'esprit du format MATER tout en l'adaptant à la micro-informatique. Il est né en 1987 à l'état de prototype développé par le Brigham Young University Translation Group de Provo, Utah, en association avec la American Translators Association et le Kent State University Institute for Applied Linguistics, Ohio, en consultation avec Infoterm (International Information Centre for Terminology).

Depuis, il a servi pour l'échange de données entre bases de données terminologiques, mais aussi entre celles-ci et d'autres types d'applications comme les systèmes de traitement de texte qui permettent l'édition de tels fichiers.

La deuxième version du MicroMATER (présentée dans Melby), sans être une version définitive, se veut encore plus accessible pour un large public tout en conservant assez de formalisme pour être facilement traitable par les diverses applications informatiques.

Sans être en tous points conforme au langage SGML, le format MicroMATER y est facilement convertible, puisqu'il en respecte plusieurs conventions.

Sans entrer dans le détail de l'explication des divers codes, voyons de quoi aurait l'air une fiche (l'exemple est tiré de Melby) :

*R01357
{EN :0LTU} fiberglass
{FR :1LTU} laine de verre
{FR :1SRC} Ged, page 493
{FR :2LTU} fibre de verre
{FR :2SRC} PR, page 701
{RL :CLS} 666

On peut considérer un fichier MicroMATER comme un document SGML dont les noms de champs sont entre accolades plutôt qu'entre < et >. D'autres caractères spéciaux sont aussi directement convertibles en étiquettes SGML. Les champs ont une longueur variable et leur étiquette de fin est omise car implicite (elle précède obligatoirement l'étiquette de début du champ suivant). Cela suppose que les champs se suivent linéairement et semble empêcher la représentation de structures hiérarchisées au moyen de paires d'étiquettes emboîtées les unes dans les autres. Mais cette structure arborescente peut se déduire des noms donnés aux champs. En effet, le MicroMATER permet, pour chaque enregistrement du fichier, une structure à cinq niveaux :
1. niveau de l'enregistrement entier (comprend l'identificateur et d'autres champs possibles, par exemple, la date de rédaction);
2. section(s) pour chaque langue;
3. dans chaque section : unité(s) terminologique(s) et champs associés (répétables);
4. langue des données entrées dans chaque champ (elle peut différer de la langue de la section);
5. données entrées dans chaque champ.

Normalement, un enregistrement correspond à une notion. Dans chaque langue traitée pour cette notion, on peut compter plus d'un terme (des synonymes par exemple). Plusieurs champs pourront être associés à chacun de ces termes.

C'est le nom que le MicroMATER donne à ces champs qui permet à l'ordinateur de savoir où il se trouve exactement dans la structure et à quel

niveau. Un nom de champ est normalement la concaténation de cinq codes, mais des valeurs par défaut permettent souvent d'alléger au strict minimum ces noms. Dans notre exemple, pour l'enregistrement identifié par R01357, l'ordinateur pourra déduire du nom du champ {FR :2SRC} que ce champ signifie «section française de l'enregistrement, second terme, (premier) champ source». On pourrait dire par analogie que c'est un langage basé sur la morphologie plutôt que sur la syntaxe. L'ordre des champs à l'intérieur d'un enregistrement demeure libre, tout comme l'ordre des mots est plus libre dans une langue agglutinante que dans une langue isolante.

Le MicroMATER permet d'associer à chaque terme des champs primaires, champs qui peuvent être qualifiés par une liste de catégories secondaires. Nous retrouvons ici la distinction que fait le SGML entre éléments et attributs. Ainsi le champ DTY (Description Type) peut prendre les attributs DEF, EXP ou CTX (respectivement DEFinition, EXPlication et ConTeXte).

Le MicroMATER est donc une application puissante du MATER qui permet de la flexibilité dans la représentation des formats (en particulier dans l'ordre des champs) tout en réduisant au minimum le codage nécessaire pour convertir un fichier texte dans ce format.

4. Nordic Terminological Record Format (NTRF)

C'est un langage de marquage conforme au SGML et utilisé pour l'échange de fichiers entre les bases de données terminologiques de trois pays scandinaves (Finlande, Norvège et Suède). Il vise à harmoniser et à fusionner les données terminologiques de ces pays en vue de créer un dictionnaire scandinave unique.

Le format NTRF permet le groupement et l'emboîtement d'éléments (sans limite de niveau), mais interdit leur chevauchement, ce qui permet d'avoir une étiquette unique de fin d'élément pour tous les éléments.

À l'intérieur de chaque enregistrement d'un fichier, on distingue les champs du niveau de l'enregistrement et les champs emboîtés. Reprenons la même fiche exemple et voyons l'aspect qu'elle prendrait :

R01357
enTE1 fiberglass
frTE1 laine de verre
 <SOURF Ged, page 493>
frTE2 fibre de verre
 <SOURF PR, p.701>

CLAS 666
=

Bien que conforme au SGML, le NTRF utilise une syntaxe concrète différente de celle qui est fournie dans la norme. Elle exploite les possibilités des techniques de réduction de marquage en se servant des caractères «retour de chariot» et «saut de ligne». Les champs du niveau de l'enregistrement sont ceux qui commencent au début d'une ligne. Seuls les champs emboîtés (ici, la source qui suit chaque terme français) sont délimités par des étiquettes visibles. L'étiquette de début est alors formée du nom du champ précédé de < et suivi d'un espace. L'étiquette de fin de champ est simplement le symbole >.

Comme pour le MicroMATER, le nom du champ est composé de la concaténation de codes. Normalement le nom est formé de trois codes : un code de langue (selon la norme ISO 639, comme pour le MicroMATER), un code de type de champ (par exemple TE, SY, DEF, NOTE) et un numéro facultatif, en cas de répétition. Ainsi «frTE2» signifie «deuxième terme en langue française». Comme pour le MicroMATER, des valeurs par défaut ou implicites permettent d'alléger le nom des champs.

Une cinquantaine de codes de type de champ sont prévus ici aussi pour représenter les nombreuses catégories terminologiques. Mais ils sont tous sur le même niveau : contrairement au MicroMATER, on n'établit pas de distinction entre éléments et attributs.

Chacun peut modifier pour ses besoins propres le jeu de codes de types de champ de façon qu'ils soient mnémoniques dans chaque langue (ainsi les Norvégiens pourraient remplacer le code SOURF par le code KILF). Pour l'échange, ce sont cependant les codes basés sur l'anglais qui doivent être utilisés. Sur ce point, le MicroMATER a l'avantage de prévoir plusieurs jeux de codes pour plusieurs langues, jeux qui peuvent tous être utilisés dans le format d'échange (un code est prévu qui spécifie dans l'en-tête le jeu utilisé).

Le NTRF prévoit de plus toute une série de codes de caractères spéciaux, de fonctions et d'instructions de traitement (jeux de caractères, alphabets et translittération, présentation visuelle). Les caractères de base sont aussi les caractères ASCII à 7 bits.

Le format NTRF présente donc plusieurs points communs avec le MicroMATER. Il a l'avantage de permettre des structures hiérarchisées sans limite de niveau (qui exigent toutefois des marqueurs de fin de champ) tout en représentant également (à un degré moindre que le MicroMATER) la structure dans les noms des champs. Si on veut, ce langage est plus

«isolant» tout en gardant un côté «agglutinant». Son «vocabulaire» de noms de champs prévus est à peu près de la même étendue (une cinquantaine de catégories), mais moins structuré (pas de distinction entre éléments et attributs).

5. TEI-TERM

Une troisième voie poursuivie est la mise au point d'un format d'échange universel basé sur le SGML qui s'inscrive également dans le cadre de la Text Encoding Initiative (TEI).

La TEI, organisée en 1987 et appuyée par un grand nombre d'institutions et d'associations prestigieuses, se donne pour but d'établir, pour certains types de données et de documents, des lignes directrices SGML pour leur échange, leur traitement, leur création. On vise également à faciliter l'extraction de données à des fins de recherche ou de gestion de bases de données.

Un groupe de travail de la TEI s'occupe particulièrement de terminologie : le Terminological Data Work Group, Analysis and Interpretation 7 (AI7), chargé de créer une liste documentée de noms d'éléments, d'attributs et de valeurs d'attribut, ainsi qu'une description de leurs relations structurelles.

Le fait d'inscrire un format d'échange terminologique dans le cadre de la TEI augmente son potentiel d'interaction dynamique avec d'autres types de documents inclus dans la même définition de type de document TEI, tels des formats d'échange pour les bases de données lexicographiques, documentaires, bibliographiques, textuelles, etc. Les possibilités d'extraction d'information s'en trouveraient accrues.

Le format d'échange de données terminologiques proposé porte le nom de TEI-TERM. Dans le TEI-TERM, la catégorie qui correspond à un enregistrement ou une entrée est la catégorie <termEntry>. Elle peut contenir un ou plusieurs termes <term> avec leurs éléments associés. L'ensemble formé par un terme et ses éléments associés est appelé «Term Information Group» (<tig>). Au minimum, un <termEntry> contient au moins un <tig> qui contient au moins un <term> (qui peut être vide dans certains cas).

Selon les capacités des bases de données en présence, le TEI-TERM permet trois niveaux de structuration des données. Leur syntaxe diffère, mais ils possèdent un contenu sémantique identique.

Le premier niveau est celui d'une entrée complètement hiérarchisée. Il permet la structuration la plus explicite. C'est une hiérarchie algébrique

stricte, permettant l'emboîtement des éléments à l'intérieur des <tigs>. Reprenons notre fiche exemple, en tabulant pour clarifier la lecture :

```
<termEntry id=R01357>
        <tig>
                <term lang=eng> fiberglass
                </term>
        </tig>
        <tig>
                <term lang=fra> laine de verre
                        <citnRef> Ged, page 493
                        </citnRef>
                </term>
        </tig>
        <tig>
                <term lang=fra> fibre de verre
                        <citnRef> PR, page 701
                        </citnRef>
                </term>
        </tig>
        <classification> 666
        </classification>
</termEntry>
```

Nous voyons que la forme des étiquettes respecte la syntaxe concrète de référence du SGML. Nous voyons aussi que le numéro d'identification de la fiche (id) est ici considéré comme un attribut de <termEntry> alors qu'il est un élément indépendant dans le format NTRF. De même, «lang» est ici un attribut de <term>. La structure arborescente de la fiche est ainsi totalement explicitée, mais ce n'est pas toujours possible pour les bases de données terminologiques.

C'est pourquoi un deuxième type de structuration est permis. C'est l'entrée linéaire basée sur l'adjacence des éléments. Dans une entrée, un élément peut se rapporter à toute l'entrée, ou bien seulement à un terme, ou bien seulement à un autre élément du <tig>. Lorsque l'emboîtement n'est pas permis, on impose deux règles d'adjacence :

(1) tout élément apparaissant dans <termEntry> avant le premier <term> est considéré comme se rapportant à toute l'entrée;

(2) tout élément apparaissant dans <termEntry> après un <term> et avant le <term> suivant est considéré comme associé au <term> qui le précède.

Une conséquence de ces règles est de rendre redondantes (et donc facultatives) les étiquettes <tig>. Les éléments à l'intérieur de l'entrée se suivent linéairement sans emboîtement explicite :

<term Entry id=R01357>
<classification> 666
</classification>
<term lang=eng> fiberglass
</term>
<term lang=fra> laine de verre
</term>
<citnRef> Ged, page 493
</citnRef>
<term lang=fra> fibre de verre
</term>
<citnRef> PR, page 701
</citnRef>
</termEntry>

Nous voyons que l'élément <classification> (la cote dans un système de classification des notions) a dû être déplacé en tête d'entrée pour se conformer aux règles d'adjacence. L'avantage de ce type de structure est de simplifier le marquage, mais l'inconvénient est une contrainte sur l'ordre des champs.

Il peut arriver dans des bases de données que des éléments, plutôt que de se rapporter au terme précédent, se rapportent à un autre élément, ou bien se rapportent à un terme qui n'est pas le précédent. Imaginons, dans notre exemple, que les deux sources françaises doivent plutôt être placées en toute fin de l'entrée. Pour représenter cela dans une entrée linéaire (on parle alors d'entrée linéaire à éléments non adjacents) on a recours à des pointeurs, des attributs dont la valeur est un numéro. Ici, cet attribut est «group», qui indique à quel «tig» logique on se réfère :

<term Entry id=R01357>
<classification> 666
</classification>
<term lang=eng> fiberglass
</term>
<term lang=fra group=1> laine de verre
</term>
<term lang=fra group=2> fibre de verre
</term>

<citnRef group=1> Ged, page 493
</citnRef>
<citnRef group=2> PR, page 701
</citnRef>
</termEntry>

Si, toujours dans une entrée linéaire, on voulait indiquer qu'un élément est sous-élément d'un autre élément plutôt que d'un terme, on utiliserait de la même façon le pointeur depend=x. L'important est de ne pas mélanger les structures emboîtées et les structures linéaires.

Des pointeurs analogues permettent de faire des renvois entre entrées différentes, en se référant à leur numéro d'identification. On peut même préciser de quel type de renvoi il s'agit (équivalent, voisin notionnel, renvoi bibliographique, etc.).

La liste de noms de catégories prévue par le TEI-TERM est plus complète que celles du MicroMATER et du NTRF : elle en compte près de 150, et cette liste demeure ouverte. Le TEI-TERM préfère limiter le nombre de catégories primaires (les éléments) et considérer les autres comme des valeurs d'attribut (le plus souvent, l'attribut est appelé type=). Le MicroMATER le fait à un degré moindre, alors que le NTRF ne le fait pas du tout. Illustrons par un exemple cette différence de point de vue. Voici comment les formats NTRF et TEI-TERM se représentent un terme avec son synonyme :

NTRF : <terme> origan </terme>
<syn> marjolaine </syn>
TEI-TERM : <terme> origan </terme>
<terme type=syn> marjolaine </terme>

On voit donc que, dans ces formats, la distribution différente des catégories en éléments ou en valeurs d'attribut est le fait d'un point de vue méthodologique différent.

Contrairement au MicroMATER et au NTRF, le nom seul des catégories de TEI-TERM ne donne aucun indice sur la structure des données. Les noms ne se forment pas par concaténation de codes. Pour reprendre notre analogie, nous avons affaire à une langue isolante plutôt qu'agglutinante. La structure est plutôt explicitée par l'ordre d'apparition des champs et des étiquettes, ainsi que par les pointeurs.

La TEI est un environnement flexible qui facilite l'échange rapide entre bases de données terminologiques et des applications plus larges. Le format TEI-TERM nous paraît plus puissant et souple que le MicroMATER et le NTRF, bien qu'il demande un marquage plus lourd.

Tous ces formats peuvent paraître bien lourds et rébarbatifs, mais il ne faut pas oublier qu'ils seront le plus souvent invisibles au commun des terminologues. Le but premier de ces formats étant un but de conversion entre systèmes informatiques, ce sont plutôt les concepteurs de systèmes et de logiciels de terminologie qui auront à les manipuler directement et, espérons-le, à les dissimuler derrière des applications conviviales pour l'utilisateur.

Mais la nécessité de créer ces formats universels est une occasion de réflexion sur la nature et la structure des données spécifiques à la terminologie, sur les «universaux» peu discutés de la méthodologie terminographique et sur d'autres points faisant l'objet de divergences de méthode et qui par le fait résistent à une normalisation trop contraignante. Ainsi la décision de considérer telle ou telle catégorie comme une valeur d'attribut plutôt qu'un élément fait partie de ces choix méthodologiques.

Si les formats présentés peuvent paraître compliqués, c'est qu'ils veulent rester souples. C'est le propre de toute norme de chercher le compromis entre l'universel et le particulier, la rigueur et la souplesse, l'utilité et la simplicité.

BIBLIOGRAPHIE

Bryan, Martin. 1988. *SGML : An Author's Guide to the Standard Generalized Markup Language.* Wokingham (Angleterre): Addison-Wesley.

International Organization for Standardization. 1986. *Information processing - Text and office systems - Standard Generalized Markup Language* (*SGML*), ISO 8879-1986, Genève.

Melby, Alan K. 1991. «MicroMATER : A Proposed Standard Format for Exchanging Lexical/Terminological Data Files», *Meta*, vol. 36, nº 1:135-160.

Norwegian Council for Technical Terminology. 1991. *Nordic Terminological Record Format* (*NTRF*), Oslo, 3 documents (4 p., 9 p., 10 p).

Wright, Sue Ellen et Alan K. Melby. 1991. *TEI-TERM : An SGML-based Interchange Format for Terminological Data within the Framework of the Text Encoding Initiative.* Document présenté dans le cadre de l'International Symposium on Terminology and Documentation in Specialized Communication, Hull, Canada (7-8 octobre 1991).

J.F.