W3C Recommendation

Ce document est la traduction en français de la recommandation du W3C portant sur la Référence du vocabulaire SKOS "Système simple d'organisation de connaissances" (18 août 2009).

Ce document est une version traduite qui peut comporter des erreurs. La seule version normative est la version originale anglaise, "SKOS Simple Knowledge Organization System Reference" qui se trouve à : http://www.w3.org/TR/2009/REC-skos-reference-20090818/

- **Adresse** : http://sparna.fr/skos/SKOS-traduction-francais.html
- **Traduction** : Thomas Francart pour Sparna - <thomas [point] francart [at] sparna.fr>
- **Date de traduction** : octobre 2013
- **Dernière mise à jour** : avril 2014

SKOS : Système Simple d'Organisation de Connaissances Référence

Recommandation W3C du 18 août 2009

Cette version:
http://www.w3.org/TR/2009/REC-skos-reference-20090818/
Dernière version:
http://www.w3.org/TR/skos-reference
Versions précédentes:
http://www.w3.org/TR/2009/PR-skos-reference-20090615/
Editeurs:
Alistair Miles, STFC Rutherford Appleton Laboratory / University of Oxford
Sean Bechhofer, University of Manchester

Veuillez consulter l'**errata** de ce document, qui peut inclure des corrections normatives.

Voir aussi les **traductions**.

Résumé

Ce document définit le Système Simple d'Organisation de Connaissances ("Simple Knowledge Organization System" en anglais - SKOS), un modèle de données partagé pour échanger et relier des systèmes d'organisation de connaissances sur le Web.

De nombreux systèmes d'organisation de connaissances, tels que les thesauri, les taxonomies, les classifications ou les systèmes de vedettes matières partagent une structure similaire, et sont utilisés par des applications semblables. SKOS capture la plupart de ces ressemblances et les explicite, de façon à permettre l'échange de données et de technologies entre tout type d'applications.

Le modèle de données SKOS permet de réaliser le portage des systèmes d'organisation de connaissances existants vers le Web Sémantique, d'une façon à la fois standard et peu coûteuse. SKOS propose également un langage simple et intuitif pour développer et partager de nouveaux systèmes d'organisation de connaissances. Ce langage peut être utilisé seul ou combiné avec des langages formels de représentation de connaissances tels que le Web Ontology Language (OWL).

Ce document est la spécification normative du Système Simple d'Organisation de Connaissances. Il s'adresse aux personnes impliquées dans le design ou l'implémentation de systèmes d'information, et qui ont déjà une bonne connaissance des technologies du Web Sémantic, en particulier de RDF et OWL

Pour un guide informatif de l'utilisation de SKOS, consultez le [SKOS-PRIMER].

Synopsis

Dans SKOS, des **concepts** peuvent être identifiés par des URIs, **libellés** avec des chaînes de caractères, peuvent se voir assigner des **notations** (codes lexicaux), peuvent être **documentés** à l'aide de plusieurs types de notes, **reliés à d'autres concepts** et organisés dans des hiérarchies informelles et des réseaux associatifs, aggrégés dans des **schémas de concepts**, regroupés dans des **collections** libellées et/ou ordonnées, et **alignés** avec des concepts d'autres schémas de concepts.

[ouvrir le panneau d'accès rapide]

Status de ce Document

Ce chapitre décrit le statut de ce document au moment de sa publication. D'autres documents peuvent remplacer ce document. Une liste des publications courantes du W3C et les dernières révisions de ce rapport technique peuvent être trouvées dans l'index des rapports techniques du W3C à l'adresse http://www.w3.org/TR/.

Ce document est une recommendation développée par le Groupe de Travail sur le Déploiement du Web Sémantique du W3C, qui fait partie de l'Activité Web Sémantique du W3C. Ce document est le résultat de la prise en compte d'une modification éditoriale survenue lors de la revue de la Proposition de Recommandation : un exemple non-normatif et le texte le précédent a été enlevé car il suggérait une façon spécifique de référencer un système de notation (en l'occurence une notation symbolique) dans les libellés lorsque le système de notation ne correspond pas à la langue naturelle. Cette suggestion a été jugée inconsistante avec la Bonne Pratique 47 de l'IETF concernant l'utilisation des étiquettes d'identification des langues. Les utilisateurs devrait se référer au vocabulaire d'extension SKOS pour la gestion de systèmes de notation alternatifs. Un rapport d'implémentation documente les utilisations connues de SKOS pendant la période de revue de la Recommandation Candidate. Un Primer SKOS mis à jour est publié en même temps que cette Recommandation.

Modifications depuis la Proposition de Recommandation du 15 juin 2009:

- Suppression du paragraphe final et de l'exemple de la section 6.5.4 sur l'utilisation de "sous-étiquettes" de langues privées.
- Changements éditoriaux aux sections de référence et citation du Primer SKOS.

Les commentaires sur ce document peuvent être envoyés à l'adresse public-swd-wg@w3.org qui a une archive publique.

Ce document a été produit par un groupe qui a travaillé en suivant la politique W3C du 5 février 2004 concernant les brevets. Le W3C maintient une liste publique de toutes les annonces de brevets établie en lien avec les livrables du groupe de travail; cette page inclut également les étapes à suivre pour faire valoir un brevet. Une personne qui aurait connaissance d'un brevet contenant à son sens des Revendications Essentielles doit rendre publique cette information, conformément à la section 6 de la politique du W3C concernant les brevets.

Ce document a été revu par des membres du W3C, par des développeurs logiciels, et par d'autres groupes du W3C ou des organisations intéressées, et a été approuvé par le Directeur comme Recommandation du W3C. C'est un document pérenne et il peut être utilisé comme support de référence ou cité depuis un autre document. Le rôle du W3C, en produisant cette recommandation, est de mettre en lumière la spécification et d'en promouvoir le plus large déploiement. Ceci permet d'améliorer la fonctionnalité et l'interopérabilité du Web.

Table des Matières

[afficher le panneau d'accès rapide]

---

1. Introduction

1.1. Contexte et Motivation

Le Système Simple d'Organisation de Connaissances est un standard d'échange de données qui fait la synthèse entre plusieurs champs de connaissance, technologies et pratiques.

Les sciences de l'information et les sciences des bibliothèques possèdent un héritage ancien et reconnu dans le développement d'outils pour organiser de large collections d'objets tels que les livres ou les pièces de musée. Ces outils sont en général connus sous l'appellation de "Systèmes d'Organisation de Connaissances" (SOC) ou parfois de "vocabulaires contrôlés (et structurés)". Diverses traditions distinctes, bien que similaires, ont émergé au fil du temps, chacune supportée par une communauté de pratiques et un ensemble de standards. Des familles différentes de systèmes d'organisation de connaissances, comme les thesaurus, les classifications, les systèmes de vedettes-matières ou les taxonomies sont reconnues et utilisées par les systèmes d'information classiques et modernes. En pratique la frontière est floue entre les thesaurus, les classifications et les taxonomies, bien que quelques caractéristiques puissent distinguer ces différentes familles (voir par exemple [BS8723-3]). La philosophie générale de SKOS est que, en plus de ces caractéristiques distinctives, toutes ces familles partagent de nombreux points communs, et peuvent souvent être utilisées de façons similaires [SKOS-UCR]. Cependant, il n'existe pas aujourd'hui de standard pour représenter ces systèmes d'organisation de connaissances comme des données, permettant ainsi de les échanger entre systèmes d'information.

L'Activité Web Sémantique du W3C [SW] a stimulé un nouveau champ de recherche et de développement technologique, à la frontière entre les systèmes de bases de données, la logique formelle et le Web. Ce travail a conduit au développement de standards qui forment les fondations du Web Sémantique. RDF (Cadre pour la Description de Ressources) fourni un modèle et des syntaxes pour les données sur le Web [RDF-PRIMER]. RDFS (langage de description de vocabulaires RDF) et OWL (langage d'Ontologie pour le Web) fournissent à eux deux un langage de modélisation de (schéma de) données pour le Web [RDFS] [OWL-GUIDE]. Le Langage de requêtes et protocole SPARQL fournit un moyen standard pour interagir avec des données sur le Web [SPARQL].

Ces technologies ont des applications variées, car beaucoup de systèmes ont besoin d'un cadre commun pour publier, partager, échanger et intégrer ("fusionner") des données de sources différentes. La possibilité de lier des données de différentes sources est le point de départ de nombreux projets, et plusieurs communautés cherchent à créer de la valeur à partir de données auparavant réparties dans des sources différentes.

L'un des aspects de la vision d'un Web Sémantique est l'espoir de mieux organiser les vastes quantités d'informations non-structurées (c.à.d. à destination des humains) sur le Web, fournissant de nouvelles possibilités de découvrir et de partager ces informations. RDFS et OWL sont des langages de représentation des connaissances formellement définis, fournissant des possibilités d'exprimer des informations pour la réalisation de calculs, qui viennent en complément de et donne du sens à des informations déjà présentes sur le Web [RDF-PRIMER] [OWL-GUIDE]. Cependant, si l'on veut utiliser ces technologies sur de larges quantités d'information, on a besoin de construire des cartes détaillées de domaines de connaissance spécifiques, en plus d'une description précise (c.à.d. d'une annotation ou d'un cataloguage) de ressources d'information sur une grande échelle, ce qui pour une bonne part ne peut pas se faire automatiquement. L'expérience accumulée et les bonnes pratiques en sciences de l'information et des bibliothèques en terme d'organisation de l'information et des connaissances sont évidemment complémentaires et applicables à cette vision, comme le sont les nombreux systèmes d'organisation de connaissances déjà développés et utilisés, tels que les "Library of Congress Subject Headings" [LCSH] ou le Thesaurus AGROVOC de l'Organisation pour l'Alimentation et l'Agriculture des Nations Unies (FAO) [AGROVOC].

Le Système Simple d'Organisation de Connaissances cherche donc à établir un rapprochement entre plusieurs communautés de pratiques au sein des sciences de l'information et des bibliothèques, qui sont impliquées dans la conception et l'application de systèmes d'organisation de connaissances. De plus, SKOS cherche à établir un rapprochement entre ces communautés et le Web Sémantique, en transférant des modèles existants d'organisation de connaissances dans le contexte des technologies du Web Sémantique, et en fournissant une façon de migrer à faible coût des systèmes d'organisation de connaissances existants vers RDF.

Anticipant sur le futur, SKOS occupe une position intermédiaire entre l'exploitation et l'analyse d'informations non-structurées, l'organisation d'informations de façon informelle, à grande échelle et utilisant les médias sociaux, et la représentation formelle des connaissances. En rendant accessible l'expérience accumulée et les compétences en organisation de connaissances des sciences de l'information et des bibliothèques, en les rendant applicables et transférables au contexte technologique du Web Sémantique, d'une façon complémentaire aux technologies du Web Sémantique existantes (et en particulier aux systèmes formels de représentation des connaissances comme OWL), SKOS espère permettre de nombreuses et importantes applications, et amener de nouvelles recherches et de nouveaux développements aussi bien dans les technologies que dans les pratiques.

1.2. Présentation de SKOS

Le Système Simple d'Organisation de Connaissances est un modèle de données commun pour les systèmes d'organisation de connaissances tels que les thesaurus, les classifications, les systèmes de vedettes-matières ou les taxonomies. En utilisant SKOS, un système d'organisation de connaissances peut être exprimé **en données compréhensibles par une machine**. Il peut ensuite être échangé entre applications informatiques et publié dans un format compréhensible par les machines sur le Web.

Le modèle de données SKOS est formellement défini dans cette spécification comme une ontologie OWL Full [OWL-SEMANTICS]. Des données SKOS sont exprimées sous forme de triplets RDF [RDF-CONCEPTS], et peuvent être encodées en utilisant n'importe quelle syntaxe RDF (comme RDF/XML [RDF-XML] ou Turtle [TURTLE]). Pour plus d'informations sur le positionnement entre SKOS, RDF et OWL, voir la sous-section suivante ci-desssous.

Le modèle de données SKOS voit un système d'organisation de connaissances comme un **schéma de concepts** comprenant un ensemble de **concepts**. Ces schémas de concepts et ces concepts SKOS sont identifiés par des URIs, permettant à chacun de s'y référer de façon non-ambigüe depuis n'importe quel contexte, et les rendant ainsi partie intégrante du World Wide Web. Voir la Section 3. La Classe skos:Concept pour plus d'informations sur l'identifications et la description des concepts SKOS, et la Section 4. Schémas de Concepts pour plus d'informations sur les schémas de concepts.

Les concepts SKOS peuvent être **libellés** avec n'importe quel nombre de chaines de caractères lexicales (UNICODE), tels que "romantic love" ou "れんあい", dans n'importe quelle langue, telle que l'Anglais ou le Japonais (écrit ici en hiragana). L'un de ces libellés dans chaque langue peut être indiqué comme le libellé préférentiel pour cette langue, et les autres comme libellés alternatifs. Les libellés peuvent aussi être "cachés", ce qui est utile si un système d'organisation de connaissances est interrogé via un index textuel. Voir la Section 5. Libellés Lexicaux pour plus d'informations sur les propriétés de libellés lexicaux SKOS.

Les concepts SKOS peuvent se voir assigner une ou plusieurs **notations**, qui sont des codes lexicaux utilisés pour les identifier dans le contexte d'un schéma de concept donné. Si les URIs sont le moyen privilégié d'identifier les concepts SKOS dans un système informatique, les notations font le lien avec d'autres systèmes d'identification déjà en usage tels que les codes de classification des catalogues de bibliothèques. Voir la Section 6. Notations pour plus d'informations sur les notations.

Les concepts SKOS peuvent être **documentés** avec des notes de plusieurs types. Le modèle de données SKOS fournit un ensemble de base de propriétés de documentation, dont les notes d'application, les définitions et les notes éditoriales, entre autres. Cet ensemble n'est pas fait pour être exhaustif, mais pour fournir un cadre qui peut être étendu par d'autres pour supporter des types de notes plus spécifiques. Voir la Section 7. Propriétés de Documentation pour plus d'informations sur les notes.

Les concepts SKOS peuvent être **reliés** à d'autres concepts SKOS via des propriétés exprimant une relation sémantique. Le modèle de données SKOS fournit la possibilité d'exprimer des liens hiérarchiques et associatifs entre concepts SKOS. Là encore, comme les autres parties du modèle de données SKOS, ces liens peuvent être étendus par d'autres pour supporter des besoins plus précis. Voir la Section 8. Relations Sémantiques pour plus d'informations sur les liens entre concepts SKOS.

Les concepts SKOS peuvent être groupés au sein de **collections**, qui peuvent être libellées et/ou ordonnées. Cet aspect du modèle de données SKOS est prévu pour supporter les "node labels" (*note du traducteur : sans doute la notion de "facettes"*) dans les thesauri, et pour les situations dans lesquelles l'ordre d'un ensemble de concepts est important ou apporte une information supplémentaire. Voir la Section 9. Collections de Concepts pour plus d'informations sur les collections.

Les concepts SKOS peuvent être **alignés** avec d'autres concepts SKOS de schémas de concepts différents. Le modèle de données SKOS fournit quatre types de base de liens d'alignement : hiérarchique, associatif, d'équivalence proche et d'équivalence exacte. Voir la Section 10. Propriétés d'Alignement pour plus d'informations sur l'alignement.

Dernier point, une extension optionnelle de SKOS est définie dans l'Appendice B. SKOS eXtension pour les Libellés (SKOS-XL). SKOS-XL fournit plus de possibilités pour identifier, décrire et relier entre eux des entités lexicales.

1.3. SKOS, RDF et OWL

Les composantes du modèle de données SKOS sont des classes et des propriétés, et la structure et l'intégrité du modèle de données sont définies par les caractéristiques logiques et les interdépendances entre ces classes et ces propriétés. C'est sans doute l'un des aspects les plus puissants de SKOS, en même temps que celui qui peut porter le plus à confusion; SKOS peut en effet, dans certaines applications les plus évoluées, être utilisé en même temps que OWL pour exprimer et échanger de la connaissance sur un domaine. Cependant, SKOS n'est **pas** un langage formel de représentation des connaissances.

Pour comprendre cette distinction, il faut savoir que la "connaissance" explicitée dans une ontologie formelle est exprimée à l'aide d'un ensemble d'axiomes et de faits. Un thesaurus ou une classification sont d'une nature complètement différente, et n'expriment pas d'axiomes ou de faits. Un thesaurus ou une classification cherchent plutôt à identifier et à décrire, à travers la langue naturelle et d'autres moyens informels, un ensemble d'idées distinctes les unes des autres, qui sont parfois commodément appelées des "concepts". Ces "concepts" peuvent être arrangés et organisés en différentes structures, souvent des hiérarchies ou des liens associatifs. Ces structures, cependant, n'ont aucune sémantique formelle, et ne peuvent pas être interprétées comme des axiomes formels ou des faits sur le monde avec un degré de confiance suffisant. En réalité ces structures n'ont jamais eu cette prétention, leur seul objectif étant de fournir une carte pratique et intuitive d'un certain domaine, qui peut être utilisé comme une aide à l'organisation et à la recherche d'objets, tels que des documents, qui ont trait à ce domaine.

Rendre explicite la "connaissance" contenue dans un thesaurus ou une classification d'une façon formelle, demande que ce thesaurus ou cette classification soit *re-travaillé* sous forme d'une ontologie formelle. En d'autres termes, quelqu'un doit faire un travail de transformation de la structure et du contenu intellectuel du thesaurus ou de la classification en un ensemble d'axiomes et de faits formels. Ce travail de transformation est compliqué intellectuellement et demande du temps; il est donc coûteux. On peut retirer beaucoup d'avantages à utiliser les thesauri et leurs semblables tels quels, c'est-à-dire comme des structures commodes de navigation dans un domaine de connaissances. Les utiliser tels quels ne demande pas de travail additionnel et est donc moins coûteux. D'autant plus que certains SOC ne sont intrinséquement pas conçus pour donner une vision logique de leur domaine. Convertir ces SOC dans une représentation basée sur une logique formelle pourrait, en pratique, impliquer des changements qui résulteraient dans une représentation qui ne remplirait plus son objectif de départ.

OWL propose quant à lui un puissant langage de modélisation de données. Nous pouvons donc utiliser OWL pour construire un modèle de données pour représenter les thesauri et les classifications en eux-même. C'est exactement ce que SKOS fait. En utilisant cette approche, les "concepts" d'un thesaurus ou d'une classification sont modélisés comme des individus dans le modèle de données SKOS, les descriptions informelles et les liens entre concepts sont modélisés comme des faits sur ces individus, jamais comme des axiomes de classes ou de propriétés. Notez bien que ces faits sont des faits qui portent *sur* le thesaurus ou la classification *en eux-mêmes*, tels que "le concept X a le libellé préférentiel 'Y' et fait partie du thesaurus 'Z'"; ce ne sont **pas** des faits qui portent sur la façon dont le monde réel est organisé pour un certain domaine de connaissance, comme pourrait l'exprimer une ontologie formelle.

Les données SKOS sont ensuites exprimées à l'aide de triplets RDF. Le graphe RDF ci-dessous (en [TURTLE] tel que décrit dans la Section 1.7.3) exprime quelques faits à propos d'un thesaurus.

```
<A> rdf:type skos:Concept ;
  skos:prefLabel "amour"@fr ;
  skos:altLabel "adoration"@fr ;
  skos:broader <B> ;
  skos:inScheme <S> .

<B> rdf:type skos:Concept ;
  skos:prefLabel "émotion"@fr ;
  skos:altLabel "sensation"@fr ;
  skos:topConceptOf <S> .

<S> rdf:type skos:ConceptScheme ;
  dct:title "Mon Premier Thesaurus" ;
  skos:hasTopConcept <B> .
```

Ce point est crucial dans la compréhension de la définition formelle du modèle de données SKOS et de la façon dont il pourrait être implémenté par des logiciels. Ce point est également crucial pour des utilisations plus avancées de SKOS, et en particulier celles où SKOS et OWL sont combinés dans une conception hybride mêlant formel et semi-formel.

Du point de vue d'un utilisateur, cependant, les frontières entre les systèmes de représentation des connaissances formels et les systèmes d'organisation des connaissances informels ou semi-formels peuvent naturellement devenir floues. En d'autres termes, cela peut ne pas être pertinent pour un utilisateur que `<A>` et `<B>` dans le graphe ci-dessous soient des individus (instances de `skos:Concept`), et `<C>` et `<D>` des classes (instances de `owl:Class`) .

```
<A> rdf:type skos:Concept ;
  skos:prefLabel "amour"@fr ;
  skos:broader <B> .

<B> rdf:type skos:Concept ;
  skos:prefLabel "émotion"@fr .

<C> rdf:type owl:Class ;
  rdfs:label "mammifères"@fr ;
  rdfs:subClassOf <D> .

<D> rdf:type owl:Class ;
  rdfs:label "animaux"@fr .
```

Un système d'information qui prend en compte le modèle de données SKOS devra cependant faire cette distinction.

Les schémas RDF pour SKOS et SKOS eXtension pour les Libellés (SKOS-XL) sont décrits dans l' Appendice C. Documents d'Espace de noms SKOS et SKOS-XL. Notez que, étant donné que certaines contraintes ne peuvent pas être capturées entièrement par le schéma, le document RDF/XML ne fournit qu'un sous-ensemble normatif de cette spécification.

1.4. Compatibilité et Intégrité

Dans les sémantiques RDF et OWL Full, le sens formel (*l'interprétation*) d'un graphe RDF est une valeur de vérité [RDF-SEMANTICS] [OWL-SEMANTICS], c'est-à-dire qu'un graphe RDF est interprété soit comme vrai soit comme faux.

En général, un graphe RDF est dit *incompatible* s'il n'est pas possible qu'il soit vrai. En d'autres termes, un graphe RDF est incompatible s'il contient une contradiction.

En utilisant seulement les vocabulaires RDF et RDFS, il n'est tout simplement pas possible d'exprimer des assertions contradictoires entre elles. Si le vocabulaire OWL est utilisé en plus, on peut alors exprimer une contradiction de plusieurs façons, par exemple considérez le graphe RDF ci-dessous.

```
<Chien> rdf:type owl:Class .
<Chat> rdf:type owl:Class .
<Chien> owl:disjointWith <Chat> .
```

```
<chienchat> rdf:type <Chien> , <Chat> .
```

Le graphe exprime que `<Chien>` et `<Chat>` sont toutes deux des classes, et qu'elles sont disjointes, c'est-à-dire qu'elles n'ont aucune instance en commun. Ceci est contredit par l'assertion que `<chienchat>` a pour type à la fois `<Chien>` et `<Chat>`. Aucune interprétation OWL Full ne peut satisfaire ce graphe, il n'est donc **pas** compatible avec OWL Full.

Quand OWL Full est utilisé comme langage de représentation de connaissances, la notion d'incompatibilité est utile pour détecter des contradictions dans les axiomes et les faits exprimés dans une ontologie. En trouvant et corrigeant ces incompatibilités on atteint les détails d'un domaine de connaissances, et on arrive à une meilleure modélisation de ce domaine, de laquelle on pourra tirer des inférences pertinentes et valides.

Quand OWL Full est utilisé comme langage de modélisation (c'est-à-dire comme schéma), la notion d'incompatibilité est là encore utile mais d'une façon différente. On ne s'intéresse plus dans ce cas à la validité logique de la connaissance humaine en tant que telle. On s'intéresse simplement à la définition formelle d'un modèle de données, de façon à pouvoir établir de façon certaine si une donnée correspond (c'est-à-dire se conforme) au modèle de données. Si la donnée est incompatible par rapport au modèle de données, alors la donnée ne correspond pas.

Dans ces cas on ne s'intéresse donc pas à la correspondance entre une certaine donnée et le monde réel, c'est-à-dire au fait qu'une certaine donnée serait vraie ou fausse au sens absolu du terme. On s'intéresse simplement au fait qu'une certaine donnée corresponde ou pas au modèle de données, car l'interopérabilité entre applications dépend de la conformité des données échangées par rapport à un modèle de données commun.

Une autre notion qui exprime cette même idée est celle d'*integrité*. Les conditions d'intégrité sont des assertions qui font partie de la définition formelle d'un modèle de données, et qui sont utilisées pour savoir si une certaine donnée est compatible avec le modèle; par exemple l'assertion que `<Chat>` et `<Chien>` sont des classes disjointes peut être considérée comme une condition d'intégrité d'un modèle de données. Etant donnée cette condition, les données ci-dessous sont donc incompatibles.

```
<chienchat> rdf:type <Chien> , <Chat> .
```

La définition du modèle de données SKOS dans ce document contient un petit nombre d'assertions qui sont des conditions d'intégrité. Ces conditions d'intégrité sont ici pour favoriser l'interopérabilité, en définissant les cas dans lesquelles des données sont **incompatibles** par rapport au modèle de données SKOS. Des outils peuvent alors être implémentés pour vérifier que tout ou partie de ces conditions d'intégrité sont respectées par des données, et donc pour savoir si ces données sont compatibles avec le modèle de données SKOS.

Ces conditions d'intégrité font partie intégrante de la définition formelle des classes et des propriétés du modèle de données SKOS. Elles sont cependant présentées à part du reste de la définition formelle car elles ont un objectif différent. Les conditions d'intégrité servent principalement à déterminer si des données sont compatibles avec le modèle de données SKOS. Toutes les autres assertions dans la définition du modèle de données SKOS servent **uniquement** de base aux inférences logiques. (Voir également la sous-section suivante.)

Les conditions d'intégrité sont définies dans le modèle de données SKOS de façon indépendante de leurs possibles stratégies d'implémentation, autant que faire ce peut. Il y a en effet plusieurs façons différentes d'implémenter une procédure pour vérifier les incompatibilités avec le modèle de données SKOS. Les incompatibilités peuvent être trouvées en utilisant un raisonneur OWL. Certaines peuvent également être déterminées en cherchant des motifs particuliers dans les données, ou par une stratégie hybride (en trouvant des inférences avec un raisonneur RDFS ou OWL, puis en cherchant des motifs particuliers dans le graphe inféré).

Il y a moins de conditions d'intégrité dans le modèle de données SKOS que l'on pourrait s'y attendre, en particulier pour ceux qui seraient habitués au monde fermé des bases de données. Voir également la sous-section suivante, et les notes dans les sections 3 à 10 ci-dessous.

1.5. Inférence, Dépendance et Hypothèse du Monde Ouvert

Ce document définit le modèle de données SKOS en utilisant une ontologie OWL Full. D'autres façons de définir le modèle de données SKOS auraient pu être utilisées, par exemple un modèle entité-relation, ou un modèle de classes UML. Bien qu'OWL Full semble proche intuitivement de ces autres approches pour modéliser, il y a cependant une distinction fondamentale.

RDF et OWL Full sont faits pour des systèmes où les données peuvent être largement distribuées (typiquement, le Web). Plus un tel système grossit, moins il devient possible de savoir où sont localisées toutes les données du système, jusqu'à ce que cela devienne virtuellement impossible. C'est pourquoi on ne peut généralement pas supposer que les données qu'on obtient depuis un tel système sont complètes. Si une partie des données semble manquante, on doit supposer, en général, que la donnée *pourrait* exister ailleurs dans le système. Cette hypothèse, pour faire simple, est connue sous le nom d'**hypothèse de monde ouvert** [OWL-GUIDE].

En pratique, cela signifie que, pour un modèle de données défini comme une ontologie OWl Full, certaines définitions peuvent être contre-intuitives. On ne peut tirer aucune conclusion d'une absence de données, et supprimer quelque chose ne rendra jamais les données restantes incompatibles. Ceci est illustré, par exemple, par la définition de `skos:semanticRelation` dans la Section 8 ci-dessous. La propriété `skos:semanticRelation` est définie comme ayant pour domaine et portée `skos:Concept`. Ces définitions de domaine d'ensemble d'arrivée donnent lieu à des **inférences**. Considérez le graphe ci-dessous.

```
<A> skos:semanticRelation <B>.
```

Dans ce cas, le graphe ci-dessus implique le graphe suivant.

```
<A> rdf:type skos:Concept .
<B> rdf:type skos:Concept .
```

On n'a donc pas besoin de dire ici *explicitement* que `<A>` et `<B>` sont instances de `skos:Concept`, car ces assertions sont déduites de la définition de `skos:semanticRelation`.

Dans le modèle de données SKOS, la plupart des définitions ne sont **pas** des règles d'intégrité, mais des assertions logiques de dépendance entre différentes parties du modèle de données, qui (sous l'hypothèse de monde ouvert), donnent lieu à un certain nombre d'inférences simples. Ceci est illustré, par exemple, par la déclaration de la Section 7 ci-dessous qui stipule que `skos:broader` et `skos:narrower` sont des propriétés inverses. Cette déclaration veut dire que

```
<A> skos:narrower <B> .
```

implique

```
<B> skos:broader <A> .
```

Pris seuls, ces deux graphes sont compatibles avec le modèle de données SKOS.

Les systèmes d'organisation de connaissances tels que les thesauri et les classifications peuvent être utilisés dans de nombreuses situations, et un système d'organisation de connaissances particulier peut être utilisé dans de nombreux systèmes d'information différents. En définissant le modèle de données SKOS comme une ontologie OWL Full, le Web Sémantique peut servir de media pour publier, échanger, partager et relier des données utilisant ces systèmes d'organisation de connaissances. C'est pour cette raison, pour le niveau d'expressivité que permet OWL Full comme langage de modélisation, et pour avoir la possibilité d'utiliser les thesauri, les classifications, etc., de la même façon que les ontologies formelles, que OWL Full a été choisi pour définir le modèle de données SKOS. L'hypothèse de monde ouvert est donc une prémisse fondamentale du modèle de données SKOS, et doit être gardée en tête tout au long de la lecture de ce document.

Voir également [RDF-PRIMER] et [OWL-GUIDE].

1.6. Logique de Conception

Comme indiqué ci-dessus, la notion de Système d'Organisation de Connaissances recouvre une large palette d'artefacts différents. Il y a donc un risque de sur-spécification du schéma SKOS, qui empêcherait alors son utilisation dans certaines applications. Pour éviter cela, en cas de doute à propos de l'inclusion d'une contrainte formelle dans le modèle (voir par exemple la discussion sur `skos:hasTopConcept`), la contrainte n'a pas été exprimée formellement. Dans certains cas, bien qu'aucune contrainte formelle ne soit exprimée, des conventions d'usage sont recommendées. Les applications qui auraient besoin de contraintes additionnelles peuvent définir des extensions au vocabulaire SKOS. Voir également le [SKOS-PRIMER].

1.7. Comment Lire ce Document

Ce document définit formellement le Système Simple d'Organisation de Connaissances comme une ontologie OWL Full. Les éléments du modèle de données SKOS sont des classes et des propriétés OWL, et un Identifiant Uniforme de Ressource (URI) est donné pour chacune de ces classes et de ces propriétés pour qu'elles puissent être utilisées de façon non-ambigüe sur le Web. Cet ensemble d'URI est le vocabulaire SKOS.

Le vocabulaire SKOS complet est donné dans la section 2 ci-dessous. Les sections 3 à 10 définissent ensuite formellement le modèle de données SKOS. La définition du modèle de données est découpée en plusieurs morceaux uniquement pour des raisons pratiques. Chacune des sections 3 à 10 suit une organisation similaire:

- **Préambule** — introduit de façon informelle les idées principales de la section.
- **Vocabulaire** — liste les URIs du vocabulaire SKOS définies dans cette section.

- **Définitions des Classes & Propriétés** — définit formellement les caractéristiques et relations logiques entre les classes et les propriétés identifiées par ces URIs.
- **Conditions d'intégrité** — donne les conditions d'intégrité le cas échéant.
- **Exemples** — donne des exemples basiques, à la fois de données qui **sont** compatibles avec le modèle de données SKOS, et (quand c'est nécessaire) de données qui ne sont **pas** compatibles avec le modèle de données SKOS.
- **Notes** — présente les notes et discussions additionnelles.

1.7.1. Définitions formelles

La plupart des définitions de classes et de propriétés données dans ce document auraient pu l'être sous forme de triplets RDF, en utilisant les vocabulaires RDF, RDFS et OWL. Cependant cela n'est pas possible pour un petit nombre d'entre elles, soit à cause de limites à l'expressivité de OWL Full ou à cause de l'absence d'URIs standards pour certaines classes. Pour rendre ce document plus lisible, plutôt que de mélanger des triplets RDF et d'autres notations, les définitions formelles et les conditions d'intégrité sont toutes indiquées dans des phrases en langage naturel.

Le style de ces phrases suit le style utilisée dans [RDFS], et devrait être clair pour un lecteur familier avec RDF et OWL.

Donc, par exemple, "`ex:Personne` est une instance de `owl:Class`" signifie:

```
ex:Personne rdf:type owl:Class .
```

"`ex:aPourParent` et `ex:aPourMere` sont chacune instance de `owl:ObjectProperty`" signifie:

```
ex:aPourParent rdf:type owl:ObjectProperty .
ex:aPourMere rdf:type owl:ObjectProperty .
```

"`ex:aPourMere` est une sous-propriété de `ex:aPourParent`" signifie:

```
ex:aPourMere rdfs:subPropertyOf ex:aPourParent .
```

"le `rdfs:range` de `ex:aPourParent` est la classe `ex:Personne`" signifie:

```
ex:aPourParent rdfs:range ex:Personne .
```

Quand certains éléments formels du modèle de données SKOS ne peuvent pas être exprimés en utilisant les vocabulaires RDF, RDFS ou OWL, la façon de traduire ces éléments en conditions formelles sur l'interprétation d'un vocabulaire RDF devrait être clair pour un lecteur avec une connaissance basique des sémantiques de RDF et OWL (par exemple, dans la Section 5, "Une ressource n'a pas plus d'une valeur de `skos:prefLabel` par étiquette de langue" signifie que, pour toute ressource x, il n'y a pas deux membres de l'ensemble { y | <x,y> est dans IEXT(I(`skos:prefLabel`)) } qui partagent la même étiquette de langue, avec I et IEXT étant des fonctions telles que définies dans [RDF-SEMANTICS]).)

1.7.2. Abbréviation des URI

Les URIs entières sont indiquées dans le texte de ce document en police monospace, entourées par des chevrons, par exemple `<http://example.org/ns/example>`. Les URIs relatives sont indiquées de la même façon, et sont relatives à la racine `<http://example.org/ns/>`, par exemple `<example>` et `<http://example.org/ns/example>` sont la même URI.

Des URIs sont également indiquées dans le texte de ce document sous leur forme abrégée. Les URIs abbréviées sont indiquées en police monospace sans chevron, et doivent être ramenées à leur forme entière en utilisant la table de correspondance ci-dessous.

Table 1. Abbréviations des URIs

URI	Abbréviation
http://www.w3.org/2004/02/skos/core#	**skos:**
http://www.w3.org/1999/02/22-rdf-syntax-ns#	rdf:
http://www.w3.org/2000/01/rdf-schema#	rdfs:
http://www.w3.org/2002/07/owl#	owl:

Donc par exemple, `skos:Concept` est une abbréviation de `<http://www.w3.org/2004/02/skos/core#Concept>`.

1.7.3. Exemples

Les exemples de graphes RDF sont donnés en utilisant le Langage de Triplets RDF Turtle [TURTLE]. Tous les exemples sont supposés être précédés de la déclaration des préfixes d'URIs suivante:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@base <http://example.org/ns/> .
```

Par conséquent, l'exemple donné ci-dessous

Exemple 1

```
<MonConcept> rdf:type skos:Concept .
```

Est équivalent au document Turtle suivant

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@base <http://example.org/ns/> .

<MonConcept> rdf:type skos:Concept .
```

qui est équivalent au document RDF/XML suivant [RDF-XML]

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:skos="http://www.w3.org/2004/02/skos/core#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xml:base="http://example.org/ns/">

    <skos:Concept rdf:about="MonConcept"/>

</rdf:RDF>
```

qui est équivalent au document N-TRIPLES suivant [NTRIPLES]

```
<http://example.org/ns/MonConcept> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2004/02/skos/core#Concept> .
```

Notez l'utilisation en Turtle des caractères ";" et "," pour abbrévier les triplets ayant le même sujet ou le même prédicat. Certains exemples utilisent également la syntaxe Turtle "(...)", qui représente une Collection RDF.

1.8. Conformité

Cette spécification ne définit pas de notion formelle de conformité.

Cependant, un graphe RDF sera **incompatible** avec le modèle de données SKOS si ce graphe et le modèle de données SKOS (tel que défini formellement ci-dessous) mis ensemble donnent lieu à une contradiction logique.

Lorsque des URIs sont utilisés pour identifier des ressources de type `skos:Concept`, `skos:ConceptScheme`, `skos:Collection` ou `skosxl:Label`, cette spécification **ne requiert pas** de comportement particulier lorsque ces URIs sont déréférencées sur le Web [WEBARCH]. Il est cependant fortemment recommandé que les producteurs de données SKOS suivent les recommandations données dans [COOLURIS] et [RECIPES].

---

2. Espace de Noms et Vocabulaire SKOS

L'URI de l'espace de noms SKOS est:

- **http://www.w3.org/2004/02/skos/core#**

Le vocabulaire SKOS est un ensemble d'URIs, données dans la colonne de gauche de la table ci-dessous.

Table 1. Vocabulaire SKOS

URI	Définition
skos:Concept	Section 3. La Classe skos:Concept
skos:ConceptScheme	Section 4. Schémes de Concepts
skos:inScheme	Section 4. Schémes de Concepts
skos:hasTopConcept	Section 4. Schémes de Concepts
skos:topConceptOf	Section 4. Schémes de Concepts
skos:altLabel	Section 5. Libellés Lexicaux
skos:hiddenLabel	Section 5. Libellés Lexicaux
skos:prefLabel	Section 5. Libellés Lexicaux
skos:notation	Section 6. Notations
skos:changeNote	Section 7. Propriétés de Documentation
skos:definition	Section 7. Propriétés de Documentation
skos:editorialNote	Section 7. Propriétés de Documentation
skos:example	Section 7. Propriétés de Documentation
skos:historyNote	Section 7. Propriétés de Documentation
skos:note	Section 7. Propriétés de Documentation
skos:scopeNote	Section 7. Propriétés de Documentation
skos:broader	Section 8. Relations Sémantiques
skos:broaderTransitive	Section 8. Relations Sémantiques
skos:narrower	Section 8. Relations Sémantiques
skos:narrowerTransitive	Section 8. Relations Sémantiques
skos:related	Section 8. Relations Sémantiques
skos:semanticRelation	Section 8. Relations Sémantiques
skos:Collection	Section 9. Collections de Concepts
skos:OrderedCollection	Section 9. Collections de Concepts
skos:member	Section 9. Collections de Concepts
skos:memberList	Section 9. Collections de Concepts
skos:broadMatch	Section 10. Propriétés d'Alignement
skos:closeMatch	Section 10. Propriétés d'Alignement
skos:exactMatch	Section 10. Propriétés d'Alignement
skos:mappingRelation	Section 10. Propriétés d'Alignement
skos:narrowMatch	Section 10. Propriétés d'Alignement
skos:relatedMatch	Section 10. Propriétés d'Alignement

Toutes les URIs dans le vocabulaire SKOS sont construites en ajoutant un nom local (par exemple "prefLabel") à l'URI de l'espace de noms SKOS.

Voir également la présentation de SKOS dans l'Appendice B et le panneau d'accès rapide.

---

3. La Classe skos:Concept

3.1. Préambule

La classe `skos:Concept` est la classe de tous les concepts SKOS.

Un concept SKOS peut être vu comme une idée ou une notion; une unité de pensée. Cependant, ce que constitue une unité de pensée est subjectif, et cette définition est faite pour être suggestive plutôt que restrictive.

La notion de concept SKOS est utile pour décrire la structure conceptuelle ou intellectuelle d'un système d'organisation de connaissances, et pour se référer à certaines idées ou notions précises dans un SOC.

Notez que, comme SKOS est conçu pour véhiculer des SOC semi-formels, tels que les thesauri et les classifications, un certain degré de flexibilité a été inclu dans la définition formelle de cette classe.

Voir le [SKOS-PRIMER] pour plus d'exemples d'identification et de description des concepts SKOS.

3.2. Vocabulaire

`skos:Concept`

3.3. Définitions des Classes & et des Propriétés

S1	`skos:Concept` est une instance de `owl:Class`.

3.4. Exemples

Le graphe ci-dessous exprime que `<MonConcept>` est un concept SKOS (c.à.d. une instance de `skos:Concept`).

Exemple 2 (compatible)
`<MonConcept> rdf:type skos:Concept .`

3.5. Notes

3.5.1. Concepts SKOS, Classes OWL et Propriétés OWL

Mise à part l'assertion que `skos:Concept` est une instance de `owl:Class`, cette spécification ne fait **pas** d' assertion supplémentaire à propos de la relation formelle entre la classe des concepts SKOS et la classe des classes OWL. La décision de ne **pas** faire de telle assertion a été prise pour laisser la liberté aux applications d'explorer plusieurs choix de conceptions pour travailler avec SKOS combiné avec OWL.

Dans l'exemple de graphe ci-dessous, `<MonConcept>` est une instance de `skos:Concept` **et** une instance of `owl:Class`.

Exemple 3 (compatible)
`<MonConcept> rdf:type skos:Concept , owl:Class .`

Cet exemple est **compatible** avec le modèle de données SKOS.

De la même façon, cette spécification ne fait **pas** d'assertion concernant la relation formelle entre la classe des concepts SKOS et la classes des propriétés OWL.

Dans le graphe d'exemple ci-dessous, `<MonConcept>` est une instance de `skos:Concept` **et** et une instance de `owl:ObjectProperty`.

Exemple 4 (compatible)
`<MonConcept> rdf:type skos:Concept , owl:ObjectProperty .`

Cet exemple est **compatible** avec le modèle de données SKOS.

---

4. Schémas de Concepts

4.1. Préambule

Un schéma de concepts SKOS peut être compris comme une aggrégation d'un ou plusieurs concepts SKOS. Les relations sémantiques (liens) entre ces concepts peuvent également être vues comme faisant partie d'un schéma de concepts. Cette définition est cependant faite pour être suggestive plutôt que restrictive, et le modèle de données formel décrit ci-dessous comporte une certaine flexibilité.

La notion de schéma de concept est utile pour travailler avec des données d'une source inconnue, et pour travailler avec des données qui décrivent deux (ou plus) systèmes d'organisation des connaissances différents.

Voir le [SKOS-PRIMER] pour plus d'exemples sur l'identification et la description des schémas de concepts.

4.2. Vocabulaire

`skos:ConceptScheme`
`skos:inScheme`
`skos:hasTopConcept`
`skos:topConceptOf`

4.3. Définitions des Classes & et des Propriétés

S2	`skos:ConceptScheme` est une instance de `owl:Class`.
S3	`skos:inScheme`, `skos:hasTopConcept` et `skos:topConceptOf` sont chacune instances de `owl:ObjectProperty`.
S4	Le `rdfs:range` de `skos:inScheme` est la classe `skos:ConceptScheme`.
S5	Le `rdfs:domain` de `skos:hasTopConcept` est la classe`skos:ConceptScheme`.
S6	Le `rdfs:range` de `skos:hasTopConcept` est la classe`skos:Concept`.
S7	`skos:topConceptOf` est une sous-propriété de `skos:inScheme`.
S8	`skos:topConceptOf` est `owl:inverseOf` de la propriété `skos:hasTopConcept`.

4.4. Conditions d'Intégrité

S9	`skos:ConceptScheme` est disjointe avec `skos:Concept`.

4.5. Exemples

Le graphe ci-dessous décrit un schéma de concepts avec deux concepts SKOS, dont l'un est un concept de premier niveau dans ce schéma.

Exemple 5 (compatible)

```
<MonScheme> rdf:type skos:ConceptScheme ;
  skos:hasTopConcept <MonConcept> .

<MonConcept> skos:topConceptOf <MonScheme> .

<UnAutreConcept> skos:inScheme <MonScheme> .
```

4.6. Notes

4.6.1. Systèmes Ouverts vs. Systèmes Fermés

La notion d'un schéma de concepts SKOS correspond **grosso modo** à la notion d'un thesaurus, d'une classification, d'un système de vedettes ou d'un autre système d'organisation de connaissances.

Cependant, dans la plupart des systèmes d'information actuels, un thesaurus ou une classification sont traitées comme des **systèmes fermés** — les unités conceptuelles définies dans ce système ne peuvent pas faire partie d'autres systèmes (même si elles peuvent être *alignées* avec les unités d'autres systèmes).

Bien que SKOS propose une approche similaire, il n'y a **pas** de condition empêchant un concept SKOS d'appartenir à zéro, un, ou plus d'un schéma de concepts.

Donc, par exemple, dans le graphe ci-dessous le concept SKOS `<MonConcept>` appartient à deux schémas de concepts différents — ce qui est **compatible** avec le modèle de données SKOS.

Exemple 6 (compatible)

```
<MonScheme> rdf:type skos:ConceptScheme .

<UnAutreScheme> rdf:type skos:ConceptScheme ;
  owl:differentFrom <MonScheme> .

<MonConcept> skos:inScheme <MonScheme> , <UnAutreScheme> .
```

Cette flexibilité est voulue car elle permet, par exemple, de décrire de nouveaux schémas de concepts en reliant ensemble deux schémas de concepts ou plus.

Notez également qu'il n'y a pas de possibilité de fermer la frontière d'un schéma de concepts. Donc, bien qu'il soit possible en utilisant `skos:inScheme` de dire que les concepts SKOS X, Y et Z appartiennent au schéma de concepts A, il n'y a pas de possibilité de dire que **seuls** X, Y et Z appartiennent à A.

Par conséquent, bien que SKOS puisse être utilisé pour **décrire** un schéma de concepts, SKOS ne fournit par de mécanisme pour complètement **définir** un schéma de concepts.

4.6.2. Schémas de Concepts SKOS et Ontologies OWL

Cette spécification ne fait **pas** d'assertion sur la relation formelle entre la classe des schémas de concepts SKOS et la classe des ontologies OWL. La décision de ne **pas** faire de telle assertion a été prise pour laisser la liberté aux applications d'explorer plusieurs choix de conceptions pour travailler avec SKOS combiné avec OWL. [OWL-GUIDE].

Dans le graphe d'exemple ci-dessous, `<MonScheme>` est à la fois un schéma de concepts SKOS et une ontologie OWL. Ceci est **compatible** avec le modèle de données SKOS.

Exemple 7 (compatible)

```
<MonScheme> rdf:type skos:ConceptScheme , owl:Ontology .

<MonConcept> skos:inScheme <MonScheme> .
```

4.6.3. Concepts Racines et Relations Sémantiques

La propriété `skos:hasTopConcept` est, par convention, utilisée pour relier un schéma de concepts au(x) concept(s) qui sont à la racine des relations hiérarchiques pour ce schéma. Cependant, il n'y a pas de condition d'intégrité contraignant cette convention. Par conséquent, le graphe ci-dessous, bien que n'adhérent pas à strictement parlé à la convention d'usage de `skos:hasTopConcept`, est toutefois **compatible** avec le modèle de données SKOS.

Exemple 8 (compatible)

```
<MonScheme> skos:hasTopConcept <MonConcept> .
<MonConcept> skos:broader <UnAutreConcept> .
<UnAutreConcept> skos:inScheme <MonScheme> .
```

Une application peut rejeter de telles données mais n'est pas obligée de le faire.

4.6.4. Contenu d'un Schéma de Concepts et Relations Sémantiques

Un lien entre deux concepts SKOS n'implique **pas** l'appartenance de ces deux concepts au même schéma de concepts. Ceci est illustré dans le graphe d'exemple ci-dessous.

Exemple 9 (non-implication)

```
<A> skos:narrower <B> .
<A> skos:inScheme <MonScheme> .
```

n'implique pas

```
<B> skos:inScheme <MonScheme> .
```

Voir aussi la Section 8 ci-dessous.

4.6.5. Domaine de skos:inScheme

Notez qu'**aucun domaine n'est déclaré** pour la propriété `skos:inScheme`, c.à.d. que son domaine est en pratique la classe de toutes les ressources (`rdfs:Resource`). La décision de ne déclarer aucun domaine a été prise pour donner de la flexibilité, permettant aux extensions de SKOS de définir de nouvelles classes de ressources tout en utilisant toujours `skos:inScheme` pour les lier à un `skos:ConceptScheme`. Voir aussi l'exemple 82 ci-dessous.

---

5. Libellés Lexicaux

5.1. Préambule

Un libellé lexical est une chaine de caractères UNICODE, telle que "romantic love" ou "れんあい", dans une langue naturelle donnée, telle que l'Anglais ou le Japonais (écrit ici en hiragana).

Le Système Simple d'Organisation des Connaissance fournit un vocabulaire de base pour associer des libellés lexicaux avec des ressources de n'importe quel type. En particulier, SKOS permet la distinction entre les libellés lexicaux préférentiels, alternatifs et "cachés" pour n'importe quelle ressource.

Les libellés préférentiels et alternatifs sont utiles pour générer ou créer des représentations lisibles par l'humain d'un système d'organisation des connaissances. Ces libellés donnent les indications les plus fortes sur le sens d'un concept SKOS.

Les libellés cachés sont utiles lorsqu'un utilisateur interagit avec un système d'organisation de connaissances via une fonction de recherche textuelle. L'utilisateur peut, par exemple, entrer des libellés mal orthographiés pour chercher un concept donné. Si la requête mal orthographiée peut être trouvée dans un libellé caché, l'utilisateur sera capable de trouver le bon concept, mais les libellés cachés ne seront pas présentés à l'utilisateur (de façon à ne pas encourager plus d'erreurs).

Formellement, un libellé lexical est un litéral RDF simple [RDF-CONCEPTS]. Un litéral RDF simple est composé d'une forme lexicale, qui est une chaine de caractères UNICODE, et d'une étiquette de langue optionnelle, qui est une chaine de caractères répondant à la syntaxe définie par [BCP47].

Voir le [SKOS-PRIMER] pour plus d'exemple de libellés sur les concepts SKOS. Notez en particulier que les exemples ci-dessous servent uniquement d'illustration des possibilités générales du modèle de données SKOS, et n'indiquent **pas** nécessairement les bonnes pratiques pour la création de libellés avec des étiquettes de langue différentes. La Référence SKOS vise à établir un modèle de données qui serait applicable dans de nombreuses situations, et qui puisse être ensuite affiné et/ou contraint par les conventions d'usage de situations spécifiques. Les usages propres à une application ou à une langue donnée par rapport aux libellés et aux étiquettes de langue ne font pas partie du périmètre de la Reference SKOS.

5.2. Vocabulaire

`skos:prefLabel`
`skos:altLabel`
`skos:hiddenLabel`

5.3. Définitions des Classes & et des Propriétés

S10	`skos:prefLabel`, `skos:altLabel` et `skos:hiddenLabel` sont chacune instances de `owl:AnnotationProperty`.
S11	`skos:prefLabel`, `skos:altLabel` et `skos:hiddenLabel` sont chacune sous-propriétés `rdfs:label`.
S12	Le `rdfs:range` de `skos:prefLabel`, `skos:altLabel` et `skos:hiddenLabel` est la classe des litéraux RDF simples.

5.4. Conditions d'Intégrité

S13	`skos:prefLabel`, `skos:altLabel` et `skos:hiddenLabel` sont des propriétés disjointes deux à deux.
S14	Une ressource n'a pas plus d'une valeur de `skos:prefLabel` par étiquette de langue.

5.5. Exemples

Le graphe suivant est **compatible**, et illustre la création de libellés lexicaux dans deux langues différentes (Français et Anglais).

Exemple 10 (compatible)

```
<MaResource>
  skos:prefLabel "animals"@en ;
  skos:altLabel "fauna"@en ;
  skos:hiddenLabel "aminals"@en ;
  skos:prefLabel "animaux"@fr ;
  skos:altLabel "faune"@fr .
```

Le graphe suivant est **compatible** et illustre la création de libellés lexicaux dans quatre variations différentes (du Japonais écrit en kanji, en script hiragana, en script katakana ou avec des caractère latins (rōmaji)).

Exemple 11 (compatible)

```
<UnAutreResource>
  skos:prefLabel "東"@ja-Hani ;
  skos:prefLabel "ひがし"@ja-Hira ;
  skos:altLabel "あずま"@ja-Hira ;
  skos:prefLabel "ヒガシ"@ja-Kana ;
  skos:altLabel "アズマ"@ja-Kana ;
  skos:prefLabel "higashi"@ja-Latn ;
  skos:altLabel "azuma"@ja-Latn .
```

Le graphe suivant n'est **pas compatible** avec le modèle de données SKOS, car deux libellés lexicaux préférentiels ont été créés avec la même étiquette de langue.

Exemple 12 (incompatible)

```
<Amour> skos:prefLabel "amour"@fr ; skos:prefLabel "adoration"@fr .
```

Le graphe suivant n'est **pas compatible** avec le modèle de données SKOS, car il y a un clash entre les libellés lexicaux préférentiels et alternatifs.

Exemple 13 (incompatible)

```
<Amour> skos:prefLabel "amour"@fr ; skos:altLabel "amour"@fr .
```

Le graphe suivant n'est **pas compatible** avec le modèle de données SKOS, car il y a un clash entre les libellés lexicaux alternatifs et cachés.

Exemple 14 (incompatible)

```
<Amour> skos:altLabel "amour"@fr ; skos:hiddenLabel "amour"@fr .
```

Le graphe suivant n'est **pas compatible** avec le modèle de données SKOS, car il y a un clash entre les libellés lexicaux préférentiels et cachés.

Exemple 15 (incompatible)

```
<Amour> skos:prefLabel "amour"@fr ; skos:hiddenLabel "amour"@fr .
```

5.6. Notes

5.6.1. Domaine des Propriétés SKOS de Libellés Lexicaux

Notez que **aucun domaine n'est précisé** pour `skos:prefLabel`, `skos:altLabel` et `skos:hiddenLabel`. En pratique le domaine de ces propriétés est donc la classe de toutes les ressources (`rdfs:Resource`).

Par conséquent, l'utilisation des propriétés `skos:prefLabel`, `skos:altLabel` et `skos:hiddenLabel` pour **libeller n'importe quel type de ressource** est **compatible** avec le modèle de données SKOS.

Dans le graphe d'exemple ci-dessous, `skos:prefLabel`, `skos:altLabel` et `skos:hiddenLabel` ont été utilisées pour libeller une ressource de type `owl:Class` — ce qui est **compatible** avec le modèle de données SKOS.

Exemple 16 (compatible)

```
<MaClass> rdf:type owl:Class ;
  skos:prefLabel "animals"@en ;
  skos:altLabel "fauna"@en ;
  skos:hiddenLabel "aminals"@en ;
  skos:prefLabel "animaux"@fr ;
  skos:altLabel "faune"@fr .
```

5.6.2. Portée des Propriétés SKOS de Libellés Lexicaux

Notez que la portée de `skos:prefLabel`, `skos:altLabel` et `skos:hiddenLabel` est la classe des litéraux RDF simples [RDF-CONCEPTS].

Par convention, les litéraux RDF simples sont toujours utilisés comme objets d'un triplet, lorsque le prédicat est `skos:prefLabel`, `skos:altLabel` ou `skos:hiddenLabel`. Si un graphe ne suit **pas** cette convention une application est en droit de rejeter les données mais n'est pas obligée de le faire. Voir aussi la note ci-dessous.

5.6.3. Définir des Relations entre Libellés

Certaines applications ont besoin de fonctionnalités supplémentaires en ce qui concerne les libellés, par exemple pour permettre la description de ces libellés ou la définition de relations supplémentaires (telles que celles d'acronymie). Ceci est possible en identifiant les libellés avec des URIs. SKOS eXtension pour les Libellés définie dans l' Appendice A donne cette possibilité.

5.6.4. Libellés Alternatifs sans Libellés Préférentiel

Dans le graphe ci-dessous, une ressource a deux libellés lexicaux alternatifs, mais pas de libellé lexical préférentiel. Ceci est **compatible** avec le modèle de données SKOS, et il n'y a pas de conséquence particulière à cela à partir du modèle de données. Cependant, notez que de nombreuses applications vont requérir un libellé lexical préférentiel pour générer un affichage lisible optimal.

Exemple 17 (compatible)

```
<Amour> skos:altLabel "adoration"@fr , "désir"@fr .
```

5.6.5. Libellés et Etiquettes de Langue

[BCP47] définit des étiquettes pour identifier les langues. Notez que "en", "en-GB", "en-US" sont trois étiquettes de langue différentes, utilisées respectivement pour l'Anglais, l'Anglais Britannique et l'Anglais Américain. De la même façon, "ja", "ja-Hani", "ja-Hira", "ja-Kana" et "ja-Latn" sont cinq étiquettes de langue différentes utilisées respectivement pour le Japonais, le Japonais écrit en kanji, le script hiragana, le script katakana ou l'écriture en caractères latins (rōmaji).

Le graphe ci-dessous est **compatible** avec le modèle de données SKOS, car "en", "en-US" et "en-GB" sont des étiquettes de langue différentes.

Exemple 18 (compatible)

```
<Colour> skos:prefLabel "color"@en , "color"@en-US , "colour"@en-GB .
```

Dans le graphe ci-dessous, il n'y a pas d'incompatibilité entre les différentes propriétés de libellés lexicaux, là encore parce que "en" et "en-GB" sont des étiquettes de langue différentes, et par conséquent le graphe est **compatible** avec le modèle de données SKOS.

Exemple 19 (compatible)

```
<Love> skos:prefLabel "love"@en ; skos:altLabel "love"@en-GB .
```

Notez cependant que, comme indiqué dans la section 5.1, ces exemples servent uniquement à illustrer les possibilités générales du modèle de données SKOS, et n'indiquent **pas** nécessairement les bonnes pratiques pour la création des libellés avec des étiquettes de langue différentes. Les usages propres à une application ou à une langue donnée par rapport aux libellés et aux étiquettes de langue ne font pas partie du périmètre de la Référence SKOS.

Il est suggéré que les applications cherchent les libellés d'une langue donnée en recherchant dans les étiquettes de langues proches qui font partie du même schéma de concepts, c.à.d. en implémentant l'algorithme de recherche défini par le [BCP 47]. Les applications qui recherchent de cette façon n'ont donc pas besoins que les libellés soient présents dans toutes les variations possibles d'une langue (qui peuvent être nombreuses), et fonctionnent avec les schémas de concepts qui fournissent seulement les libellés dont la forme lexicale est différente pour une langue ou une collection de langues donnée.

---

6. Notations

6.1. Préambule

Une notation est une chaine de caractères telle que "T58.5" ou "303.4833" utilisée pour identifier un concept de manière unique dans le un schéma de concepts donné.

Une notation est différente d'un libellé lexical en ce qu'une notation n'est pas normalement reconnaissable comme un mot ou une suite de mots dans une langue naturelle.

Cette section définit la propriété `skos:notation`. Cette propriété est utilisée pour assigner une notation par un libellé typé [RDF-CONCEPTS].

6.2. Vocabulaire

```
skos:notation
```

6.3. Définitions des Classes & et des Propriétés

S15	`skos:notation` est une instance de `owl:DatatypeProperty`.

6.4. Exemples

L'exemple ci-dessous illustre une ressource `<http://example.com/ns/MonConcept>` avec une notation dont la forme lexicale est la chaine de caractères UNICODE "303.4833" et dont le type de données est indiqué par l'URI `<http://example.com/ns/MonDatatypeDeNotation>`.

Exemple 20 (compatible)

```
<MonConcept> skos:notation "303.4833"^^<MonDatatypeDeNotation> .
```

6.5. Notes

6.5.1. Notations, Litéraux Typés et Types de Donnée

Un litéral typé est une chaine de caractères UNICODE combinée avec une URI de type de données [RDF-CONCEPTS].

Les litéraux typés sont communément utilisés pour indiquer les valeurs telles que les entiers, les nombres à virgule flottante et les dates, et un certain nombre de types de données sont prédéfinis par la spécification XML Schema [XML-SCHEMA] tels que `xs:integer`, `xs:float` et `xs:date`.

Pour d'autres situations, de nouveaux types de données peuvent être définis, et sont alors appelés "types de données définis par l'utilisateur" [SWBP-DATATYPES].

Par convention, la propriété `skos:notation` est utilisée seulement avec un litéral typé comme objet du triplet, et l'URI du type de données indique un type de données défini par l'utilisateur correspondant à un système particulier de notation ou de codes de classification.

Dans de nombreuses situations il peut être suffisant de simplement créer une URI de type de données pour un système de notation particulier, et de définir le type de données de façon informelle dans un document qui décrit comment les notations sont construites et/ou quelles formes lexicales sont autorisées. Notez, cependant, qu'il est également possible de définir au moins le champ lexical d'un type de données plus formellement via le langage de Schémas XML, voir [SWBP-DATATYPES] section 2. Les utilisateurs doivent être conscients du fait que le niveau de support des types de données varie d'un outil à l'autre. Cependant, comme vu dans [OWL-REFERENCE] section 6.3, les outils devraient au moins traiter les litéraux identiques lexicalement comme égaux.

6.5.2. Notations Multiples

Il n'y a pas de contrainte sur la cardinalité de la propriété `skos:notation`. Un concept peut avoir zero, 1 ou plusieurs notations.

Lorsqu'on concept a plus d'une notation, celles-ci peuvent être issue du même système de notation ou de systèmes de notation différents. Dans le cas où les notations sont issues de systèmes de notation différents, des types de données différents peuvent être utilisés pour l'indiquer. Ce n'est pas dans les pratiques en usage d'assigner plus d'une notation d'un même système de notation (c.à.d. avec la même URI de type de données).

6.5.3. Notations Uniques dans les Schémas de Concepts

Par convention, deux concepts d'un même schéma de concepts ne doivent pas avoir une notation identique. Si c'était le cas, il ne serait pas possible d'utiliser la notation pour se référer de façon unique au concept (c.à.d. que la notation deviendrait ambigüe).

6.5.4. Notations et Libellés Préférentiels

Il n'y a pas de contraintes sur l'utilisation combinée de `skos:notation` et `skos:prefLabel`. Dans l'exemple ci-dessous, la même chaine est utilisée à la fois comme forme lexicale d'une notation et comme forme lexicale d'un libellé préférentiel.

Exemple 21 (compatible)

```
<Potassium>
  skos:prefLabel "K"@en ;
  skos:notation "K"^^<NotationPourSymbolesChimiques> .
```

Les litéraux typés consistent en une chaine de caractères et une URI de type de données. Par convention, `skos:notation` est seulement utilisée avec des **litéraux typés** comme objet du triplet.

Des litéraux simples consistent en une chaine de caractères et une étiquette de langue. Par convention, `skos:prefLabel` (et `skos:altLabel` et `skos:hiddenLabel`) sont seulement utilisés avec des **litéraux simples** comme objet du triplet.

Des litéraux RDF portant à la fois une étiquette de langue et une URI de type de données n'existent pas; un litéral typé n'a pas d'étiquette de langue, un litéral simple n'a pas d'URI de type de données.

6.5.5. Domaine de skos:notation

Notez qu' **aucun domaine n'est déclaré** pour `skos:notation`. En pratique son domaine est la classe de toutes les ressources (`rdfs:Resource`). Par conséquent, l'utilisation de `skos:notation` sur n'importe quel type de ressource est compatible avec le modèle de données SKOS.

---

7. Propriétés de Documentation (Propriétés de Note)

7.1. Préambule

Les notes sont utilisées pour donner des informations sur les concepts SKOS. Il n'y a pas de restriction sur la nature de ces informations, c.à.d. que celles-ci pourraient être sous forme de texte, de liens hypertextes, ou d'images; cela pourrait être une définition, une information sur le champ d'application d'un concept, une information éditoriale, ou tout autre type d'information.

Il y a sept propriétés en SKOS pour associer des notes à des concepts, définies formellement dans cette section. Pour plus d'informations sur les utilisations recommandées de chaque propriété de document SKOS, voir le [SKOS-PRIMER].

Ces sept propriétés ne sont pas faites pour répondre à toutes les situations, mais pour couvrir les situations les plus courantes, et fournir des points d'extension pour définir des types de notes plus spécifiques. Pour plus d'informations sur les bonnes pratiques recommandées pour étendre SKOS, voir le [SKOS-PRIMER].

Trois scenarios d'usage différents des propriétés de documentation SKOS sont recommandés dans le [SKOS-PRIMER] — "documentation comme un litéral RDF", "documentation comme une ressource associée" et "documentation comme un document de référence". Le modèle de données défini dans cette section est fait pour couvrir ces trois scenarios.

7.2. Vocabulaire

`skos:note`
`skos:changeNote`
`skos:definition`
`skos:editorialNote`
`skos:example`
`skos:historyNote`
`skos:scopeNote`

7.3. Définitions des Classes & et des Propriétés

S16	`skos:note`, `skos:changeNote`, `skos:definition`, `skos:editorialNote`, `skos:example`, `skos:historyNote` et `skos:scopeNote` cont chacune instances de `owl:AnnotationProperty`.
S17	`skos:changeNote`, `skos:definition`, `skos:editorialNote`, `skos:example`, `skos:historyNote` et `skos:scopeNote` sont chacune sous-propriétés de `skos:note`.

7.4. Exemples

Le graphe ci-dessous donne un exemple du scenario "documentation comme un litéral RDF".

Exemple 22 (compatible)

```
<MaResource> skos:note "ceci est une note"@fr .
```

Le graphe ci-dessous donne un exemple du scenario "documentation comme un document de référence".

Exemple 23 (compatible)

```
<MaResource> skos:note <MaNote> .
```

7.5. Notes

7.5.1. Domaine des Propriétés SKOS de Documentation

Notez qu' **aucun domaine n'est spécifié** pour les propriétés de document SKOS. Leur domaine en pratique est donc la classe de toutes les ressources (`rdfs:Resource`). Par conséquent, l'utilisation des propriétés de documentation SKOS pour indiquer de l'information sur **n'importe quel type de ressource** est compatible avec le modèle de données SKOS.

Dans le graphe d'exemple ci-dessous, `skos:definition` est utilisée pour donner une définition sous forme de texte d'une ressource de type `owl:Class` — ce qui est compatible avec le modèle de données SKOS.

Exemple 24 (compatible)

```
<Protein> rdf:type owl:Class ;
  skos:definition """A physical entity consisting of a sequence of amino-acids; a protein monomer; a single polypeptide chain. An example is the EGFR
protein."""@en .
```

7.5.2. Portée des Propriétés SKOS de Documentation

Notez qu'aucune portée n'est spécifiée pour les propriétés SKOS de documentation, et leur portée en pratique est donc la classe de toutes les ressources (`rdfs:Resource`). Sous les régimes de sémantique RDF et OWL Full, tout est une ressource, y compris les litéraux RDF simples.

---

8. Relations Sémantiques

8.1. Préambule

Les relations sémantiques SKOS sont des liens entre concepts SKOS, inhérents au sens des concepts reliés.

Le Système Simple d'Organisation de Connaissances distingue deux catégories de relations sémantiques : **hiérarchiques** et **associatives**. Un lien hiérarchique entre deux concepts indique que l'un est d'une certaine façon plus général ("broader" en anglais) que l'autre ("narrower" en anglais). Un lien associatif entre deux concepts indique que les deux sont de façon inhérente "reliés", mais qu'**aucun** n'est en quelque façon que ce soit plus général que l'autre.

Les propriétés `skos:broader` et `skos:narrower` sont utilisées pour exprimer un lien hiérarchique direct entre deux concepts SKOS. Un triplet `<A> skos:broader <B>` exprime le fait que `<B>`, l'objet du triplet, est un concept plus générique que `<A>`, le sujet du triplet. De façon similaire, le triplet `<C> skos:narrower <D>` exprime le fait que `<D>`, l'objet du triplet, est un concept plus spécifique que C `<C>`, le sujet du triplet.

Par convention, `skos:broader` et `skos:narrower` sont **seulement** utilisées pour exprimer un lien hiérarchique **direct** (c'est-à-dire immédiat) entre deux concepts SKOS. Cela permet aux applications d'accéder de façon simple et fiable, depuis n'importe quel concept, aux concepts liés par des liens "plus générique" ou "plus spécifique". Notez que, pour respecter cette convention, les propriétés `skos:broader` et `skos:narrower` ne sont **pas** déclarées comme des propriétés transitives.

Certaines applications ont besoin d'utiliser **à la fois des liens hiérarchiques directs et indirects** entre concepts, par exemple pour améliorer le rappel d'une application de recherche à l'aide d'extensions de requêtes. Pour ce besoin, les propriétés `skos:broaderTransitive` et `skos:narrowerTransitive` sont fournies. Un triplet `<A> skos:broaderTransitive <B>` représente un lien hiérarchique direct ou indirect, où `<B>` est un "ancêtre" plus générique de `<A>`. De façon similaire un triplet `<C> skos:narrowerTransitive <D>` représente un lien hiérarchique direct ou indirect, où `<D>` est un "descendant" plus spécifique de `<C>`.

Par convention, les propriétés `skos:broaderTransitive` et `skos:narrowerTransitive` ne sont **pas** utilisées directement pour faire des assertions. Ces propriétés sont plutôt utilisées pour inférer la fermeture transitive des liens hiérarchiques, qui peut ensuite être utilisée pour accéder aux liens hiérarchiques directs ou indirects entre concepts.

La propriété `skos:related` est utilisée pour exprimer un lien associatif entre deux concepts SKOS.

Pour des exemples supplémentaires sur les façons d'exprimer des liens hiérarchiques et associatifs, reportez-vous au [SKOS-PRIMER].

8.2. Vocabulaire

`skos:semanticRelation`
`skos:broader`
`skos:narrower`
`skos:related`
`skos:broaderTransitive`
`skos:narrowerTransitive`

8.3. Définitions des Classes & Propriétés

S18	`skos:semanticRelation`, `skos:broader`, `skos:narrower`, `skos:related`, `skos:broaderTransitive` et `skos:narrowerTransitive` sont chacune instances de `owl:ObjectProperty`.
S19	Le `rdfs:domain` de `skos:semanticRelation` est la classe `skos:Concept`.
S20	Le `rdfs:range` de `skos:semanticRelation` est la classe `skos:Concept`.
S21	`skos:broaderTransitive`, `skos:narrowerTransitive` et `skos:related` sont chacune sous-propriétés de `skos:semanticRelation`.
S22	`skos:broader` est une sous-propriété de `skos:broaderTransitive`, et `skos:narrower` est une sous-propriété de `skos:narrowerTransitive`.
S23	`skos:related` est une instance de `owl:SymmetricProperty`.
S24	`skos:broaderTransitive` et `skos:narrowerTransitive` sont chacune instances de `owl:TransitiveProperty`.
S25	`skos:narrower` est `owl:inverseOf` la propriété `skos:broader`.
S26	`skos:narrowerTransitive` ets `owl:inverseOf` la propriété `skos:broaderTransitive`.

8.4. Conditions d'Intégrité

S27	`skos:related` est disjoint avec la propriété `skos:broaderTransitive`.

Notez que par le fait que `skos:related` soit une propriété symétrique, et `skos:broaderTransitive` et `skos:narrowerTransitive` soient inverses l'une de l'autre, `skos:related` est également disjoint avec `skos:narrowerTransitive`.

8.5. Exemples

Le graphe ci-dessous exprimer un lien hiérarchique direct entre `<A>` et `<B>` (où `<B>` est plus générique que `<A>`), et un lien associatif entre `<A>` et `<C>`, et est **compatible** avec le modèle de données SKOS.

Exemple 25 (compatible)

```
<A> skos:broader <B> ; skos:related <C> .
```

Le graphe ci-dessous est **incompatible** par rapport au modèle de données SKOS, car il y a une incompatibilité entre les liens associatifs et hiérarchiques.

Exemple 26 (incompatible)

```
<A> skos:broader <B> ; skos:related <B> .
```

Le graphe ci-dessous est **incompatible** par rapport au modèle de données SKOS, car de la même façon il y a une incompatibilité entre les liens associatifs et hiérarchiques.

Exemple 27 (incompatible)

```
<A> skos:broader <B> ; skos:related <C> .
<B> skos:broader <C> .
```

Dans l'exemple ci-dessus, l'incompatibilité n'est pas immédiatement évidente. Elle devient apparente lorsqu'on calcule les inférences basées sur les définitions de classes et de propriétés données plus haut, ce qui donne le graphe suivant.

Exemple 28 (incompatible)

```
<A> skos:broaderTransitive <C> ; skos:related <C> .
```

Le graphe ci-dessous est **incompatible** par rapport au modèle de données SKOS, car encore une fois il y a une incompatibilité entre les liens associatifs et hiérarchiques, qui peut être déduite des définitions de classes et de propriétés données plus haut.

Exemple 29 (incompatible)

```
<A> skos:narrower <B> ; skos:related <C> .
<B> skos:narrower <C> .
```

8.6. Notes

8.6.1. Relations de Sous-Propriété

Le diagramme ci-dessous illustre informellement les relations de sous-propriété entre les propriétés de relation sémantique SKOS.

```
skos:semanticRelation
 |
 +- skos:related
 |
 +- skos:broaderTransitive
 |    |
 |    +- skos:broader
 |
 +- skos:narrowerTransitive
      |
      +- skos:narrower
```

8.6.2. Domaine et Portée des Propriétés de Relation Sémantique SKOS

Notez que le domaine et l'ensemble d'arrivée de `skos:semanticRelation` est la classe `skos:Concept`. A cause du fait que `skos:broader`, `skos:narrower` et `skos:related` sont chacune sous-propriétés de `skos:semanticRelation`, le graphe dans l'exemple 26 ci-dessus implique donc que `<A>`, `<B>` et `<C>` sont chacun instance de `skos:Concept`.

8.6.3. Symétrie de skos:related

`skos:related` est une propriété symétrique. L'exemple ci-dessous illustre le résultat d'un raisonnement qui découle de cette condition.

Exemple 30 (implication)

```
<A> skos:related <B> .
```

implique

```
<B> skos:related <A> .
```

Notez que, bien que `skos:related` soit une propriété symétrique, cette condition n'entraine **pas** de restrictions sur les sous-propriétés de `skos:related` (c'est-à-dire que les sous-propriétés de `skos:related` peuvent être symétriques, non symétriques ou antisymétriques et rester conforme au modèle de données SKOS).

Pour illustrer ce point, dans l'exemple ci-dessous, deux nouvelles propriétés qui ne sont **pas** symétriques sont déclarées comme sous-propriétés de `skos:related`. Cet exemple, qui est **compatible** avec le modèle de données SKOS, montre également le résultat de quelques implications.

Exemple 31 (implication)

```
<cause> rdf:type owl:ObjectProperty ;
  rdfs:subPropertyOf skos:related .

<effet> rdf:type owl:ObjectProperty ;
 rdfs:subPropertyOf skos:related ;
  owl:inverseOf <cause> .

<A> <cause> <B> .
```

implique

```
<A> skos:related <B> .

<B> <effet> <A> ; skos:related <A> .
```

Voir aussi le [SKOS-PRIMER] pour des recommendations et des bonnes pratiques pour étendre SKOS.

8.6.4. skos:related et la Transitivité

Notez que `skos:related` n'est **pas** une propriété transitive. Par conséquent, le modèle de données SKOS ne permet **pas** le raisonnement illustré dans l'exemple ci-dessous.

Exemple 32 (non-implication)

```
<A> skos:related <B> .
<B> skos:related <C> .
```

n'implique pas

```
<A> skos:related <C> .
```

8.6.5. skos:related et la Réflexivité

Notez que cette spécification ne déclare **pas** `skos:related` comme une propriété réflexive, et ne déclare pas **non plus** `skos:related` comme une propriété irréflexive.

Comme `skos:related` n'est **pas** définie comme une propriété irréflexive, le graphe ci-dessous est **compatible** avec le modèle de données SKOS.

Exemple 33 (compatible)

```
<A> skos:related <A> .
```

Cependant, pour beaucoup d'applications qui utilisent des systèmes d'organisation de connaissances, les assertions de la forme X `skos:related` X peuvent poser un problème. Si c'est le cas, l'application peut rechercher ces assertions avant de commencer à traiter les données SKOS, même si la façon pour l'application de traiter ces assertions n'est pas définie dans cette spécification et peut varier d'une application à l'autre.

8.6.6. skos:broader et la Transitivité

Notez que `skos:broader` n'est **pas** une propriété transitive. De la même façon, `skos:narrower` n'est **pas** une propriété transitive. Par conséquent, le modèle de données SKOS ne permet **pas** le raisonnement illustré dans l'exemple ci-dessous.

Exemple 34 (non-implication)

```
<A> skos:broader <B> .
<B> skos:broader <C> .
```

n'implique pas

```
<A> skos:broader <C> .
```

Cependant, `skos:broader` est une sous-propriété de `skos:broaderTransitive`, qui **est** une propriété transitive. De la même façon, `skos:narrower` est une sous-propriété de `skos:narrowerTransitive`, qui **est** une propriété transitive. Par conséquent le modèle de données SKOS **permet** le raisonnement illustré ci-dessous.

Exemple 35 (implication)

```
<A> skos:broader <B> .
<B> skos:broader <C> .
```

implique

```
<A> skos:broaderTransitive <B> .
<B> skos:broaderTransitive <C> .
<A> skos:broaderTransitive <C> .
```

Remarquez plus particulièrement que, par convention, `skos:broader` et `skos:narrower` sont **seulement** utilisés pour exprimer des liens hiérarchiques immédiats (c'est-à-dire directs) entre deux concepts SKOS. Par convention, `skos:broaderTransitive` et `skos:narrowerTransitive` ne sont **pas** utilisées directement pour exprimer les liens, mais sont utilisées seulement dans les conclusions des inférences.

Cette façon de faire permet aux liens hiérarchiques directs (c'est-à-dire immédiats) d'être conservés, ce qui est nécessaire dans beaucoup de situations (par exemple construire des représentations graphiques des systèmes d'organisation de connaissances) , tout en offrant un mécanisme pour interroger la fermeture transitive de ces liens hiérarchiques (qui incluera les liens directs et indirects), ce qui est utile dans d'autres situations (par exemple les algorithmes d'extension de requête).

Notez également qu'une sous-propriété d'une propriété transitive n'est **pas** nécessairement transitive.

Voir également la note sur les chemins alternatifs ci-dessous.

8.6.7. skos:broader et la Réflexivité

Notez que cette spécification ne dit rien de particulier concernant la réflexivité de la relation `skos:broader`. Elle ne déclare **pas** `skos:broader` comme une propriété réflexive, et ne déclare pas **non plus** `skos:broader` comme une propriété irréflexive. Par conséquent pour tout graphe et toute ressource `<A>`, le triplet:

Exemple 36 (compatible)

```
<A> skos:broader <A> .
```

peut ou pas être présent. Cette posture prudente permet d'utiliser SKOS pour modéliser des Systèmes d'Organisation de Connaissances dans lesquels l'interprétation de `skos:broader` est réflexive (par exemple une traduction directe d'une hiérarchie de classes inférée en OWL), ou d'autres pour lesquels `skos:broader` sera considéré comme irréflexive (ce qui est approprié pour la plupart des thesauri et des classifications).

De la même façon, rien n'est définit sur la réflexivité ou l'irréflexivité de `skos:narrower`.

Cependant, pour beaucoup d'applications qui utilisent des systèmes d'organisation de connaissances, les assertions de la forme X `skos:broader` X ou Y `skos:narrower` Y peuvent poser problème. Si c'est le cas, l'application peut rechercher ces assertions avant de commencer à traiter les données SKOS, même si la façon pour l'application de traiter ces assertions n'est pas définie dans cette spécification et peut varier d'une application à l'autre.

8.6.8. Cycles dans les Relations Hiérarchiques (skos:broaderTransitive et la Réflexivité)

Dans le graphe ci-dessous, un cycle a été déclaré dans la relation hiérarchique. Notez que ce graphe est **compatible** avec le modèle de données SKOS, c'est-à-dire qu'il n'y a **aucune** règle qui vérifie que `skos:broaderTransitive` doit être irréflexive.

Exemple 37 (compatible)

```
<A> skos:broader <B> .
<B> skos:broader <A> .
```

Cependant, pour beaucoup d'applications dans lesquelles les systèmes d'organisation de connaissances sont utilisés, un cycle dans une relation hiérarchique peut poser problème. Si c'est le cas, une stratégie commode pour trouver les cycles dans une relation hiérarchique peut consister à calculer la fermeture transitive de `skos:broaderTransitive` et ensuite à trouver les assertions de la forme `X skos:broaderTransitive X`. La façon pour l'application de traiter ensuite ces assertions n'est pas définie dans cette spécification et peut varier d'une application à l'autre.

8.6.9. Chemins Hiérarchiques Multiples

Dans le graphe ci-dessous, il y a deux chemins possibles dans la relation hiérarchique pour aller de A à C.

Exemple 38 (compatible)

```
<A> skos:broader <B> , <C> .
<B> skos:broader <C> .
```

Dans le graphe ci-dessous, il y a deux chemins possibles dans la relation hiérarchique pour aller de A à D.

Exemple 39 (compatible)

```
<A> skos:broader <B> , <C> .
<B> skos:broader <D> .
<C> skos:broader <D> .
```

C'est une forme de graphe qui arrive de façon naturelle dans les systèmes d'organisation de connaissances poly-hiérarchiques.

Ces deux graphes sont **compatibles** avec le modèle de données SKOS, c'est-à-dire qu'il n'y a **aucune** règle qui vérifie qu'il n'y a qu'un seul chemin hiérarchique entre deux noeuds.

8.6.10. skos:related et skos:broaderTransitive sont Disjointes

Cette spécification traite les relations hiérarchiques et associatives comme fondamentalement disjointes par nature. C'est pourquoi la co-existence de liens hiérarchiques et associatifs entre les mêmes concepts n'est **pas** compatible par rapport au modèle de données SKOS. Les exemples ci-dessus illustrent un certain nombres de situations dans lesquelles cette incompatibilité se produit.

Cette position suit les définitions classiques des liens hiérarchiques et associatifs dans les standards sur les thesaurus [ISO2788] et [BS8723-2], et reflète les pratiques courantes dans beaucoup de systèmes d'organisation de connaissances.

Notez que cette spécification prend la position plus ferme que, non seulement les liens hiérarchiques immédiats (c'est-à-dire directs) et associatifs sont disjoints, mais que les liens associatifs sont également disjoints avec les liens hiérarchiques *indirects*. Ceci est capturé formellement dans la règle d'intégrité qui spécifie que `skos:related` et `skos:broaderTransitive` sont des propriétés disjointes.

---

9. Collections de Concepts

9.1. Préambule

Les collections de concepts SKOS sont des groupes nommés et/ou ordonnés de concepts SKOS.

Les collections sont utiles dans les cas où plusieurs concepts partagent un aspect commun et qu'il devient pratique de les rassembler dans un même groupe, ou bien dans les cas où certains concepts doivent être placés dans un ordre précis.

9.2. Vocabulaire

`skos:Collection`
`skos:OrderedCollection`
`skos:member`
`skos:memberList`

9.3. Définition des Classes & Propriétés

S28	`skos:Collection` et `skos:OrderedCollection` sont chacune instance de `owl:Class`.
S29	`skos:OrderedCollection` est une sous-classe de `skos:Collection`.
S30	`skos:member` et `skos:memberList` sont chacune instance de `owl:ObjectProperty`.
S31	Le `rdfs:domain` de `skos:member` est la classe `skos:Collection`.
S32	Le `rdfs:range` de `skos:member` est l'union des classes `skos:Concept` et `skos:Collection`.
S33	Le `rdfs:domain` de `skos:memberList` est la classe `skos:OrderedCollection`.
S34	Le `rdfs:range` de `skos:memberList` est la classe `rdf:List`.
S35	`skos:memberList` est une instance de `owl:FunctionalProperty`.
S36	Quelle que soit la ressource, tout élément de la liste utilisée comme valeur de la propriété `skos:memberList` est également valeur de la propriété `skos:member`.

9.4. Conditions d'Intégrité

S37	`skos:Collection` est disjointe avec `skos:Concept` et disjointe avec `skos:ConceptScheme`.

9.5. Exemples

Le graphe ci-dessous illustre une collection SKOS comportant 3 membres.

Exemple 40 (compatible)

```
<MaCollection> rdf:type skos:Collection ;
  skos:member <X> , <Y> , <Z> .
```

Le graphe ci-dessous illustre une collection ordonnée SKOS comportant 3 membres. Notez l'utilisation de la syntaxe Turtle `(...)`, représentant une Collection RDF (une liste).

Exemple 41 (compatible)

```
<MyOrderedCollection> rdf:type skos:OrderedCollection ;
  skos:memberList ( <X> <Y> <Z> ) .
```

9.6. Notes

9.6.1. Déduire des Collections à Partir de Collections Ordonnées

La règle S36 pose une relation logique entre les propriétés `skos:memberList` et `skos:member`. Cette relation veut dire qu'une collection simple peut être déduite d'une collection ordonnée. Ceci est illustré dans l'exemple ci-dessous.

Exemple 42 (implication)

```
<MyOrderedCollection> rdf:type skos:OrderedCollection ;
  skos:memberList ( <X> <Y> <Z> ) .
```

implique

```
<MyOrderedCollection> rdf:type skos:Collection ;
  skos:member <X> , <Y> , <Z> .
```

Notez que SKOS ne propose aucune façon d'exprimer explicitement qu'une collection n'est **pas** ordonnée.

9.6.2. Intégrité de skos:memberList

Notez que `skos:memberList` est une propriété fonctionnelle, c'est-à-dire qu'elle n'a pas plus d'une valeur. Le but est de capturer dans le modèle de données SKOS le fait qu'il n'y a aucune raison qu'une collection ordonnée ait plus d'une liste de membres. Malheureusement, il n'y a pas la possibilité d'utiliser cette condition comme condition d'intégrité sans exprimer le fait que deux listes sont des objets différents. En d'autres termes, bien que le graphe ci-dessous soit **compatible** avec le modèle de données SKOS, il entraîne une absurdité (une liste avec deux éléments de tête et une "queue fourchue", c'est-à-dire deux éléments de fin).

Exemple 43 (implication)

```
<OrderedCollectionResource>
  skos:memberList ( <A> <B> ) , ( <X> <Y> ) .
```

implique

```
<OrderedCollectionResource>
  skos:memberList [ rdf:first <A> , <X> ; rdf:rest [ rdf:first <B> ; rdf:rest rdf:nil ] , [ rdf:first <Y> ; rdf:rest rdf:nil ] ] .
```

Cependant, comme indiqué dans [RDF-SEMANTICS] section 3.3.3, des extensions sémantiques à RDF peuvent ajouter des restrictions de conformité supplémentaires sur l'utilisation du vocabulaire RDF des collections (`rdf:first`, `rdf:rest`, `rdf:nil`) de façon à éviter ce genre de graphes.

9.6.3. Collections Imbriquées

Dans l'exemple ci-dessous, une collection est imbriquées dans une autre collection.

Exemple 44 (compatible)

```
<MyCollection> rdf:type skos:Collection ;
  skos:member <A> , <B> , <MyNestedCollection> .

<MyNestedCollection> rdf:type skos:Collection ;
  skos:member <X> , <Y> , <Z> .
```

Cet exemple est **compatible** avec le modèle de données SKOS, car la portée de `skos:member` est l'union de `skos:Concept` et `skos:Collection`.

9.6.4. Concepts SKOS, Collections de Concepts et Relations Sémantiques

Dans le modèle de données SKOS, `skos:Concept` et `skos:Collection` sont des classes disjointes. Par ailleurs le domaine et la portée des propriétés de relation sémantique SKOS est `skos:Concept`. Par conséquent, si n'importe laquelle des relations sémantiques SKOS (par exemple `skos:narrower`) est utilisée sur une collection ou pour référencer une collection, le graphe ne sera **pas** compatible avec le modèle de données SKOS.

Ceci est illustré dans l'exemple ci-dessous, qui n'est **pas** compatible avec le modèle de données SKOS.

Exemple 45 (incompatible)

```
<A> skos:narrower <B> .
<B> rdf:type skos:Collection .
```

De la même façon, le graphe ci-dessous n'est **pas** compatible avec le modèle de données SKOS.

Exemple 46 (incompatible)

```
<A> skos:broader <B> .
<B> rdf:type skos:Collection .
```

De la même façon, le graphe ci-dessous n'est **pas** compatible avec le modèle de données SKOS.

Exemple 47 (incompatible)

```
<A> skos:related <B> .
<B> rdf:type skos:Collection .
```

Cependant, le graphe ci-dessous est compatible avec le modèle de données SKOS.

Exemple 48 (compatible)

```
<A> skos:narrower <B> , <C> , <D> .

<ResourceCollection> rdfs:label "Collection de Ressources"@fr ;
  skos:member <B> , <C> , <D> .
```

Cela signifie que, pour des thesauri et d'autres systèmes d'organisation de connaissances où les libellés des entrées sont utilisés dans un affichage systématique de ce thesaurus, la représentation conforme en SKOS demande une analyse attentive. Par ailleurs, lorsque les libellés des entrées sont utilisés dans l'affichage systématique, il n'est pas toujours possible de reconstruire entièrement cet affichage systématique à partir de la seule représentation SKOS. Modéliser de façon exhaustive les informations nécessaires à l'affichage systématique d'un thesaurus ou d'un autre système d'organisation de connaissances, y compris les détails de mise en page et de présentation, va au-delà du périmètre de SKOS.

10. Propriétés d'alignement

10.1. Préambule

Les propriétés définissant les alignements en SKOS sont `skos:closeMatch`, `skos:exactMatch`, `skos:broadMatch`, `skos:narrowMatch` et `skos:relatedMatch`. Ces propriétés sont utilisées pour déclarer des liens de correspondance (ou d'alignement) entre des concepts SKOS appartenant à des schémas de concepts différents, dans le cas où ces liens peuvent se déduire du sens des concepts ainsi reliés.

Les propriétés `skos:broadMatch` et `skos:narrowMatch` sont utilisés pour indiquer un lien d'alignement hiérarchique entre deux concepts.

La propriété `skos:relatedMatch` est utilisée pour indiquer un lien d'alignement associatif entre deux concepts.

La propriété `skos:closeMatch` est utilisée pour relier deux concepts qui sont suffisamment similaires pour être utilisé de façon interchangeable dans **certaines** applications de recherche d'information. Afin d'éviter les "erreurs d'approximation" qui pourraient survenir en suivant les alignements à travers plusieurs schémas de concepts, `skos:closeMatch` n'est **pas** déclarée comme une propriété transitive.

La propriété skos:exactMatch est utilisée pour relier deux concepts, indiquant par là un fort degré de confiance que les concepts puissent être utilisés de façon interchangeable dans une large majorité d'applications de recherche d'informations. skos:exactMatch est une propriété transitive, et est une sous-propriété de skos:closeMatch.

10.2. Vocabulaire

skos:mappingRelation
skos:closeMatch
skos:exactMatch
skos:broadMatch
skos:narrowMatch
skos:relatedMatch

10.3. Définitions des Classes & Propriétés

S38	skos:mappingRelation, skos:closeMatch, skos:exactMatch, skos:broadMatch, skos:narrowMatch et skos:relatedMatch sont chacune instances de owl:ObjectProperty.
S39	skos:mappingRelation est une sous-propriété de skos:semanticRelation.
S40	skos:closeMatch, skos:broadMatch, skos:narrowMatch et skos:relatedMatch sont chacune sous-propriétés de skos:mappingRelation.
S41	skos:broadMatch est une sous-propriété de skos:broader, skos:narrowMatch est une sous-propriété de skos:narrower, et skos:relatedMatch est une sous-propriété de skos:related.
S42	skos:exactMatch est une sous-propriété de skos:closeMatch.
S43	skos:narrowMatch est owl:inverseOf de la propriété skos:broadMatch.
S44	skos:relatedMatch, skos:closeMatch et skos:exactMatch sont chacune instance de owl:SymmetricProperty.
S45	skos:exactMatch est une instance de owl:TransitiveProperty.

10.4. Conditions d'Intégrité

S46	skos:exactMatch est disjointe avec chacune des propriétés skos:broadMatch et skos:relatedMatch.

Notez que par le fait que skos:exactMatch soit une propriété symétrique et que skos:broadMatch et skos:narrowMatch soient inverses l'une de l'autre, skos:exactMatch est également disjointe avec skos:narrowMatch.

10.5. Exemples

Le graphe ci-dessous exprime un lien de correspondance exacte entre <A> et <B>.

Exemple 49 (compatible)

```
<A> skos:exactMatch <B> .
```

Le graphe ci-dessous exprime un lien de correspondance proche entre <A> et <B>.

Exemple 50 (compatible)

```
<A> skos:closeMatch <B> .
```

Le graphe ci-dessous exprime un lien de correspondance hiérarchique entre <A> et <B> (où <B> est plus générique que <A>), et un lien de correspondance associative entre <A> et <C>.

Exemple 51 (compatible)

```
<A> skos:broadMatch <B> ; skos:relatedMatch <C> .
```

Le graphe ci-dessous n'est **pas compatible** avec le modèle de données SKOS, car il y a une incompatibilité entre les liens de correspondance exacts et hiérarchiques.

Exemple 52 (incompatible)

```
<A> skos:exactMatch <B> ; skos:broadMatch <B> .
```

Le graphe ci-dessous n'est **pas compatible** avec le modèle de données SKOS, car il y a une incompatibilité entre les liens de correspondance exacts et associatifs.

Exemple 53 (incompatible)

```
<A> skos:exactMatch <B> ; skos:relatedMatch <B> .
```

10.6. Notes

10.6.1. Propriétes d'Alignement, Propriétés de Relation Sémantique et Schémas de Concepts

Par convention, les propriétés SKOS d'alignement sont utilisées seulement pour relier des concepts appartenant à des schémas de concepts **différents**. Cependant, notez que l'utilisation des propriétés SKOS de relation sémantique (skos:broader, skos:narrower, skos:related) pour relier des concepts dans des schémas de concepts **différents** est également **compatible** avec le modèle de données SKOS (voir la Section 8).

Les propriétés d'alignement skos:broadMatch, skos:narrowMatch et skos:relatedMatch sont proposées pour faciliter les situations dans lesquelles la provenance des données est connue, et où il est utile de pouvoir distinguer directement les liens internes à un schéma de concepts des liens d'alignement externes entre schémas de concepts différents.

Cependant, l'utilisation des propriétés d'alignement SKOS **ne se substitue pas** à la gestion rigoureuse des graphes RDF ou à l'utilisation des mécanismes d'indication de provenance des données.

La raison de ces choix est qu'il est difficile de faire une distinction nette entre des liens internes à un schéma de concepts et des liens d'alignement entre schémas de concepts. Ceci est particulièrement vrai dans un environnement ouvert où différentes personnes peuvent tout à fait réorganiser des concepts dans des schémas de concepts différents. Ce que l'un considère comme deux schémas de concepts avec des liens d'alignement entre les deux, un autre peut le voir comme un schéma de concepts unique avec des liens internes seulement. Cette spécification permet aux deux points de vue de coexister, ce qui promouvra (c'est à espérer) la flexibilité et l'innovation dans la réutilisation des données SKOS sur le web.

Il y a donc une proximité très forte entre les propriétés SKOS de relations sémantiques et celles d'alignement. La propriété skos:broadMatch est une sous-propriété de skos:broader, skos:narrowMatch est une sous-propriété de skos:narrower, et skos:relatedMatch est une sous-propriété de skos:related. L'ensemble complet de ces liens de sous-propriété est

illustré ci-dessous.

```
skos:semanticRelation
 |
 +- skos:related
 |   |
 |   +- skos:relatedMatch
 |
 +- skos:broaderTransitive
 |   |
 |   +- skos:broader
 |       |
 |       +- skos:broadMatch
 |
 +- skos:narrowerTransitive
 |   |
 |   +- skos:narrower
 |       |
 |       +- skos:narrowMatch
 |
 +- skos:mappingRelation
     |
     +- skos:closeMatch
     |   |
     |   +- skos:exactMatch
     |
     +- skos:relatedMatch
     |
     +- skos:broadMatch
     |
     +- skos:narrowMatch
```

Les exemples ci-dessous illustrent quelques implications de cet arbre de sous-propriétés, ainsi que du domaine et de la portée de `skos:semanticRelation`.

Exemple 54 implication

```
<A> skos:broadMatch <B> .
```

implique

```
<A> skos:mappingRelation <B> .
<A> skos:broader <B> .
<A> skos:broaderTransitive <B> .
<A> skos:semanticRelation <B> .
<A> rdf:type skos:Concept .
<B> rdf:type skos:Concept .
```

Exemple 55 implication

```
<A> skos:narrowMatch <B> .
```

implique

```
<A> skos:mappingRelation <B> .
<A> skos:narrower <B> .
<A> skos:narrowerTransitive <B> .
<A> skos:semanticRelation <B> .
<A> rdf:type skos:Concept .
<B> rdf:type skos:Concept .
```

Exemple 56 implication

```
<A> skos:relatedMatch <B> .
```

implique

```
<A> skos:mappingRelation <B> .
<A> skos:related <B> .
<A> skos:semanticRelation <B> .
<A> rdf:type skos:Concept .
<B> rdf:type skos:Concept .
```

Exemple 57 implication

```
<A> skos:exactMatch <B> .
```

implique

```
<A> skos:closeMatch <B> .
<A> skos:mappingRelation <B> .
<A> skos:semanticRelation <B> .
<A> rdf:type skos:Concept .
<B> rdf:type skos:Concept .
```

Notez également que, du fait que différentes personnes ont la possibilité de réorganiser des concepts dans des schémas de concepts différents, un graphe peut exprimer des liens d' **alignement** entre concepts du **même** schéma de concepts, et il n'y a **pas** de règle d'intégrité formelle dans le modèle de données SKOS qui rendrait ce graphe incompatible; par exemple, le graphe ci-dessous est **compatible** par rapport au modèle de données SKOS. Cependant, on s'attend à ce qu'un tel graphe ne provienne que de la fusion de deux graphes ou plus provenant de sources différentes.

Exemple 58 (compatible)

```
<A> skos:broadMatch <B> ; skos:relatedMatch <C> .

<A> skos:inScheme <MyScheme> .
<B> skos:inScheme <MyScheme> .
<C> skos:inScheme <MyScheme> .
```

10.6.2. Incompatibilité Entre les Liens Hiérarchiques et Associatifs

Les exemples ci-dessous illustrent des "clashs" entre des liens d'alignement hiérarchiques et associatifs, qui sont **incompatibles** par rapport au modèle de données SKOS (à cause du lien de sous-propriété illustré ci-dessus, et à cause du modèle de données SKOS pour les relations sémantiques défini dans la Section 8).

Exemple 59 (incompatible)

```
<A> skos:broadMatch <B> ; skos:relatedMatch <B> .
```

Exemple 60 (incompatible)

```
<A> skos:narrowMatch <B> ; skos:relatedMatch <B> .
```

Exemple 61 (incompatible)

```
<A> skos:broadMatch <B> .
<B> skos:broadMatch <C> .
<A> skos:relatedMatch <C> .
```

10.6.3. Propriétés d'Alignement et Transitivité

La seule propriété d'alignement SKOS déclarée comme transitive est `skos:exactMatch`. Un exemple de raisonnement est illustré ci-dessous :

Exemple 62 implication

```
<A> skos:exactMatch <B> .
<B> skos:exactMatch <C> .
```

implique

```
<A> skos:exactMatch <C> .
```

Aucune autre propriété d'alignement SKOS n'est transitive. Par conséquent, les raisonnements illustrés dans les exemples ci-dessous ne sont **pas** supportés par le modèle de données SKOS.

Exemple 63 (non-implication)

```
<A> skos:broadMatch <B> .
<B> skos:broadMatch <C> .
```

n'implique pas

```
<A> skos:broadMatch <C> .
```

Exemple 64 (non-implication)

```
<A> skos:relatedMatch <B> .
<B> skos:relatedMatch <C> .
```

n'implique pas

```
<A> skos:relatedMatch <C> .
```

Exemple 65 (non-implication)

```
<A> skos:closeMatch <B> .
<B> skos:closeMatch <C> .
```

n'implique pas

```
<A> skos:closeMatch <C> .
```

10.6.4. Propriétés d'Alignement et Refléxivité

Aucune des propriétés SKOS d'alignement n'est réflexive, et **aucune** n'est irréflexive.

Par le fait que `skos:exactMatch`, `skos:broadMatch` et `skos:relatedMatch` ne sont **pas irréflexives**, le graphe ci-dessous est **compatible** avec le modèle de données SKOS.

Exemple 66 (compatible)

```
<A> skos:exactMatch <A> .
<B> skos:broadMatch <B> .
<C> skos:relatedMatch <C> .
```

Consultez également la Section 8 sur la réflexivité des propriétés SKOS de relation sémantiques.

10.6.5. Cycles et Chemins Hiérarchiques Multiples Utilisant skos:broadMatch

Il n'y a pas de règle d'intégrité formelle empêchant les cycles ou les chemins hiérarchiques multiples dans un graphe de liens d'alignement hiérarchiques.

Le graphe ci-dessous contient deux cycles utilisant `skos:broadMatch`. Ce graphe est **compatible** avec le modèle de données SKOS.

Exemple 67 (compatible)

```
<A> skos:broadMatch <B> .
<B> skos:broadMatch <A> .

<X> skos:broadMatch <Y> .
<Y> skos:broadMatch <Z> .
<Z> skos:broadMatch <X> .
```

Le graphe ci-dessous contient deux chemins hiérarchiques possibles utilisant `skos:broadMatch`. Ce graphe est **compatible** avec le modèle de données SKOS.

Exemple 68 (compatible)

```
<A> skos:broadMatch <B> .
<B> skos:broadMatch <C> .
<A> skos:broadMatch <C> .
```

Consultez également la Section 8 sur les cycles et les chemins hiérarchiques multiples utilisant `skos:broader`.

10.6.6. Cycles Utilisant skos:exactMatch et skos:closeMatch

Exemple 69 (implication)

```
<A> skos:exactMatch <B>
```

implique

```
<A> skos:exactMatch <A> .
<A> skos:closeMatch <A> .
```

De par le raisonnement ci-dessus (qui découle d'une combinaison de S42, S44 et S45), les applications doivent se montrer capables de gérer des cycles de `skos:exactMatch` et `skos:closeMatch`.

10.6.7. Chaines de Sous-Propriétés Utilisant skos:exactMatch

Le modèle de données SKOS ne contient pas d'axiomes de chainage des sous-propriétés utilisant les propriétés `skos:exactMatch` ou `skos:closeMatch`. Par conséquent les raisonnements illustrés ci-dessous ne sont **pas** supportés.

Exemple 70 (non-implication)

```
<A> skos:exactMatch <B> .
<B> skos:broadMatch <C> .
```

n'implique pas

```
<A> skos:broadMatch <C> .
```

Exemple 71 (non-implication)

```
<A> skos:exactMatch <B> .
<B> skos:relatedMatch <C> .
```

n'implique pas

```
<A> skos:relatedMatch <C> .
```

Exemple 72 (non-implication)

```
<A> skos:closeMatch <B> .
<B> skos:broadMatch <C> .
```

n'implique pas

```
<A> skos:broadMatch <C> .
```

Exemple 73 (non-implication)

```
<A> skos:closeMatch <B> .
<B> skos:relatedMatch <C> .
```

n'implique pas

```
<A> skos:relatedMatch <C> .
```

10.6.8. skos:closeMatch, skos:exactMatch, owl:sameAs, owl:equivalentClass, owl:equivalentProperty

OWL propose trois propriétés qui pourraient, au premier abord, sembler similaires à `skos:closeMatch` ou à `skos:exactMatch`. `owl:sameAs` est utilisée pour relier deux individus dans une ontologie, et indique qu'ils représentent le même individu (c'est-à-dire la même ressource). `owl:equivalentClass` est utilisée pour relier deux classes dans une ontologie, et indique que ces deux classes ont la même extension de classe. `owl:equivalentProperty` est utilisée pour relier deux propriétés dans une ontologie et indique que ces deux propriétés ont la même extension de propriété.

`skos:closeMatch` et `skos:exactMatch` sont utilisées pour relier des concepts SKOS dans des schémas de concepts différents. Un lien `skos:closeMatch` indique que deux concepts sont suffisamment similaires pour être utilisé de façon interchangeable par **certaines** applications de recherche d'information. Un lien `skos:exactMatch` indique un fort degré de confiance que deux concepts puissent être utilisés de façon interchangeable dans une large majorité d'applications de recherche d'informations.

L'utilisation de `owl:sameAs`, `owl:equivalentClass` ou `owl:equivalentProperty` pour relier des concepts SKOS dans des schémas de concepts différents serait inappropriée, car elle aurait des implications logiques indésirables.

L'exemple ci-dessous illustre quelque-unes des implications indésirables qui résulteraient d'une telle utilisation de `owl:sameAs`.

Exemple 74 (implication)

```
<A> rdf:type skos:Concept ;
  skos:prefLabel "amour"@fr ;
  skos:inScheme <UnScheme> .

<B> rdf:type skos:Concept ;
  skos:prefLabel "adoration"@fr ;
  skos:inScheme <UnAutreScheme> .

<A> owl:sameAs <B> .
```

implique

```
<A>
  skos:prefLabel "amour"@fr ;
  skos:prefLabel "adoration"@fr ;
  skos:inScheme <UnScheme> ;
  skos:inScheme <UnAutreScheme> .

<B>
  skos:prefLabel "amour"@fr ;
  skos:prefLabel "adoration"@fr ;
  skos:inScheme <UnScheme> ;
  skos:inScheme <UnAutreScheme> .
```

Dans cet exemple, l'utilisation de `owl:sameAs` pour relier deux concepts de schémas de concepts différents amène effectivement à une inconsistence avec le modèle de données SKOS, car aussi bien `<A>` que `<B>` portent dorénavant deux libellés préférentiels dans la même langue. Ce ne sera pas pour autant toujours le cas.

---

11. Références

[AGROVOC]
AGROVOC Thesaurus, Food and Agriculture Organization of the United Nations (FAO). Disponible à http://www.fao.org/agrovoc

[BCP47]
Tags for Language Identification, A. Phillips et M. Davis, Editeurs, Septembre 2006. Disponible à http://www.rfc-editor.org/rfc/bcp/bcp47.txt

[BS8723-2]
BS8723 Structured Vocabularies for Information Retrieval Part 2: Thesauri, British Standards Institution (BSI), 2005.

[BS8723-3]
BS8723 Structured Vocabularies for Information Retrieval Part 3: Vocabularies Other Than Thesauri, British Standards Institution (BSI), 2005.

[COOLURIS]
Cool URIs for the Semantic Web, Leo Sauermann and Richard Cyganiak, Editors, W3C Interest Group Note, 31 March 2008, http://www.w3.org/TR/2008/NOTE-cooluris-20080331/. Dernière version disponible à http://www.w3.org/TR/cooluris/

[ISO2788]
ISO 2788:1986 Documentation -- Guidelines for the establishment and development of monolingual thesauri, International Organization for Standardization (ISO), 1986.

[LCSH]
Library of Congress Subject Headings, The Library of Congress Cataloging Distribution Service. Disponible à http://www.loc.gov/cds/lcsh.html and at http://id.loc.gov/

[NTRIPLES]
RDF Test Cases, Jan Grant and Dave Beckett, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/. Dernière version available at http://www.w3.org/TR/rdf-testcases/

[OWL-GUIDE]
OWL Web Ontology Language Guide, Michael K. Smith, Chris Welty and Deborah L. McGuinness, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-owl-guide-20040210/. Dernière version disponible à http://www.w3.org/TR/owl-guide/

[OWL-REFERENCE]
OWL Web Ontology Language Reference, Mike Dean and Guus Schreiber, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-owl-ref-20040210/. Dernière version disponible à http://www.w3.org/TR/owl-ref/

[OWL-SEMANTICS]
OWL Web Ontology Language Semantics and Abstract Syntax, Peter F. Patel-Schneider, Patrick Hayes and Ian Horrocks, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-owl-semantics-20040210/. Dernière version available at http://www.w3.org/TR/owl-semantics/

[RDF-CONCEPTS]
Resource Description Framework (RDF): Concepts and Abstract Syntax, Graham Klyne and Jeremy J. Carroll, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. Dernière version available at http://www.w3.org/TR/rdf-concepts/
[RDF-PRIMER]
RDF Primer, Frank Manola and Eric Miller, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-primer-20040210/. Dernière version disponible à http://www.w3.org/TR/rdf-primer/
[RDF-SEMANTICS]
RDF Semantics, Patrick Hayes, Editor, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-mt-20040210/. Dernière version disponible à http://www.w3.org/TR/rdf-mt/
[RDF-XML]
RDF/XML Syntax Specification (Revised), Dave Beckett, Editor, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/. Dernière version disponible à http://www.w3.org/TR/rdf-syntax-grammar/
[RDFS]
RDF Vocabulaire Description Language 1.0: RDF Schema, Dan Brickley and R. V. Guha, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-schema-20040210/. Dernière version available at http://www.w3.org/TR/rdf-schema/
[RECIPES]
Best Practice Recipes for Publishing RDF Vocabularies, Diego Berrueta and Jon Phipps, Editors, W3C Working Draft, 23 January 2008, http://www.w3.org/TR/2008/WD-swbp-vocab-pub-20080123/. Dernière version disponible à http://www.w3.org/TR/swbp-vocab-pub/
[SKOS-HTML]
SKOS Namespace Document - HTML Variant. Dernière version disponible à http://www.w3.org/TR/skos-reference/skos.html
[SKOS-PRIMER]
SKOS Simple Knowledge Organization System Primer, Antoine Isaac and Ed Summers, Editors, W3C Working Group Note, 18 August 2009, http://www.w3.org/TR/2009/NOTE-skos-primer-20090818. Dernière version disponible à http://www.w3.org/TR/skos-primer
[SKOS-RDF]
SKOS Namespace Document - RDF/XML Variant. Dernière version disponible à http://www.w3.org/TR/skos-reference/skos.rdf
[SKOS-RDF-OWL1-DL]
SKOS RDF Schema - OWL 1 DL Sub-set. Dernière version disponible à http://www.w3.org/TR/skos-reference/skos-owl1-dl.rdf
[SKOS-UCR]
SKOS Use Cases and Requirements, Antoine Isaac, Jon Phipps and Daniel Rubin, Editors, W3C Working Group Note, 18 August 2009, http://www.w3.org/TR/2009/NOTE-skos-ucr-20090818. Dernière version disponible à http://www.w3.org/TR/skos-ucr
[SKOS-XL-HTML]
SKOS-XL Namespace Document - HTML Variant. Dernière version disponible à http://www.w3.org/TR/skos-reference/skos-xl.html
[SKOS-XL-RDF]
SKOS-XL Namespace Document - RDF/XML Variant. Dernière version disponible à http://www.w3.org/TR/skos-reference/skos-xl.rdf
[SPARQL]
SPARQL Query Language for RDF, Eric Prud'hommeaux and Andy Seaborne, Editors, W3C Recommendation, 15 January 2008, http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/. Dernière version disponible à http://www.w3.org/TR/rdf-sparql-query/
[SW]
W3C Semantic Web Activity. Disponible à http://www.w3.org/2001/sw/
[SWBP-DATATYPES]
XML Schema Datatypes in RDF and OWL, Jeremy J. Carroll and Jeff Z. Pan, Editors, W3C Working Group Note, 14 March 2006, http://www.w3.org/TR/2006/NOTE-swbp-xsch-datatypes-20060314/. Dernière version disponible à http://www.w3.org/TR/swbp-xsch-datatypes/
[TURTLE]
Turtle - Terse RDF Triple Language, David Beckett and Tim Berners-Lee, W3C Team Submission, 14 January 2008, http://www.w3.org/TeamSubmission/2008/SUBM-turtle-20080114/. Dernière version disponible à http://www.w3.org/TeamSubmission/turtle/
[WEBARCH]
Architecture of the World Wide Web, Volume One, Ian Jacobs and Norman Walsh, Editors, W3C Recommendation, 15 December 2004, http://www.w3.org/TR/2004/REC-webarch-20041215/. Dernière version disponible à http://www.w3.org/TR/webarch/
[XML-SCHEMA]
XML Schema Part 2: Datatypes Second Edition, Paul V. Biron and Ashok Malhotra, Editors, W3C Recommendation, 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/. Dernière version available at http://www.w3.org/TR/xmlschema-2/

12. Remerciements

Ce document est le résultat de discussions approfondies au sein du Groupe de Travail W3C sur le Déploiement du Web Sémantique. Ce document s'appuie sur les expériences de groupes de travail ou de projets précédents, en particulier SWAD-Europe et le Groupe de Travail W3C sur les Bonnes Pratiques et le Déploiement du Web Sémantique. Les participants à la liste de diffusion public-esw-thes ont également apporté des contributions significatives.

Appendice A. Classes et Propriétés SKOS

A.1. Classes du Modèle de Données SKOS

skos:Collection	
URI:	`http://www.w3.org/2004/02/skos/core#Collection`
Définition:	Section 9. Collections de Concepts
Libellé anglais:	*Collection*
Classes disjointes:	`skos:Concept` `skos:ConceptScheme`
skos:Concept	
URI:	`http://www.w3.org/2004/02/skos/core#Concept`
Définition:	Section 3. La Classe skos:Concept
Libellé anglais:	*Concept*
Classes disjointes:	`skos:Collection` `skos:ConceptScheme`
skos:ConceptScheme	
URI:	`http://www.w3.org/2004/02/skos/core#ConceptScheme`
Définition:	Section 4. Concept Schemes
Libellé anglais:	*Concept Scheme*
Classes disjointes:	`skos:Collection` `skos:Concept`
skos:OrderedCollection	

URI:	http://www.w3.org/2004/02/skos/core#OrderedCollection
Définition:	Section 9. Collections de Concepts
Libellé anglais:	*Ordered Collection*
Super-classes:	skos:Collection

A.2. Propriétés du Modèle de Données SKOS

skos:altLabel	
URI:	http://www.w3.org/2004/02/skos/core#altLabel
Définition:	Section 5. Libellés Lexicaux
Libellé anglais:	*alternative label*
Super-propriétés:	http://www.w3.org/2000/01/rdf-schema#label
skos:broadMatch	
URI:	http://www.w3.org/2004/02/skos/core#broadMatch
Définition:	Section 10. Propriétés d'Alignement
Libellé anglais:	*has broader match*
Super-propriétés:	skos:broader skos:mappingRelation
Inverse de:	skos:narrowMatch
skos:broader	
URI:	http://www.w3.org/2004/02/skos/core#broader
Définition:	Section 8. Relations Sémantiques
Libellé anglais:	*has broader*
Super-propriétés:	skos:broaderTransitive
Inverse de:	skos:narrower
skos:broaderTransitive	
URI:	http://www.w3.org/2004/02/skos/core#broaderTransitive
Définition:	Section 8. Relations Sémantiques
Libellé anglais:	*has broader transitive*
Super-propriétés:	skos:semanticRelation
Inverse de:	skos:narrowerTransitive
Autres caractéristiques:	Transitive
skos:changeNote	
URI:	http://www.w3.org/2004/02/skos/core#changeNote
Définition:	Section 7. Propriétés de Documentation
Libellé anglais:	*change note*
Super-propriétés:	skos:note
skos:closeMatch	
URI:	http://www.w3.org/2004/02/skos/core#closeMatch
Définition:	Section 10. Propriétés d'Alignement
Libellé anglais:	*has close match*
Super-propriétés:	skos:mappingRelation
Autres caractéristiques:	Symétrique
skos:definition	
URI:	http://www.w3.org/2004/02/skos/core#definition
Définition:	Section 7. Propriétés de Documentation
Libellé anglais:	*definition*
Super-propriétés:	skos:note
skos:editorialNote	
URI:	http://www.w3.org/2004/02/skos/core#editorialNote
Définition:	Section 7. Propriétés de Documentation
Libellé anglais:	*editorial note*

Super-propriétés:	skos:note

skos:exactMatch	
URI:	http://www.w3.org/2004/02/skos/core#exactMatch
Définition:	Section 10. Propriétés d'Alignement
Libellé anglais:	*has exact match*
Super-propriétés:	skos:closeMatch
Autres caractéristiques:	Transitive Symétrique

skos:example	
URI:	http://www.w3.org/2004/02/skos/core#example
Définition:	Section 7. Propriétés de Documentation
Libellé anglais:	*example*
Super-propriétés:	skos:note

skos:hasTopConcept	
URI:	http://www.w3.org/2004/02/skos/core#hasTopConcept
Définition:	Section 4. Concept Schemes
Libellé anglais:	*label*
Domaine:	skos:ConceptScheme
Portée:	skos:Concept
Inverse de:	skos:topConceptOf

skos:hiddenLabel	
URI:	http://www.w3.org/2004/02/skos/core#hiddenLabel
Définition:	Section 5. Libellés Lexicaux
Libellé anglais:	*hidden label*
Super-propriétés:	http://www.w3.org/2000/01/rdf-schema#label

skos:historyNote	
URI:	http://www.w3.org/2004/02/skos/core#historyNote
Définition:	Section 7. Propriétés de Documentation
Libellé anglais:	*history note*
Super-propriétés:	skos:note

skos:inScheme	
URI:	http://www.w3.org/2004/02/skos/core#inScheme
Définition:	Section 4. Concept Schemes
Libellé anglais:	*is in scheme*
Portée:	skos:ConceptScheme

skos:mappingRelation	
URI:	http://www.w3.org/2004/02/skos/core#mappingRelation
Définition:	Section 10. Propriétés d'Alignement
Libellé anglais:	*is in mapping relation with*
Super-propriétés:	skos:semanticRelation

skos:member	
URI:	http://www.w3.org/2004/02/skos/core#member
Définition:	Section 9. Collections de Concepts
Libellé anglais:	*has member*
Domaine:	skos:Collection
Portée:	union de skos:Concept et skos:Collection

skos:memberList	
URI:	http://www.w3.org/2004/02/skos/core#memberList
Définition:	Section 9. Collections de Concepts
Libellé anglais:	*has member list*
Domaine:	skos:OrderedCollection

Portée:	http://www.w3.org/1999/02/22-rdf-syntax-ns#List
Autres caractéristiques:	Fonctionelle
skos:narrowMatch	
URI:	http://www.w3.org/2004/02/skos/core#narrowMatch
Définition:	Section 10. Propriétés d'Alignement
Libellé anglais:	*has narrower match*
Super-propriétés:	skos:mappingRelation skos:narrower
Inverse de:	skos:broadMatch
skos:narrower	
URI:	http://www.w3.org/2004/02/skos/core#narrower
Définition:	Section 8. Relations Sémantiques
Libellé anglais:	*has narrower*
Super-propriétés:	skos:narrowerTransitive
Inverse de:	skos:broader
skos:narrowerTransitive	
URI:	http://www.w3.org/2004/02/skos/core#narrowerTransitive
Définition:	Section 8. Relations Sémantiques
Libellé anglais:	*has narrower transitive*
Super-propriétés:	skos:semanticRelation
Inverse de:	skos:broaderTransitive
Autres caractéristiques:	Transitive
skos:notation	
URI:	http://www.w3.org/2004/02/skos/core#notation
Définition:	Section 6. Notations
Libellé anglais:	*notation*
skos:note	
URI:	http://www.w3.org/2004/02/skos/core#note
Définition:	Section 7. Propriétés de Documentation
Libellé anglais:	*note*
skos:prefLabel	
URI:	http://www.w3.org/2004/02/skos/core#prefLabel
Définition:	Section 5. Libellés Lexicaux
Libellé anglais:	*preferred label*
Super-propriétés:	http://www.w3.org/2000/01/rdf-schema#label
skos:related	
URI:	http://www.w3.org/2004/02/skos/core#related
Définition:	Section 8. Relations Sémantiques
Libellé anglais:	*has related*
Super-propriétés:	skos:semanticRelation
Autres caractéristiques:	Symétrique
skos:relatedMatch	
URI:	http://www.w3.org/2004/02/skos/core#relatedMatch
Définition:	Section 10. Propriétés d'Alignement
Libellé anglais:	*has related match*
Super-propriétés:	skos:mappingRelation skos:related
Autres caractéristiques:	Symétrique
skos:scopeNote	
URI:	http://www.w3.org/2004/02/skos/core#scopeNote
Définition:	Section 7. Propriétés de Documentation
Libellé anglais:	*scope note*

Super-propriétés:	skos:note
skos:semanticRelation	
URI:	http://www.w3.org/2004/02/skos/core#semanticRelation
Définition:	Section 8. Relations Sémantiques
Libellé anglais:	*is in semantic relation with*
Domaine:	skos:Concept
Portée:	skos:Concept
skos:topConceptOf	
URI:	http://www.w3.org/2004/02/skos/core#topConceptOf
Définition:	Section 4. Concept Schemes
Libellé anglais:	*is top concept in scheme*
Super-propriétés:	skos:inScheme
Inverse de:	skos:hasTopConcept

Appendice B. SKOS eXtension pour les Libellés (SKOS-XL)

Cet appendice définit une extension **optionelle** au Système Simple d'Organisation de Connaissances, appelée SKOS eXtension pour les Libellés (SKOS-XL). Cette extension fournit la possibilité d'identifier, décrire et lier des entités lexicales.

Une classe spéciale d'entités lexicales, appelée `skosxl:Label`, est introduite. Chaque instance de cette classe porte une unique forme litérale en RDF, mais deux instances de cette classe ne représente pas forcément le même individu si elles partagent la même forme litérale.

Trois propriétés de libellé, `skosxl:prefLabel`, `skosxl:altLabel` et `skosxl:hiddenLabel`, sont introduites. Ces propriétés sont utilisées pour libeller des concepts SKOS avec des instances de `skosxl:Label`, et sont pour le reste analogues aux propriétés ayant le même nom local définies en SKOS (`skos:prefLabel`, `skos:altLabel` et `skos:hiddenLabel` respectivement).

Le modèle de données SKOS définit par ailleurs la propriété `skosxl:labelRelation`. Cette propriété peut être utilisée pour indiquer un lien direct (binaire) entre instances de `skosxl:Label`. Cette propriété sert avant tout comme base à des extensions pour spécifier des types de liens plus précis. Aucune spécialisation de `skosxl:labelRelation` n'est fournie, mais quelques exemples de la marche à suivre sont donnés.

B.1. Espace de Noms et Vocabulaire SKOS-XL

L'URI de l'espace de noms SKOS-XL est:

- **http://www.w3.org/2008/05/skos-xl#**

Dans ce document le préfixe `skosxl:` est utilisé comme abbréviation de l'URI de l'espace de noms SKOS-XL.

Le vocabulaire SKOS-XL est l'ensemble des URIs indiquées dans la colonne de gauche du tableau ci-dessous.

Table 2. Le vocabulaire SKOS-XL

URI	Définie par (section dans cet appendice)
`skosxl:Label`	La classe skosxl:Label
`skosxl:literalForm`	La classe skosxl:Label
`skosxl:prefLabel`	skosxl:Labels Préférrés, Alternatifs et Cachés
`skosxl:altLabel`	skosxl:Labels Préférrés, Alternatifs et Cachés
`skosxl:hiddenLabel`	skosxl:Labels Préférrés, Alternatifs et Cachés
`skosxl:labelRelation`	Liens entre skosxl:Labels

Ici le "vocabulaire SKOS-XL" fait référence à l'union du vocabulaire SKOS et du vocabulaire SKOS-XL.

Ici le "modèle de données XL" fait référence aux définitions de classes et de propriétés définies dans cet appendice seulement. "Le modèle de données SKOS+XL" fait référence à l'union du modèle de données défini dans les sections 3 à 10 et au modèle de données XL.

B.2. La classe skosxl:Label

B.2.1. Préambule

La classe `skosxl:Label` est une classe particulière d'entités lexicales.

Une instance de la classe `skosxl:Label` est une ressource et peut être nommées avec une URI.

Une instance de la classe `skosxl:Label` a une unique forme litérale. Cette forme litérale est un litéral RDF simple (qui est une chaine de caractères UNICODE plus une étiquette de langue optionnelle, voir [RDF-CONCEPTS]). Le propriété `skosxl:literalForm` est utilisée pour donner la forme litérale d'un `skosxl:Label`.

Si deux instances de la classe `skosxl:Label` ont la même forme litérale, elles ne sont **pas** nécessairement la même ressource.

B.2.2. Définitions des Classes et Propriétés

S47	`skosxl:Label` est une instance de `owl:Class`.
S48	`skosxl:Label` est disjoint avec chacune des classes `skos:Concept`, `skos:ConceptScheme` et `skos:Collection`.
S49	`skosxl:literalForm` est une instance de `owl:DatatypeProperty`.
S50	Le `rdfs:domain` de `skosxl:literalForm` est la classe `skosxl:Label`.
S51	Le `rdfs:range` de `skosxl:literalForm` est la classe des litéraux simples RDF.
S52	`skosxl:Label` est une sous-classe de la restriction sur `skosxl:literalForm` avec cardinalité exactement égale à 1.

B.2.3. Exemples

L'exemple ci-dessous décrit un `skosxl:Label` nommé avec l'URI `<http://example.com/ns/A>`, avec la forme litérale "amour" en Français.

Exemple 75 (compatible)

```
<A> rdf:type skosxl:Label ; skosxl:literalForm "amour"@fr .
```

Les quatres exemples ci-dessous sont tous **incompatibles** par rapport au modèle de données XL, car un `skosxl:Label` est décrit avec deux formes litérales différentes.

Exemple 76 (incompatible)

```
<B> rdf:type skosxl:Label ; skosxl:literalForm "amour" ; skosxl:literalForm "adoration" .
```

Exemple 77 (incompatible)

```
<B> rdf:type skosxl:Label ; skosxl:literalForm "love"@en ; skosxl:literalForm "amour"@fr .
```

Exemple 78 (incompatible)

```
<B> rdf:type skosxl:Label ; skosxl:literalForm "love"@en-GB ; skosxl:literalForm "love"@en-US .
```

Exemple 79 (incompatible)

```
<B> rdf:type skosxl:Label ; skosxl:literalForm "東"@ja-Hani ; skosxl:literalForm "ひがし"@ja-Hira .
```

B.2.4. Notes

B.2.4.1. Identité et Raisonnement

Comme défini plus haut, toute instance de la classe `skosxl:Label` a **une et une seule forme litérale**. En d'autres termes, il existe une fonction associant l'extension de la classe `skosxl:Label` à l'ensemble des litéraux simples RDF. Cette fonction est définie par l'extension de la propriété `skosxl:literalForm`. Il faut noter deux choses à propos de cette fonction.

Premièrement, cette fonction n'est **pas** injective. En d'autre termes il n'y a **pas** de correspondance 1-1 entre les instances de `skosxl:Label` et l'ensemble des litéraux simples RDF (en réalité il s'agit d'une correspondance *-1). Cela veut dire que deux instances de `skosxl:Label` qui ont la même forme litérale ne sont **pas nécessairement** le même individu. Donc, par exemple, le raisonnement illustré ci-dessous n'est **pas** supporté par le modèle de données XL.

Exemple 80 (non-implication)

```
<A> skosxl:literalForm "amour"@fr .
<B> skosxl:literalForm "amour"@fr .
```

n'implique pas

```
<A> owl:sameAs <B> .
```

Deuxièmement, cette fonction n'est **pas** surjective. En d'autres termes, pour un litéral simple donné `l`, il peut ne pas y avoir d'instance de `skosxl:Label` avec la forme litérale `l`.

B.2.4.2. Appartenance à un Schéma de Concepts

L'appartenance d'une instance de `skosxl:Label` à un Schéma de Concepts SKOS peut être déclaré avec la propriété `skos:inScheme`.

Exemple 81 (compatible)

```
<A> rdf:type skosxl:Label ; skosxl:literalForm "amour"@fr ; skos:inScheme <MonSchema> .
```

B.3. skosxl:Labels Préférrés, Alternatifs et Cachés

B.3.1. Préambule

Les trois propriétés `skosxl:prefLabel`, `skosxl:altLabel` et `skosxl:hiddenLabel` sont utilisées pour donner respectivement les libellés préférés, alternatifs et cachés d'une ressource, lorsque ces libellés sont instances de la classe `skosxl:Label`. Ces propriétés sont analogues aux propriétés de même nom local définies dans le vocabulaire SKOS, et il existe des dépendances logiques entre ces deux ensembles de propriétés qui sont définies ci-dessous.

B.3.2. Définitions des Classes et Propriétés

S53	`skosxl:prefLabel`, `skosxl:altLabel` et `skosxl:hiddenLabel` sont chacune instances de `owl:ObjectProperty`.
S54	Le `rdfs:range` de chacune de `skosxl:prefLabel`, `skosxl:altLabel` et `skosxl:hiddenLabel` est la classe `skosxl:Label`.
S55	Le chainage des propriétés (`skosxl:prefLabel`, `skosxl:literalForm`) est une sous-propriété de `skos:prefLabel`.
S56	Le chainage des propriétés (`skosxl:altLabel`, `skosxl:literalForm`) est une sous-propriété de `skos:altLabel`.
S57	Le chainage des propriétés (`skosxl:hiddenLabel`, `skosxl:literalForm`) est une sous-propriété de `skos:hiddenLabel`.
S58	`skosxl:prefLabel`, `skosxl:altLabel` et `skosxl:hiddenLabel` sont des propriétés disjointes deux à deux.

B.3.3. Exemples

L'exemple ci-dessous illustre l'utilisation des trois propriétés XL de libellé, et est compatible avec le modèle de données SKOS+XL.

Exemple 82 (compatible)

```
<Love>
  skosxl:prefLabel <A> ;
  skosxl:altLabel <B> ;
  skosxl:hiddenLabel <C> .

<A> rdf:type skosxl:Label ;
  skosxl:literalForm "love"@en .

<B> rdf:type skosxl:Label ;
  skosxl:literalForm "adoration"@en .

<C> rdf:type skosxl:Label ;
  skosxl:literalForm "luv"@en .
```

B.3.4. Notes

B.3.4.1. Retrouver des Libellés Lexicaux SKOS

Les axiomes de chainage des propriétés S55, S56 et S57 permettent de rapporter des libellés XL à des libellés lexicaux SKOS de base via une inférence. Ceci est illustré dans l'exemple ci-dessous.

Exemple 83 (implication)

```
<Love>
  skosxl:prefLabel <A> ;
  skosxl:altLabel <B> ;
  skosxl:hiddenLabel <C> .

<A> rdf:type skosxl:Label ;
  skosxl:literalForm "love"@en .

<B> rdf:type skosxl:Label ;
  skosxl:literalForm "adoration"@en .

<C> rdf:type skosxl:Label ;
  skosxl:literalForm "luv"@en .
```

implique

```
<Love>
  skos:prefLabel "love"@en ;
  skos:altLabel "adoration"@en ;
  skos:hiddenLabel "luv"@en .
```

B.3.4.2. Intégrité des Libellés en SKOS-XL

Dans la Section 5, deux conditions d'intégrité ont été définies sur les propriétés de libellé SKOS basiques. Premièrement, les propriétés `skos:prefLabel`, `skos:altLabel` et `skos:hiddenLabel` sont disjointes deux à deux. Deuxièmement, une ressource n'a pas plus d'une valeur de `skos:prefLabel` par langue. A cause des axiomes de chainage de propriétés définis ci-dessus, les quatres exemples suivants, bien que compatibles avec le seul modèle de données XL, ne sont **pas** compatibles avec le modèle de données SKOS+XL.

Exemple 84 (incompatible)

```
# Deux libellés préférentiels dans la même langue

<Amour> skosxl:prefLabel <A> ; skosxl:prefLabel <B> .
<A> skosxl:literalForm "amour"@fr .
<B> skosxl:literalForm "adoration"@fr .
```

Exemple 85 (incompatible)

```
# Incompatibilité entre libellés préférentiels et alternatifs

<Amour> skosxl:prefLabel <A> ; skosxl:altLabel <B> .
<A> skosxl:literalForm "amour"@fr .
<B> skosxl:literalForm "amour"@fr .
```

Exemple 86 (incompatible)

```
# Incompatibilité entre libellés alternatifs et cachés

<Amour> skosxl:altLabel <A> ; skosxl:hiddenLabel <B> .
<A> skosxl:literalForm "amour"@fr .
<B> skosxl:literalForm "amour"@fr .
```

Exemple 87 (incompatible)

```
# Incompatibilité entre libellés préférés et cachés

<Amour> skosxl:prefLabel <A> ; skosxl:hiddenLabel <B> .
<A> skosxl:literalForm "amour"@fr .
<B> skosxl:literalForm "amour"@fr .
```

B.4. Liens Entre skosxl:Labels

B.4.1. Préambule

Cette section définit une façon de faire pour représenter des liens entre instances de la classe `skosxl:Label`.

Notez que le vocabulaire défini dans cette section n'a pas pour but d'être utilisé directement, mais d'être utilisé comme une base pour des extensions qui répondraient à des besoins de nommage spécifiques.

B.4.2. Définitions des Classes et des Propriétés

S59	`skosxl:labelRelation` est une instance de `owl:ObjectProperty`.
S60	Le `rdfs:domain` de `skosxl:labelRelation` est la classe `skosxl:Label`.
S61	Le `rdfs:range` de `skosxl:labelRelation` est la classe `skosxl:Label`.
S62	`skosxl:labelRelation` est une instance de `owl:SymmetricProperty`.

B.4.3. Exemples

L'exemple ci-dessous illustre un lien entre deux instances de la classe `skosxl:Label`.

Exemple 88 (compatible)

```
<A> rdf:type skosxl:Label ; skosxl:literalForm "love" .
<B> rdf:type skosxl:Label ; skosxl:literalForm "adoration" .
<A> skosxl:labelRelation <B> .
```

B.4.4. Notes

B.4.4.1. Spécialisations de cet Usage

Comme mentionné ci-dessus, la propriété `skosxl:labelRelation` sert comme base à des extensions, qui peuvent la spécialiser pour répondre à des scénarios de nommage spécifiques.

Dans l'exemple ci-dessous, une tierce-partie a spécialisé la propriété `skos:labelRelation` pour exprimer des relations d'acronyme, et l'a utilisé pour exprimer le fait que "FAO" est un acronyme de "Food and Agriculture Organization".

Exemple 89 (compatible)

```
# Définir d'abord une extension de skosxl:labelRelation
ex:acronym rdfs:subPropertyOf skosxl:labelRelation .

# Ensuite utiliser cette nouvelle propriété
<A> rdf:type skosxl:Label ; skosxl:literalForm "FAO"@en .
<B> rdf:type skosxl:Label ; skosxl:literalForm "Food and Agriculture Organization"@en .
<B> ex:acronym <A> .
```

Notez qu'une sous-propriété d'une propriété symétrique n'est pas nécessairement symétrique.

B.5. Résumé du Modèle SKOS-XL

La table suivante donne un résumé du vocabulaire SKOS-XL.

B.5.1 Classes dans le modèle de données SKOS-XL

skosxl:Label	
URI:	`http://www.w3.org/2008/05/skos-xl#Label`
Définition:	Section B.2. La Classe skosxl:Label
Libellé anglais:	*Label*
Super-classes:	restriction sur `skosxl:literalForm` avec cardinalité exactement égale à 1

Classes disjointes:	`skos:Collection`
	`skos:Concept`
	`skos:ConceptScheme`

B.5.2. Propriétés dans le Modèle de Données SKOS-XL

skosxl:altLabel	
URI:	`http://www.w3.org/2008/05/skos-xl#altLabel`
Définition:	Section B.3. skosxl:Labels Préférrés, Alternatifs et Cachés
Libellé anglais:	*alternative label*
Portée:	`skosxl:Label`
skosxl:hiddenLabel	
URI:	`http://www.w3.org/2008/05/skos-xl#hiddenLabel`
Définition:	Section B.3. skosxl:Labels Préférrés, Alternatifs et Cachés
Libellé anglais:	*hidden label*
Portée:	`skosxl:Label`
skosxl:labelRelation	
URI:	`http://www.w3.org/2008/05/skos-xl#labelRelation`
Définition:	Section B.4. Liens entre skosxl:Labels
Libellé anglais:	*label relation*
Domaine:	`skosxl:Label`
Portée:	`skosxl:Label`
Autres caractéristiques:	Symétrique
skosxl:literalForm	
URI:	`http://www.w3.org/2008/05/skos-xl#literalForm`
Définition:	Section B.2. La Classe skosxl:Label
Libellé anglais:	*literal form*
Domaine:	`skosxl:Label`
skosxl:prefLabel	
URI:	`http://www.w3.org/2008/05/skos-xl#prefLabel`
Définition:	Section B.3. skosxl:Labels Préférrés, Alternatifs et Cachés
Libellé anglais:	*preferred label*
Portée:	`skosxl:Label`

Appendice C. Documents des Espaces de Noms SKOS et SKOS-XL

D'après le document d'Architecture du World Wide Web, Volume Un [WEBARCH], un "document d'espace de noms" est une "ressource informationnelle qui contient les informations utiles, utilisables par la machine et/ou par un humain, à propos des termes d'un espace de noms".

Le vocabulaire SKOS est une ressource conceptuelle identifiée par l'URI d'espace de noms `http://www.w3.org/2004/02/skos/core#`. Le définition normative du vocabulaire SKOS se trouve dans la Référence SKOS (ce document).

Les documents d'espace de noms suivants donnent d'autres représentations alternatives du vocabulaire SKOS:

C.1. Document d'Espace de Noms SKOS - Version HTML (normatif)

Le vocabulaire SKOS est résumé dans le document d'espace de noms SKOS - Version HTML [SKOS-HTML], qui est accessible à l'URI de l'espace de noms http://www.w3.org/2004/02/skos/core# via négociation de contenu en utilisant la méthode 3 des "Bonnes Pratiques pour Publier un Vocabulaire" [RECIPES]. Les clients demandant une version HTML ou XHTML doivent inclure une entête "Accept" appropriée dans leur requête HTTP. Le document d'espace de noms SKOS - Version HTML peut également être référencé directement avec son URI: http://www.w3.org/TR/skos-reference/skos.html.

Le document d'espace de noms SKOS - Version HTML duplique l' Appendice A. Classes et Propriétés SKOS de ce document.

C.2. Document d'Espace de Noms SKOS - Version RDF/XML (normatif)

Le Document d'Espace de Noms SKOS - Version RDF/XML [SKOS-RDF] exprime le vocabulaire SKOS et son modèle de données (dans la mesure du possible) avec des triplets RDF. Il est accessible via négociation de contenu en utilisant la méthode 3 des "Bonnes Pratiques pour Publier un Vocabulaire" [RECIPES]. Les clients demandant une version RDF/XML doivent inclure une entête "Accept" appropriée dans leur requête HTTP. Le document d'espace de noms SKOS - Version RDF/XML peut également être référencé (et téléchargé) directement avec son URI: http://www.w3.org/TR/skos-reference/skos.rdf.

Il n'est pas possible d'exprimer toutes les définitions du modèle de données SKOS sous forme de triplets RDF, ce schéma forme donc un sous-ensemble normatif de la Référence SKOS. Le schéma RDF définit une ontologie OWL Full [OWL-SEMANTICS] [OWL-REFERENCE]. Les vocabulaires SKOS peuvent être définis comme instances de cette ontologie.

C.3. Schéma RDF SKOS - Sous-ensemble OWL 1 DL (informatif)

Par commodité pour les outils et les applications qui souhaiteraient fonctionner avec les contraintes OWL DL, le Schéma RDF SKOS - Sous-ensemble OWL 1 DL [SKOS-RDF-OWL1-DL] fournit un schéma modifié, informatif, qui est conforme à ces contraintes. Notez que ce schéma est obtenu par suppression des triplets représentants les axiomes qui violent les containtes OWL DL. D'autres suppressions seraient possibles.

Le sous-ensemble SKOS OWL 1 est accessible en utilisant sont URI: http://www.w3.org/TR/skos-reference/skos-owl1-dl.rdf

C.4. Document d'Espace de Noms SKOS-XL - Version HTML (normatif)

Le vocabulaire SKOS-XL est résumé dans le Document d'Espace de Noms SKOS-XL - Version HTML [SKOS-XL-HTML], qui est accessible à l'URI de l'espace de noms http://www.w3.org/2008/05/skos-xl# via négociation de contenu en utilisant la méthode 3 des "Bonnes Pratiques pour Publier un Vocabulaire" [RECIPES]. Les clients demandant une version HTML ou XHTML doivent inclure une entête "Accept" appropriée dans leur requête HTTP. Le Document d'Espace de Noms SKOS-XL - Version HTML peut également être référencé directement avec son URI: http://www.w3.org/TR/skos-reference/skos-xl.html.

Le Document d'Espace de Noms SKOS-XL - Version HTML duplique l' Appendice B.5 Résumé du Schéma SKOS-XL de ce document.

C.5. Document d'Espace de Noms SKOS-XL - Version RDF/XML (normatif)

Ce document en schéma RDF exprime le vocabulaire SKOS-XL et son modèle de données (dans la mesure du possible) avec des triplets RDF. Il est accessible depuis l'URI de l'espae de noms http://www.w3.org/2008/05/skos-xl# via la négociation de contenus en utilisant la méthode 3 des "Bonnes Pratiques pour Publier un Vocabulaire" [RECIPES]. Les clients demandant une version RDF/XML doivent inclure une entête "Accept" appropriée dans leur requête HTTP. Le schéma RDF peut également être référencé (et téléchargé) directement avec son URI: http://www.w3.org/TR/skos-reference/skos-xl.rdf.

Appendice D. Espace de noms SKOS : note historique

Le schéma SKOS définit son vocabulaire dans l'espace de noms http://www.w3.org/2004/02/skos/core#. Cet espace de noms était utilisé par le schéma SKOS d'origine qui a servi de point de départ à cette Recommandation. Pour cette raison, des éléments des versions précédentes du schéma interprétable par les machines ont été supprimés de la version actuelle. Dans certains cas, la définition ou la sémantique des éléments dans le schéma a évolué.

Garder l'espace de noms SKOS existant évite certains problèmes avec des systèmes d'organisation de connaissances qui utilisent déjà l'espace de noms SKOS. Les utilisateurs doivent cependant être conscients du changement de sémantique de `skos:broader` (et `skos:narrower`) qui *pourrait* impacter les applications SKOS.

Pour les éléments supprimés, aucune information de dépréciation explicite n'a été intégrée dans le schéma. Les anciennes versions du schéma SKOS, sont, cependant, disponibles sur la page "Historique des versions" du site web SKOS, et les éléments supprimés de la version récente du vocabulaire sont listés ci-dessous.

- `skos:symbol`
- `skos:prefSymbol`
- `skos:altSymbol`
- `skos:CollectableProperty`
- `skos:subject`
- `skos:isSubjectOf`
- `skos:primarySubject`
- `skos:isPrimarySubjectOf`
- `skos:subjectIndicator`

Dans le cas de `skos:broader` et `skos:narrower`, la sémantique de ces éléments de vocabulaire a changé - ces propriétés ne sont plus déclarées comme transitives. Par conséquent l'implication ci-dessous est fausse.

Exemple 90 (non-implication)

```
<A> skos:broader <B> .
<B> skos:broader <C> .
```

n'implique pas

```
<A> skos:broader <C> .
```

Une super-propriété transitive de `skos:broader` — `skos:broaderTransitive` — est fournie pour interroger la fermeture transitive des relations `skos:broader`. Une propriété similaire — `skos:narrowerTransitive` — est fournie pour interroger la fermeture transitive de `skos:narrower`.