Wikis sémantiques distribués sur réseaux pair-à-pair

Charbel Rahhal

Nancy-Université

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Wikis sémantiques distribués sur réseaux pair-à-pair

THÈSE

présentée et soutenue publiquement le 9 novembre 2010

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

Charbel RAHHAL

Composition du jury

Rapporteurs :	M. Fabien GANDON, Chargé de recherche et HDR à l'INRIA Sophia Antipolis M. Sylvain LECOMTE, Professeur à l'Université de Valenciennes et du Hainaut-Cambrésis
Examinateurs :	Mme Isabelle CHRISMENT, Professeur à l'Université Henri Poincaré, Nancy 1 M. Hervé MARTIN, Professeur à l'Université Joseph Fourier, Grenoble 1 M. Olivier FESTOR, Directeur de recherche à l'INRIA-Nancy Grand Est
Directeur de Thèse :	M. Pascal MOLLI, Professeur à l'Université de Nantes
Encadrante de Thèse :	Mme Hala SKAF-MOLLI, Maître de Conférences à l'Université de Nantes

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Cette thèse est effectuée en co-direction avec l'Univeristé Libanaise sous la direction de Hala Naja-Jazzar.
Elle a été partiellement financée par le projet Rorax (Réplication optimiste et réparation automatique de documents) du programme CEDRE.

Remerciements

Je tiens tout d'abord à remercier Hala Skaf-Molli pour avoir encadré mon travail durant ces années de thèse.

Je souhaite exprimer toute ma gratitude à Pascal Molli, mon directeur de thèse, pour son encadrement, ses conseils et de m'avoir dirigé dans mes recherches.

Je tiens à remercier Claude Godart, ex-responsable du projet ECOO (Environnements pour la COOpération), pour l'opportunité qu'il m'a offerte en m'accueillant au sein de son équipe.

Je souhaite présenter mes remerciements aux membres du jury pour l'attention particulière qu'ils ont bien voulu porter sur mes travaux.

À Fabien Gandon et Sylvain Lecomte pour m'avoir fait l'honneur d'accepter d'être rapporteurs de cette thèse. Je les remercie pour leur lecture approfondie de mon manuscrit et pour les remarques détaillées et très bénéfiques qu'ils m'ont donné.

À Hala Naja-Jazzar pour l'opportunité qu'elle m'a offerte.

À l'ensemble des membres de l'équipe SCORE (précédemment ECOO) avec qui j'ai eu le plaisir de travailler. Je pense particulièrement à Claude Godart, Pascal Molli, Hala Skaf-Molli, Gérald Oster, Claudia-Lavinia Ignat, Stéphane Weiss, Nawal Guermouche, Khaled Mobayed, Nagham Haddad et Hien Thi Thu Truong.

Je remercie aussi tous mes amis avec lesquels j'ai vécu des moments inoubliables. En particulier, à Nawal Guermouche et à Stéphane Weiss pour les nombreuses conversations captivantes tout au long des années de thèse. À Georges Khoury, pour son soutien lors du dernier mois de la thèse.

À tous ceux que j'ai oublié de citer, qu'ils m'en excusent. Enfin, je tiens à témoigner toute ma reconnaissance à mes sœurs et mes frères qui m'ont toujours soutenu et encouragé durant mes longues années d'études.

Table des matières

Table des figures

Introduction

1 Contexte et objectifs de la thèse

Le Web 2.0 [O'R05] a montré l'importance des systèmes collaboratifs. Il a permis de transformer des internautes en écrivains du Web. De nombreux outils sociaux sont disponibles comme les wikis, les blogs, les systèmes de messages instantanés et de visio-conférences. Ces outils sociaux sont utilisés par des larges communautés produisant une grande quantité d'informations. L'approche du Web sémantique [BLHL01] vise à structurer ces informations de manière à rendre le Web compréhensible par les machines. Un des objectifs du Web sémantique est de permettre des recherches plus précises et une navigation plus efficace. Les outils sociaux évoluent en intégrant les technologies du Web sémantique i.e. RDF [RDF04], RDFS [RDF], OWL [OWL] etc. Les wikis en tant qu'outils sociaux suivent cette tendance et évoluent vers des wikis sémantiques comme Semantic MediaWiki [VKV+06], SweetWiki [BGE+08], IkeWiki [Sch06] et OntoWiki [AD+06].

L'introduction des technologies du Web sémantique dans les outils sociaux pose un certain nombre de problèmes :

- le passage à l'échelle et la tolérance aux pannes de ces systèmes est un problème très difficile. L'introduction des technologies du Web sémantique rend ces problèmes encore plus aigus. Le passage à l'échelle et la tolérance aux pannes des wikis sémantiques est clairement un problème ouvert.
- La garantie de la qualité des contenus est un problème général des systèmes collaboratifs. La qualité du contenu est liée aux processus de coordination des acteurs de la communauté [KK08]. L'introduction des technologies du Web sémantique positionne les wikis sémantiques à la frontière entre systèmes collaboratifs et ingénierie des ontologies. Le support des procédés d'acquisition et de maintenance des ontologies est un élément important de l'ingénierie des ontologies. Le support de ces procédés est clairement un problème ouvert dans le cadre des wikis sémantiques.

Le modèle de réplication optimiste permet d'adresser à la fois les problèmes de passage à l'échelle, de tolérance aux pannes et de représentation des procédés. Le modèle de réplication optimiste considère N sites répliquant des objets partagés. Un objet est modifié sur un site en appliquant des opérations localement. Ces opérations sont propagées aux autres sites. Les autres sites ré-exécutent les opérations reçues. Le système est correct s'il respecte des propriétés comme la causalité [Lam78] et la cohérence à terme [SS05a]. Le

système obtenu passe à l'échelle car il permet de répartir la charge sur les différents sites. La panne d'un site n'entraîne pas la panne du système. Le contrôle de la propagation des opérations entre les sites permet de représenter des procédés.

La contribution générale de cette thèse est de montrer comment il est possible d'instancier le modèle de réplication optimiste dans le cadre des wikis sémantiques. Nous nous plaçons dans un contexte où le nombre de sites n'est pas connu et varie à l'exécution à l'instar des réseaux pair-à-pair. En tant que système collaboratif, un système de réplication optimiste est correct s'il respecte le modèle de cohérence CCI (Causalité, Convergence, Intention) [SJZ+98] :

préservation de Causalité : l'exécution des opérations doit respecter la relation de précédence définie par Lamport [Lam78],

Convergence : lorsque toutes les répliques ont reçu toutes les opérations, elles ont le même document,

préservation de l'Intention : les effets de l'exécution d'une opération doivent être les mêmes sur tous les sites et son exécution ne doit pas modifier les effets d'opérations indépendantes.

Le développement d'un wiki sémantique pair-à-pair (P2P) basé sur une réplication optimiste soulève plusieurs questions telles que : comment manipuler et répliquer le nouveau type de données des wikis sémantiques (pages wikis et annotations) ? comment propager et synchroniser les changements sur ces données ? Et comment assurer le modèle de cohérence CCI sur ces données répliquées ?

Les approches de réplication dans le web sémantique P2P se concentrent sur le partage des données sémantiques et non sur leur édition collaborative. Dans le cadre des systèmes distribués, les algorithmes de synchronisation existants ne garantissent pas le modèle de cohérence CCI pour un type de données mêlant texte et annotations sémantiques.

Le défi de cette thèse est de proposer une instanciation du modèle de réplication optimiste pour les wikis sémantiques. L'instanciation du modèle de réplication optimiste sur ce type de données nécessite de définir les opérations d'édition sur ces données et leurs intentions, de proposer des modèles de propagation et d'adapter les algorithmes de synchronisation pour garantir CCI sur un type de données mêlant texte et annotations sémantiques.

2 Résumé des contributions

Nos résultats sont les suivants :

1. Nous avons proposé Swooki le premier wiki sémantique P2P. Swooki est une instanciation du modèle de réplication optimiste pour des données composées de pages wikis et d'annotations. Il combine les avantages des wikis sémantiques et ceux des wikis pair-à-pair.

- Nous avons défini les opérations d'édition du nouveau type de données répliqué et leurs intentions.
- Pour assurer la réception des opérations par tous les pairs, nous avons proposé d'utiliser l'algorithme Lpbcast [EGH+03] pour propager rapidement les modifications à tous les pairs connectés et un mécanisme d'anti-entropie [DGH+88] pour supporter le mode déconnecté.
- Nous avons modifié Woot [WUM07] un algorithme de réplication optimiste pour synchroniser le nouveau type de données tout en préservant l'intention des opérations et la convergence des pages et des annotations des pairs.
- De plus, nous avons développé un mécanisme d'annulation de groupe dans Swooki qui permet d'annuler toute modification.
- Nous avons montré que Swooki assure le modèle de cohérence CCI sur les pages wikis contenant des annotations sémantiques et des entrepôts sémantiques des pairs.

Ainsi Swooki adresse bien les motivations relatives au passage à l'échelle et à la tolérance aux pannes. Par contre, Swooki ne permet pas de représenter des procédés.

2. Pour supporter l'édition multi-synchrone et les procédés d'édition collaborative, nous avons proposé Distributed Semantic MediaWiki (DSMW) une instanciation du modèle de réplication optimiste pour les wikis sémantiques.
   - Nous avons défini le modèle de données de DSMW sous forme d'une ontologie qui permet d'exploiter l'historique au moyen des requêtes sémantiques.
   - Nous avons développé un modèle Publier/Souscrire basé sur des flux pour la propagation des changements entre les pairs. La propagation est faite sous le contrôle des utilisateurs, ce qui permet de créer un réseau ami-à-ami sémantique. En effet, les utilisateurs choisissent avec qui ils souhaitent collaborer, ce qui permet de modéliser des procédés d'édition collaborative et de supporter l'édition multi-synchrone.
   - Nous avons montré que DSMW assure le modèle de cohérence CCI pour les pages wikis sémantiques et les entrepôts sémantiques.

   Tout comme Swooki, DSMW supporte le modèle de cohérence CCI et préserve les mêmes intentions. Par contre, le modèle Publier/Souscrire de DSMW permet d'apporter l'édition multi-synchrone et l'implémentation des procédés d'édition collaborative.

3 Organisation du document

Notre mémoire s'organise de la manière suivante.

Dans le chapitre 1, nous présentons le contexte général et les motivations. Premièrement, nous présentons les problèmes de passage à l'échelle, de la tolérance aux pannes et de la coordination dans les wikis et les wikis sémantiques. Ensuite, nous nous intéressons

au modèle de réplication optimiste. Enfin, nous présentons les approches de réplication existantes qui pourraient être utilisées pour construire un wiki sémantique P2P.

Nous proposons Swooki au chapitre 2, une instanciation du modèle de réplication optimiste pour les wikis sémantiques. Dans ce chapitre, nous présentons la conception de Swooki. Nous détaillons son architecture et son implémentation ainsi que son mécanisme d'annulation.

Dans le chapitre 3, nous proposons Distributed Semantic MediaWiki (DSMW) une deuxième instanciation du modèle de réplication optimiste pour les wikis sémantiques. DSMW répond complètement aux motivations de notre problématique. Dans ce chapitre, nous présentons l'approche DSMW et son fonctionnement. Nous développons également son architecture et son implémentation.

Finalement, le dernier chapitre décrit le bilan de notre travail et présente les perspectives.

3.1 Publications

La contribution sur Swooki a donné lieu aux publications suivantes :

- Charbel Rahhal, Hala Skaf-Molli, and Pascal Molli. **Swooki, un wiki sémantique sur réseau pair-à-pair.** *In "Ingénierie des Systèmes d'Information", vol. 14, no 1, 2009, p. 117-140.*
- Hala Skaf-Molli, Charbel Rahhal, and Pascal Molli. **Peer-to-peer Semantic Wikis.** *In the 20th International Conference on Database and Expert Systems Applications - DEXA'09.* Linz, Austria, 31 August - 4 September 2009.
- Charbel Rahhal, Stéphane Weiss, Hala Skaf-Molli, Pascal Urso, and Pascal Molli. **Undo in peer-to-peer semantic wikis.** *In The 4th Workshop on Semantic Wikis : The Semantic Wiki Web (SemWiki2009) at the 6th Annual European Semantic Web Conference (ESWC).* Crete, Greece, June 2009.
- Charbel Rahhal Hala Skaf-Molli and Pascal Molli. **Swooki : A peer-to-peer semantic wiki (poster).** *In Third Semantic Wiki Workshop co-located with the 5th European Semantic Web Conference (ESWC).* Tenerife, Spain, June 2008.
- Hala Skaf-Molli, Charel Rahhal and Pascal Molli. **Wiki sémantique sur réseau pair- à-pair.** *In IC2.0, Workshop in assocation with the 19èmes journées Francophone d'Ingéniérie des Connaissances*, Nancy, France, June 2008.

La contribution sur DSMW a donné lieu à la publication suivante :

- Charbel Rahhal, Hala Skaf-Molli and Pascal Molli. **Multi-synchronous Collaborative Semantic Wikis.** *In the 10th International Conference on Web Information Systems Engineering (WISE 2009).* Poznan, Poland, October 5-7, 2009.

J'ai également travaillé sur d'autres problématiques connexes qui ne sont pas présentées dans cette thèse qui ont donné lieu aux publications suivantes :

- Hala Naja-Jazzar, Nishadi De Silva, Hala Skaf-Molli, Charbel Rahhal and Pascal Molli. **OntoRest : A RST-based Ontology for Enhancing Documents**

Content Quality in Collaborative Writing. *INFOCOMP journal of computer science, volume 8, number 3, 2009.*

– Hala Skaf-Molli, Hala Naja-Jazzar, Charbel Rahhal and Pascal Molli. **Collaborative Writing of XML Documents.** *IEEE 3nd International Conference on Information and Communication Technologies : From Theory to Applications (ICTTA'08)*, Damascus, Syria, April 7-11, 2008.

– Charbel Rahhal, Hala Skaf-Molli, Pascal Molli, and Nishadi De Silva. **SemCW : Semantic collaborative writing using RST.** *In The 3rd International Conference on Collaborative Computing : Networking, Applications and Worksharing - CollaborateCom 2007.* New York, USA, November 2007.

– Hala Skaf-Molli, Claudia-Lavinia Ignat, Charbel Rahhal and Pascal Molli. **New Work Modes for Collaborative Writing.** *In International Conference on Enterprise Infor- mation Systems and Web Technologies - EISWT 2007.* Orlando, Florida , july 2007.

Chapitre 1

Contexte et motivations

Sommaire

1.1 Systèmes d'édition collaboratifs

L'édition collaborative est devenue de plus en plus commune dans le travail académique et dans les entreprises. La majorité des travaux écrits sont produits en collaboration [TMGS97], l'écriture des revues, des manuels techniques et la planification des présentations constituent quelques exemples d'activités collaboratives communes. L'édition collaborative est définie [LCL04] comme un processus de deux ou plusieurs personnes qui travaillent ensemble dans la production d'un document complexe. Les avantages de l'édition collaborative incluent la réduction du temps de terminaison d'une tâche, la réduction des erreurs, l'obtention des différents points de vue et compétences ainsi que l'obtention d'un texte plus précis [NR04, TMGS97]. La nature de la collaboration varie fortement [TMGS97] selon les stratégies d'édition du groupe, les relations et les rôles des membres du groupe et la proximité et la synchronicité des activités collaboratives. Les éditeurs collaboratifs permettent à plusieurs utilisateurs d'éditer de manière collaborative des documents à travers un réseau informatique. Un éditeur collaboratif permet l'édition simultanée d'un même document complexe par plusieurs utilisateurs distribués dans l'espace, le temps et à travers les organisations. Par la suite, nous détaillons trois méthodes de classification des outils coopératifs.

1.1.1 Matrice CSCW

La matrice d'Ellis [Joh88] illustrée à la figure 1.1 est souvent utilisée pour classifier les outils d'édition collaboratifs en se reposant sur deux caractéristiques : l'espace et le temps. L'axe de l'espace considère la distance spatiale entre les utilisateurs, alors que l'axe du temps considère la distance temporelle.

- L'axe espace se divise comme suit :
  1. même lieu : correspond au cas où les membres du groupe sont co-localisés au moment de l'utilisation du système collaboratif,
  2. lieux différents : correspond au cas où les membres du groupe sont distribués géographiquement dans l'espace.
- L'axe temps est divisé comme suit :
  1. même moment, ou synchrone : correspond au cas où les membres du groupe utilisent simultanément le système collaboratif.
  2. moments différents, ou asynchrone : correspond au cas où les membres du groupe utilisent le système collaboratif à des moments différents.

Cette matrice permet de classifier les outils de collaboration selon les modes d'interaction qu'ils proposent : co-localisés ou à distance, synchrone ou asynchrone. Quatre régions distinctes sont représentées dans la matrice d'Ellis :

1. Tout d'abord celle issue du croisement "même moment" et "même lieu", dans laquelle se trouvent les systèmes informatiques qui supportent le travail en mode face-à-face

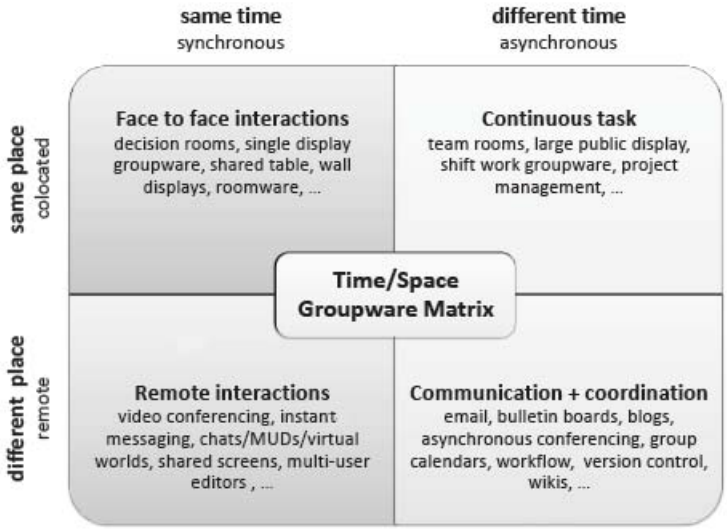

FIGURE 1.1 – La matrice d'Ellis [SMIRM07]

(aussi appelée "meeting facilities"), tels que RoomWare [PST04]. L'idée est d'équiper l'environnement d'une salle de réunion (murs, tables) d'outils interactifs favorisant le travail de groupe.

2. Dans la deuxième région, "même lieu" et "différents moments", apparaissent les interactions asynchrones sur un même lieu. Laurillau et al. [LN02] illustre cette catégorie avec la prise de parole au cours d'une présentation où l'enchaînement des actions est planifié et elles se déroulent dans un même lieu. [MYS05] proposent un outil pour le contrôle interactif des applications dans une salle équipée de plusieurs ordinateurs branchés sur des projecteurs. L'idée est de permettre aux participants dans une telle salle de contrôler, à tour de rôle, les applications qui sont projetées dans la salle.

3. Les autres régions sont caractérisées par une distribution géographique, les utilisateurs se trouvant dans des lieux différents (dans des bureaux distincts, dans différents lieux d'une même ville, ou même dans des villes, pays ou continents différents). La troisième région, "même moment" et "différents lieux" correspond aux systèmes informatiques qui supportent les interactions synchrones distribuées. Dans ce type de systèmes, les utilisateurs sont connectés au même moment [LN02], l'interaction a lieu en temps réel mais les utilisateurs sont dispersés. Les Mediaspaces [DB92, BHI93] et les messageries instantanées sont des systèmes relevant typiquement de cette catégorie. De nombreux outils d'édition de documents fonctionnant en mode synchrone (ou "temps réel") ont également été proposés, comme Grove [EGR91], rIBIS [RE91], DistEdit [KP90] et GroupKit [RG96].

4. La dernière région "différents lieux" et "différents moments" concerne les systèmes qui supportent des interactions asynchrones et distribuées, dans lesquels les utilisateurs n'ont pas besoin d'être connectés simultanément pour interagir. L'interaction peut se produire en temps différé entre des utilisateurs distribués géographiquement. Il existe plusieurs outils coopératifs dans cette catégorie : (1) des outils de coordination tels que les workflow, les agendas et les calendriers partagés (2) des outils de communication tels que le courrier électronique classique et les forums en ligne et (3) des outils du web social tels que marque-pages partagés, géo-référencement, wikis et blogs.

Dans cette thèse, nous nous intéressons aux éditeurs collaboratifs qui appartiennent à la quatrième région, ceux qui permettent aux utilisateurs dispersés dans l'espace de collaborer à des moments différents.

Interaction multi-synchrone

Il existe un autre mode d'interaction collaborative intéressant appelé *multi-synchrone* qui ne figure pas dans la matrice d'Ellis. Le mode d'interaction multi-synchrone illustré dans la figure 1.2 est défini par Dourish [Dou95], il supporte la notion de flux multiples et parallèles d'activités. Dans ce mode, les participants travaillent indépendamment et en parallèle. La divergence se produit durant le travail parallèle des participants. Périodiquement leur effort individuel sera intégré afin d'aboutir à un état cohérent et de faire progresser l'activité du groupe. Cette intégration est appelée synchronisation. Dans ce mode, le travail collaboratif progresse au moyen des phases de divergence et de synchronisation. Concrètement, deux personnes peuvent travailler au même moment mais sur des copies différentes.

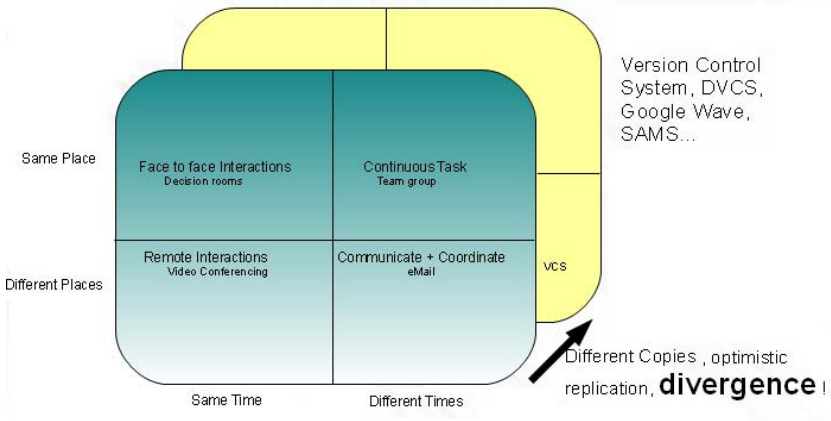

FIGURE 1.2 – Mode d'édition multi-synchrone

Les applications multi-synchrones diffèrent de celles synchrones et asynchrones par leur gestion des flux multiples d'activités plutôt que donner l'illusion d'un seul flux. Elles per-

mettent aux utilisateurs de synchroniser les données soit d'une façon immédiate après avoir généré des changements, soit plus tard lorsqu'ils le souhaitent. Permettre une propagation reportée des changements et la divergence offrent plusieurs avantages [MSMO02] : la parallélisation des activités, le travail en isolation, l'exécution des changements partiels et expérimentaux avant de les publier. Les systèmes de contrôle de versions distribués (DVCS) tels que Git [git] et Mercurial [Mer] sont des systèmes collaboratifs multi-synchrones.

1.1.2 Taxonomie de Dix

Le modèle du travail coopératif [DFAB97] propose une classification des outils collaboratifs en se basant sur leur utilisation. Ce modèle illustré dans la figure 1.3 identifie les relations entre les participants impliqués dans le travail collaboratif et les relations entre les participants et les outils "artefacts" qu'ils utilisent. Ces relations sont les suivantes :

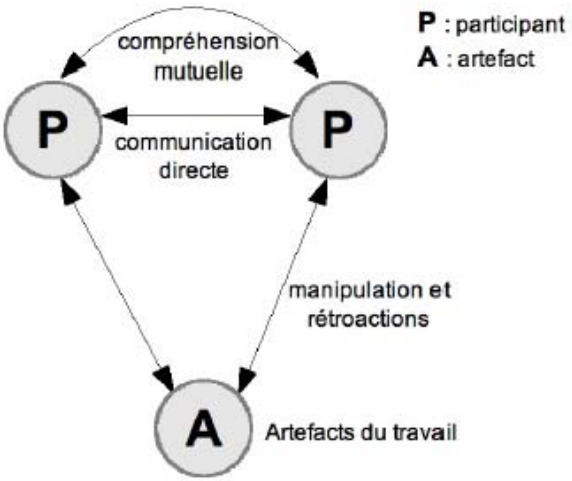

FIGURE 1.3 – Modèle du travail coopératif

- La communication directe entre les participants au cours d'une activité se produit en utilisant des outils tels que courrier électronique, audio ou vidéo.
- L'établissement d'une compréhension mutuelle afin de réaliser des actions conjointes.
- La manipulation des artefacts. La rétroaction est le résultat de la manipulation perçue par les autres utilisateurs.

Pour établir une compréhension mutuelle entre eux, les participants doivent communiquer, s'organiser et coordonner leurs activités. Le concept de comprehension mutuelle rejoint le concept de conscience de groupe introduit par [DB92][Dourish et Belloti] :

la compréhension des activités des autres fournit le contexte à ses propres activités.

A partir du modèle du travail coopératif, [DFAB97] propose une taxonomie relative aux trois relations identifiées dans ce modèle :

- La première catégorie regroupe les systèmes qui offrent une communication directe entre les participants. Cette catégorie inclut le courrier électronique, les forums de discussion, la visioconférence et les mediaspaces.
- La deuxième catégorie regroupe les systèmes de réunions et d'aide à la décision, des systèmes qui visent à aider la compréhension mutuelle afin de faciliter la prise de décisions entre les différents participants. Cette catégorie contient les systèmes d'aide à la décision (GDSS), les systèmes de réunion virtuelle (bureau partagé) et réelle (tableau blanc réel et partagé).
- La troisième catégorie regroupe les systèmes d'espaces partagés dans lesquels les participants manipulent des artefacts dans des espaces partagés. Cette catégorie inclut les éditeurs partagés et des calendriers partagés.

1.1.3 Modèle du trèfle

Un système collaboratif est caractérisé selon le modèle du trèfle [Sal95] par trois dimensions (voir figure 1.4) :

1. **L'espace de coordination** définit à la fois les acteurs (individus, rôles, activités) et les relations entre eux. La coordination dans un système collaboratif peut être de nature différente :
   - elle peut être informelle (non vérifiée automatiquement par un outil) comme avec un chef de projet et des plannings. Les procédés sont définis de manière informelle et vérifiés par des personnes.
   - elle peut être définie de manière formelle comme avec des outils de workflow. Dans ce cas, les procédés sont définis de manière formelle, vérifiés et exécutés par des machines.
2. **L'espace de communication** permet l'échange de l'information entre les acteurs. Il se sert des outils coopératifs tels que la messagerie instantanée, le courrier électronique et la visioconférence.
3. **L'espace de production** correspond aux objets produits ou manipulés lors d'une activité de groupe. Par exemple, ces objets peuvent être un livre, un logiciel, un film, etc. Cette notion s'applique aussi bien aux objets produits dans l'espace commun aux membres du groupe, qu'aux objets produits dans l'espace privé de chacun des membres du groupe. L'espace de production doit permettre à tout utilisateur d'annuler les changements faits sur ces objets. Par la suite, nous présentons l'annulation de groupe.

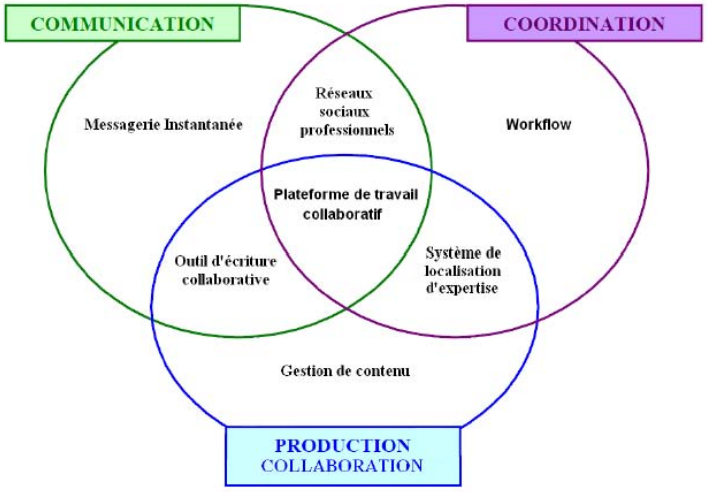

FIGURE 1.4 – Le modèle du trèfle des systèmes collaboratifs extrait de [gui]

Mécanisme d'annulation de groupe

L'annulation est une fonctionnalité très utile supportée largement dans les systèmes collaboratifs [CS01]. Elle permet à un utilisateur de corriger ses erreurs, de visionner des modifications précédentes et de traiter les actions de vandalisme. Le mécanisme d'annulation est reconnu comme une fonctionnalité importante dans les systèmes d'édition collaborative [AD91, CD95]. Dans le cadre d'une application non-répartie mono-utilisateur, les différents modes d'annulation qui peuvent être proposés sont les suivants : (1) annulation de la dernière opération uniquement, (2) annulation suivant l'ordre chronologique inverse et (3) annulation libre sélective [Vid02]. Le mécanisme d'annulation de groupe fait la transposition de la fonctionnalité d'annulation dans les éditeurs collaboratifs. Contrairement aux applications mono-utilisateur, dans lesquelles les opérations sont indifférenciées et traitées de façon égale, dans un éditeur collaboratif partagé, les opérations exécutées peuvent être distinguées suivant l'utilisateur qui les a générées. Un utilisateur peut annuler les opérations des autres utilisateurs. Il peut avoir plusieurs utilisateurs qui annulent en concurrence les mêmes opérations. Un utilisateur peut annuler l'annulation d'une opération faite par un autre utilisateur. Durant l'exécution de l'annulation de groupe par un utilisateur, les autres utilisateurs peuvent continuer à apporter des modifications au document en concurrence. Le modèle général d'un mécanisme d'annulation de groupe permet à tout utilisateur d'annuler à tout moment n'importe quelle opération d'édition. Nous présentons la définition de l'annulation de groupe comme définie dans [Sun02] :

Définition 1. *Annuler une modification fait retourner le système à l'état qu'il aurait atteint si cette modification n'a jamais été produite.*

Un mécanisme d'annulation de groupe intégré dans un éditeur collaboratif doit res-

pecter la définition de l'annulation et les propriétés du système collaboratif [Sun02, Sun00].

Le modèle du trèfle, divisant en trois espaces l'ensemble des sous-domaines que regroupent les systèmes collaboratifs, peut être interprété de manière fusionnelle ou de manière successive. La manière fusionnelle correspond au cas où le système collaboratif utilise au même moment au moins deux espaces. La manière successive, quant à elle, correspond au cas où le système collaboratif utilise de manière non simultanée plusieurs espaces du modèle.

1.2 Systèmes d'édition wiki

Les dix dernières années ont vu un développement sans précédent des outils collaboratifs. Le plus célèbre d'entre eux est certainement le WikiWikiWeb [Cun95] ou Wiki.

1.2.1 Wikis

Les wikiwikiwebs [Cun95] ou wikis constituent une famille d'éditeurs collaboratifs très populaires, le plus populaire des wikis étant Wikipedia [1]. Les wikiwikiwebs ont permis de transformer le web en un espace collaboratif [DPV05]. En effet, une page wiki peut être modifiée facilement en cliquant sur le bouton "modifier" (voir figure 1.5) depuis son navigateur web. Les wikis sont aussi des outils sociaux dans le sens où ils permettent de transformer une communauté d'étrangers en une communauté de collaborateurs.

Selon la matrice d'Ellis, les wikis sont des outils collaboratifs distribués dans le temps et dans l'espace. Ils appartiennent donc à la classe des outils asynchrones. Les wikis offrent les trois dimensions du modèle de trèfle de la manière suivante : ils fournissent des pages de discussion pour les espaces de coordination et de communication, tandis que les articles sont les résultats de l'espace de production. Ils fournissent un mécanisme d'annulation permettant d'annuler les modifications faites sur les pages wikis. Les wikis appartiennent à la catégorie des espaces partagés de la taxonomie de Dix. La conscience de groupe dans les wikis est assurée via les pages "Historique" et les pages de suivi. Les wikis ont permis à des millions d'utilisateurs de produire des pages wikis traitant des domaines les plus variés. Une expérience d'une telle ampleur est sans précédent. Le succès le plus significatif des wikis est sans aucun doute la Wikipedia. Malgré leur succès, les wikis souffrent de certaines limitations dans la recherche et dans la navigation [KVV+07]. En effet, il n'est pas facile de trouver une information dans les wikis [BGE+08]. Les wikis sémantiques ont été développés pour surmonter ces limitations.

1. http ://www.wikipedia.org

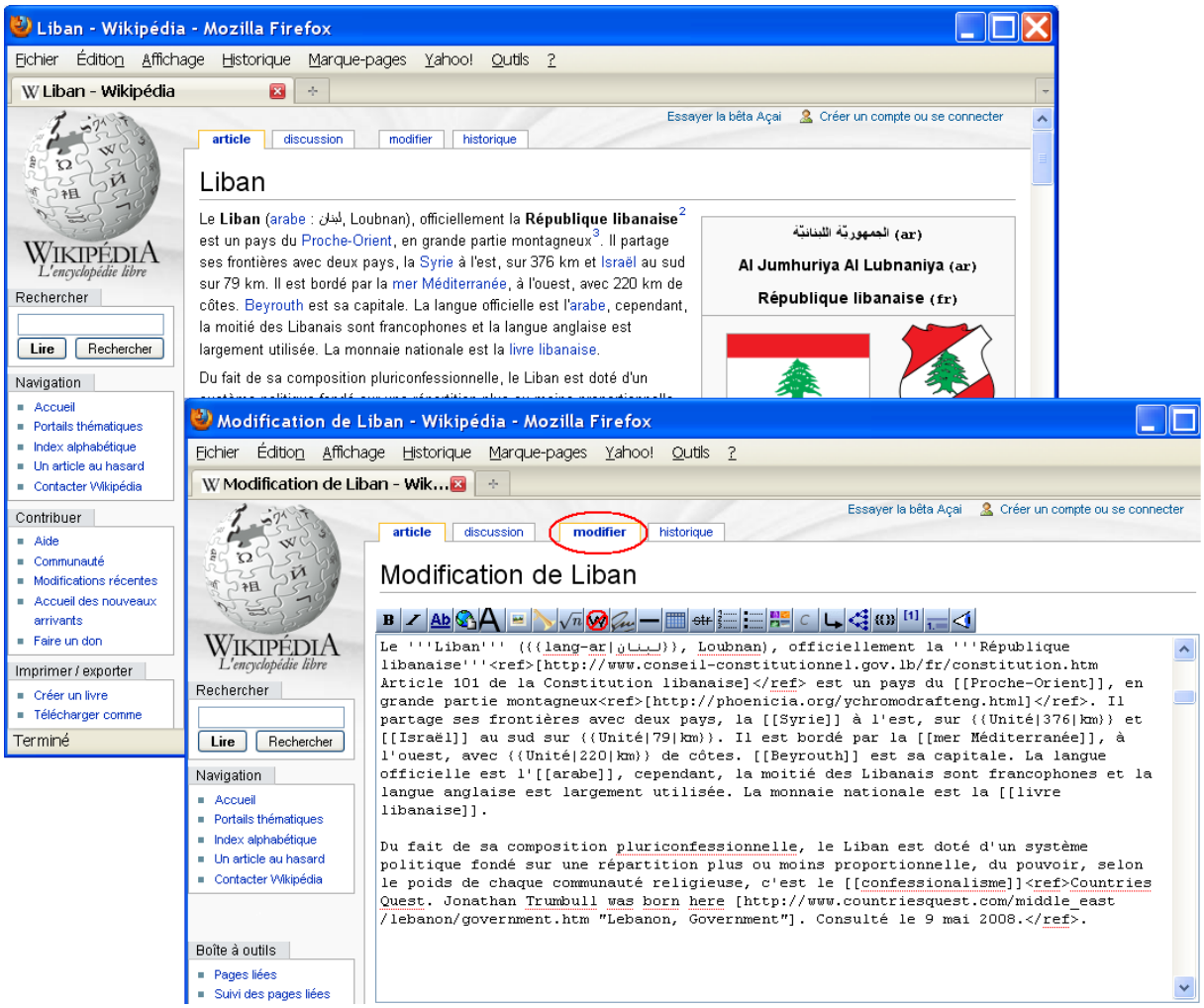

FIGURE 1.5 – Edition d'une page wiki

1.2.2 Wikis sémantiques

Les wikis sémantiques sont une nouvelle génération d'outils d'édition collaborative. Ils intègrent les technologies du web sémantique [SBBK08] dans les wikis. Cette intégration permet de représenter des métadonnées sur les pages et les relations entre elles sous formes d'annotations sémantiques. Les wikis sémantiques traitent donc un nouveau type de données constitué des pages wikis et des annotations sémantiques. Les utilisateurs peuvent alors collaborer non seulement dans l'écriture des pages wikis mais aussi dans l'annotation de ces pages. Les wikis sémantiques se distinguent par la façon d'éditer les données sémantiques (à l'intérieur ou à l'extérieur des pages wikis), de les stocker (dans une base de données ou dans un entrepôt spécifique), de les afficher (au moyen d'un calendrier, d'une carte ou dans la page wiki) et de les formaliser (en utilisant des balises simples, le modèle RDF [RDF04]/RDFS [RDF] et le vocabulaire OWL [OWL]).

De nombreux wikis sémantiques sont actuellement développés tels que Semantic MediaWiki [VKV+06], SweetWiki [BGE+08], IkeWiki [Sch06] et OntoWiki [AD+06]. [KSV07] propose une classification des wikis sémantiques qui distingue deux approches :

Wikis pour des données sémantiques Les wikis sémantiques appartenant à cette approche peuvent être utilisés comme des éditeurs collaboratifs d'ontologies. Ils supportent une édition efficace de l'information ontologique. Ce qui permet le développement d'une ontologie formelle en utilisant parfois le contenu textuel des pages wikis dans la création d'une spécification informelle. Des wikis sémantiques appartenant à cette approche sont IkeWiki [Sch06] et OntoWiki [AD+06].

Données sémantiques pour les wikis Dans cette approche, les wikis sémantiques utilisent les données sémantiques soit pour rendre une partie du contenu du wiki exploitable par des machines, soit pour simplifier la maintenance du wiki en exploitant ces métadonnées. Ce qui permet de simplifier l'extraction de l'information. Semantic MediaWiki [VKV+06] et SweetWiki [BGE+08] sont des wikis sémantiques appartenant à cette approche.

En plus des avantages des wikis traditionnels, les wikis sémantiques offrent :

Amélioration dans la recherche et la navigation Les wikis sémantiques fournissent une meilleure structuration des wikis en offrant un moyen de naviguer et de rechercher les pages wikis en se basant sur les annotations qu'elles contiennent. La recherche et la navigation deviennent 'intelligentes' basées sur la sémantique et non seulement sur les mots clés.

Interopérabilité La structuration du contenu des wikis sémantiques rend ce contenu accessible et réutilisable par d'autres applications [BGE+08, SBBK08]. L'échange des données avec d'autres systèmes dans les deux sens peut avoir lieu.

Collaboration Homme-Machine Le contenu des wikis sémantiques est exploitable par les utilisateurs et les machines à la fois. Une interaction homme-machine dans le même espace de travail donne naissance à des fonctionnalités avancées telles que la mise à jour dynamique, la vérification du contenu, la notification des changements [BGE+08] et le raisonnement [BCC+09].

Dans la suite de cette section, nous présentons le fonctionnement de deux wikis sémantiques dont chacun appartient à une approche différente.

Semantic MediaWiki

Semantic Media (SMW) [KVV+07] est un wiki sémantique appartenant à l'approche des *données sémantiques pour les wikis*. Il est une extension de MediaWiki conçu pour aider à la recherche, à organiser, à annoter, à naviguer et à partager le contenu du wiki. SMW permet l'ajout des annotations sémantiques dans les pages wikis afin de profiter de la puissance du web sémantique dans le wiki.

Annotations des pages Dans SMW, les annotations sémantiques sont intégrées directement dans le texte *via* un langage de balisage spécial illustré à la figure 1.6. Lors de la sauvegarde d'une page, ses annotations sont extraites et mettent à jour l'entrepôt des annotations. Elles sont stockées sous forme de triplets RDF. Chaque article représente un sujet. Une annotation dans un article est un fait sur ce sujet. Elle représente ses propriétés et spécifie ses relations avec d'autres pages. La classification des pages est faite manuellement par les utilisateurs en utilisant l'annotation *Catégorie*. La localisation des annotations dans le texte assure une meilleure lisibilité des annotations. Les différentes formes d'annotations possibles dans SMW sont :

FIGURE 1.6 – Une page wiki sémantique dans Semantic MediaWiki

Catégories sont une forme d'annotation simple qui permet aux utilisateurs de classifier les pages. Les catégories sont déjà disponibles dans MediaWiki et ont une interprétation formelle comme les classes OWL. Pour indiquer que la page France appartient à la catégorie des pays, il faut écrire [[Category :pays]] à l'intérieur de cette page.

Relations décrivent les relations entre deux pages en attribuant des annotations aux liens existants. Par exemple, il existe une relation entre la page France et la page Union Européenne. Pour exprimer cela, il faut changer le lien [[Union Européenne]] à [[membre : :Union Européenne]].

Attributs permettent aux utilisateurs de spécifier des relations d'articles à des choses qui ne sont pas des articles. Par exemple, on peut déclarer que la population de la France est égale à 65 073 482 en écrivant [[population := 65 073 482]].

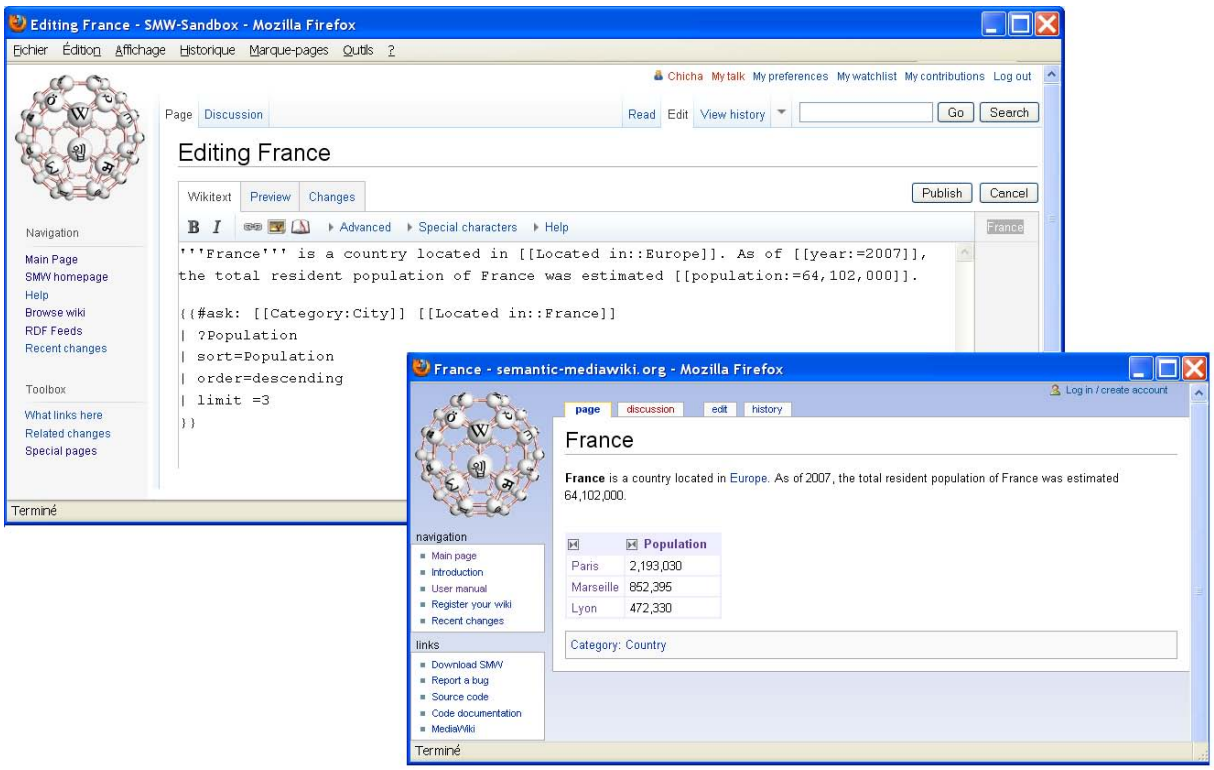

FIGURE 1.7 – Requête imbriquée dans SMW

Navigation, recherche et requêtes sémantiques Dans SMW, les annotations des pages peuvent être utilisées soit pour naviguer entre les pages qui sont sémantiquement dépendantes, soit pour rechercher les pages de SMW qui ont une certaine propriété ou attribut. SMW définit un langage de requêtes simple pour accéder à la connaissance du wiki qui a une syntaxe similaire à celle des annotations. Le langage de requêtes peut être utilisé de deux manières : soit interroger directement le wiki, soit imbriquer des requêtes appelées *inline* dans le contenu des pages wikis. L'écriture d'une requête *inline* pour chercher les trois villes françaises les plus peuplées et les afficher par l'ordre décroissant de leur population est illustrée dans la figure 1.7.

Dans SMW, il est souhaité de spécifier explicitement un type de données pour chaque propriété pour éviter toute confusion. Par défaut, le type d'une propriété est Type :Page.

SMW définit plusieurs types de données tels que Type :Page, Type :String, Type :Number qui peuvent être attribués aux propriétés. Par exemple, pour spécifier que la propriété *Car* est de type chaîne de caractères (voir figure 1.8), il suffit d'écrire [[has type : :Type :String]].

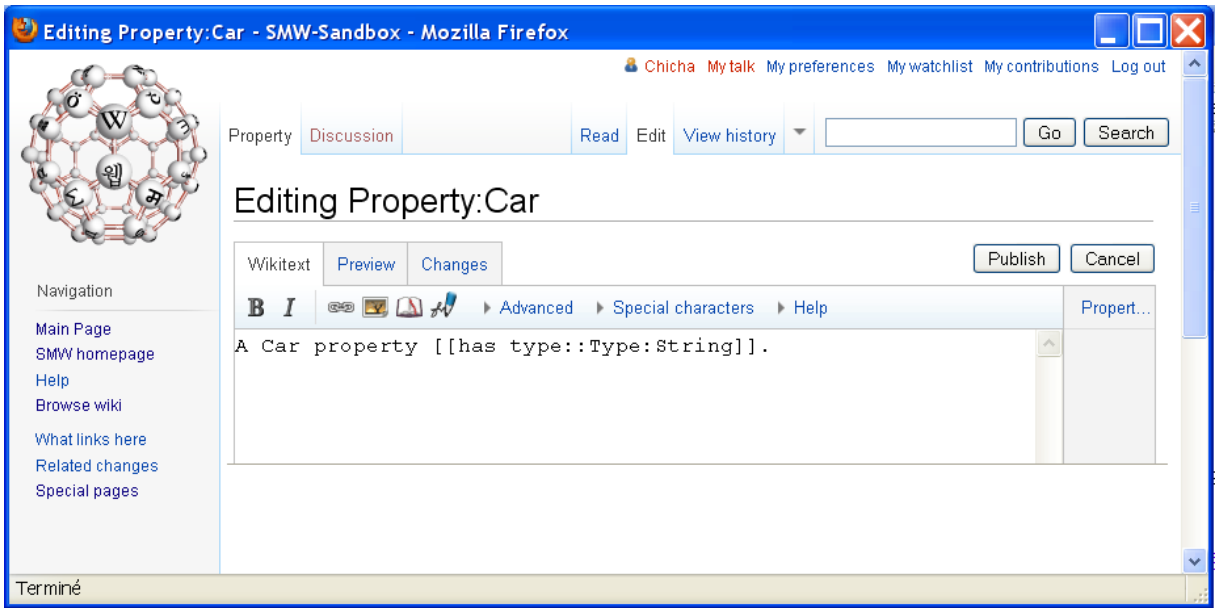

FIGURE 1.8 – Définir un type pour une propriété Car dans SMW

Correspondance avec le vocabulaire OWL Dans SMW, les pages peuvent être naturellement reliées au vocabulaire de base de OWL. SMW emploie différents espaces de nommage pour distinguer le type des pages et ce mécanisme de typage de pages est décrit comme suit :

- *Individus OWL* sont représentés par les pages d'articles normales. Ces pages constituent généralement la majorité du contenu du wiki, elles sont contenues dans l'espace de nom principal de MediaWiki.
- *Classes OWL* sont représentées par les catégories de MediaWiki. Elles sont les pages qui appartiennent à l'espace de nom Category. Elles peuvent être organisées d'une façon hiérarchique mais une catégorie ne peut pas contenir autres catégories. Le système de catégorie de Wikipedia est proche des classes de OWL DL [HPSMW07].
- *Propriétés OWL* ont été introduites par SMW. OWL distingue entre les propriétés d'objets (décrivant les relations entre deux individus) et les propriétés des données (associant aux individus des valeurs d'un type de données). Une même distinction est faite dans SMW, les propriétés d'objets sont représentées par des pages de l'espace de noms Relations, alors que les propriétés des données sont représentées par des pages appartenant à l'espace de noms Attributs.

OWL	**Semantic MediaWiki**
OWL individual	normal article page
owl:Class	article in namespace Category
owl:ObjectProperty	article in namespace Relation
owl:DataTypeProperty	article in namespace Attribute
Statement about element *page*	**Syntax in wiki-source of *page***
object-property	[[property_name::object_article]]
attribute-property	[[property_name:=value_string]]
rdf:type *class_name*	[[Category:*class_name*]] (on article pages)
rdfs:subClassOf *class_name*	[[Category:*class_name*]] (on category pages)

TABLE 1.1 – Représentation des éléments ontologiques dans Semantic MediaWiki[sem06]

En se basant sur la correspondance entre les données et OWL, SMW permet aux utilisateurs de décrire différents assertions ontologiques dans le wiki. Un aperçu des éléments OWL qui peuvent être représentés dans le wiki est donné dans le tableau 1.1.

OntoWiki

OntoWiki [AD+06] est un wiki appartenant à l'approche des wikis sémantiques utilisant *les wikis pour des données sémantiques*. OntoWiki est un outil qui supporte l'ingénierie collaborative des connaissances dans un environnement web. OntoWiki permet de créer facilement des bases de connaissance. Dans OntoWiki, les utilisateurs sont capables de créer, de supprimer et de modifier des ressources, des propriétés et des instances de ressources dans les différentes bases de connaissance. Par exemple, l'édition des propriétés d'une ressource est illustrée dans la figure 1.9. OntoWiki fournit deux vues génériques pour éditer et afficher les ontologies : (1) une vue de la ressource et (2) une vue de liste. La première est utilisée pour afficher toutes les informations concernant une ressource, la deuxième représente un ensemble de ressources, généralement, les instances d'un certain concept. Ces vues sont illustrées dans la figure 1.10.

Navigation, recherche et requêtes Dans OntoWiki, chaque vue est représentée par une page wiki. Elle contient plusieurs cadres qui affichent chacun des informations reliant cette page. Par exemple, dans le cas de la vue d'une ressource, les ressources similaires ou distinctes ainsi que les ressources ayant comme domaine, rang ou type cette ressource sont affichées dans les cadres de la vue (voir figure 1.10). OntoWiki permet d'utiliser les ressources contenues dans une vue pour naviguer d'une ressource à une autre. Il fournit également une interface de navigation qui permet de filtrer la recherche selon le type de la ressource que ce soit une classe ou une propriété. OntoWiki permet d'exécuter des requêtes SPARQL [spa07] de deux manières, soit en utilisant un éditeur de requêtes SPARQL classique, soit en utilisant une interface de requêtes graphiques (voir figure1.11)

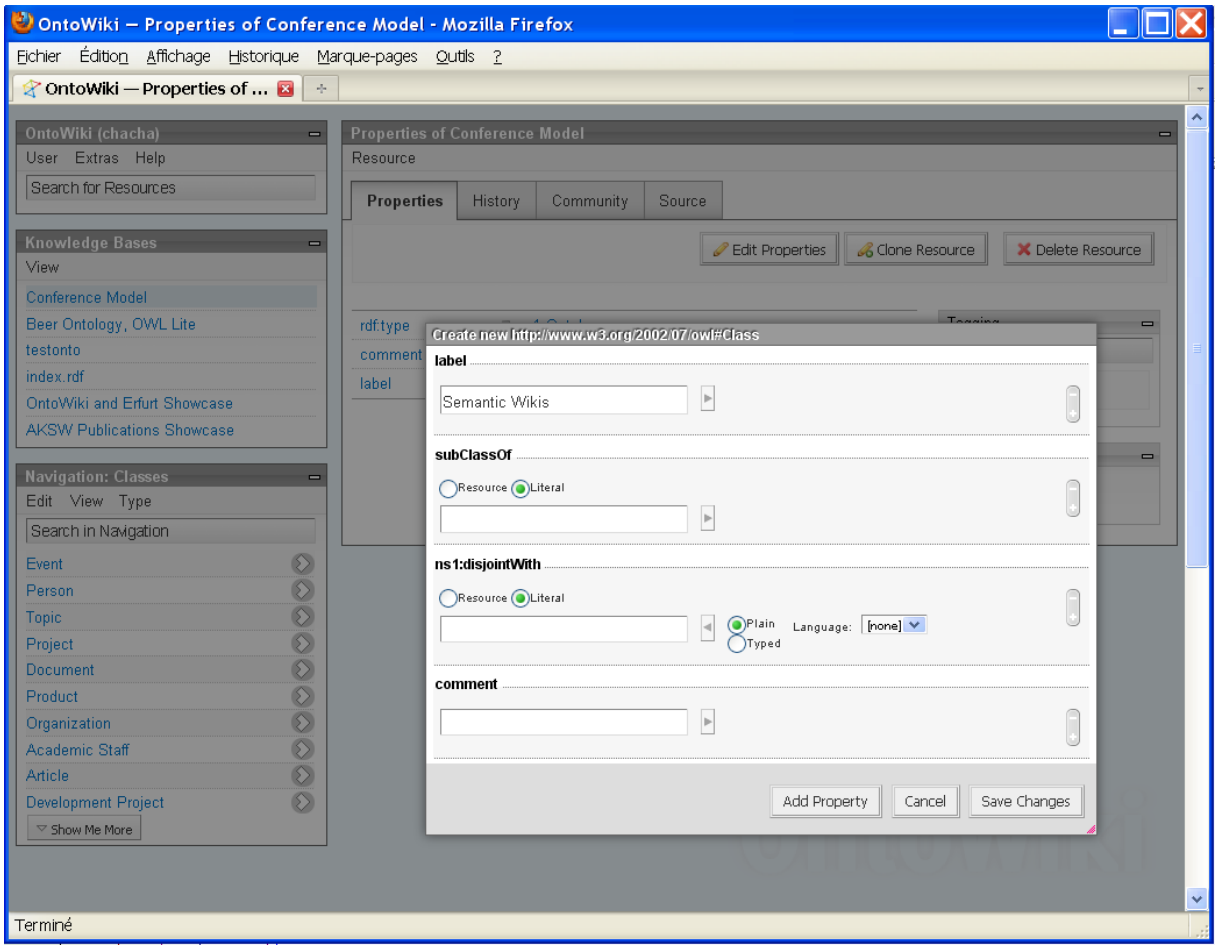

FIGURE 1.9 – Edition d'une ressource dans OntoWiki

dans laquelle les ressources peuvent être glissées-déposées (drag and drop). Chaque requête peut être sauvegardée et réutilisée ultérieurement.

Mécanisme d'annulation Dans OntoWiki, chaque page wiki est associée à une historique qui permet de suivre les changements effectués sur cette ressource. Les derniers changements d'une page peuvent être affichés dans un cadre sur la page wiki. Tout changement d'une base de connaissance, d'une classe, d'une propriété ou d'une instance est suivi et stocké dans un journal. Les changements dans l'historique d'une page sont : l'ajout, la suppression ou la mise à jour des propriétés. Chacun de ces changements peut être annulé sélectivement par un roll-back.

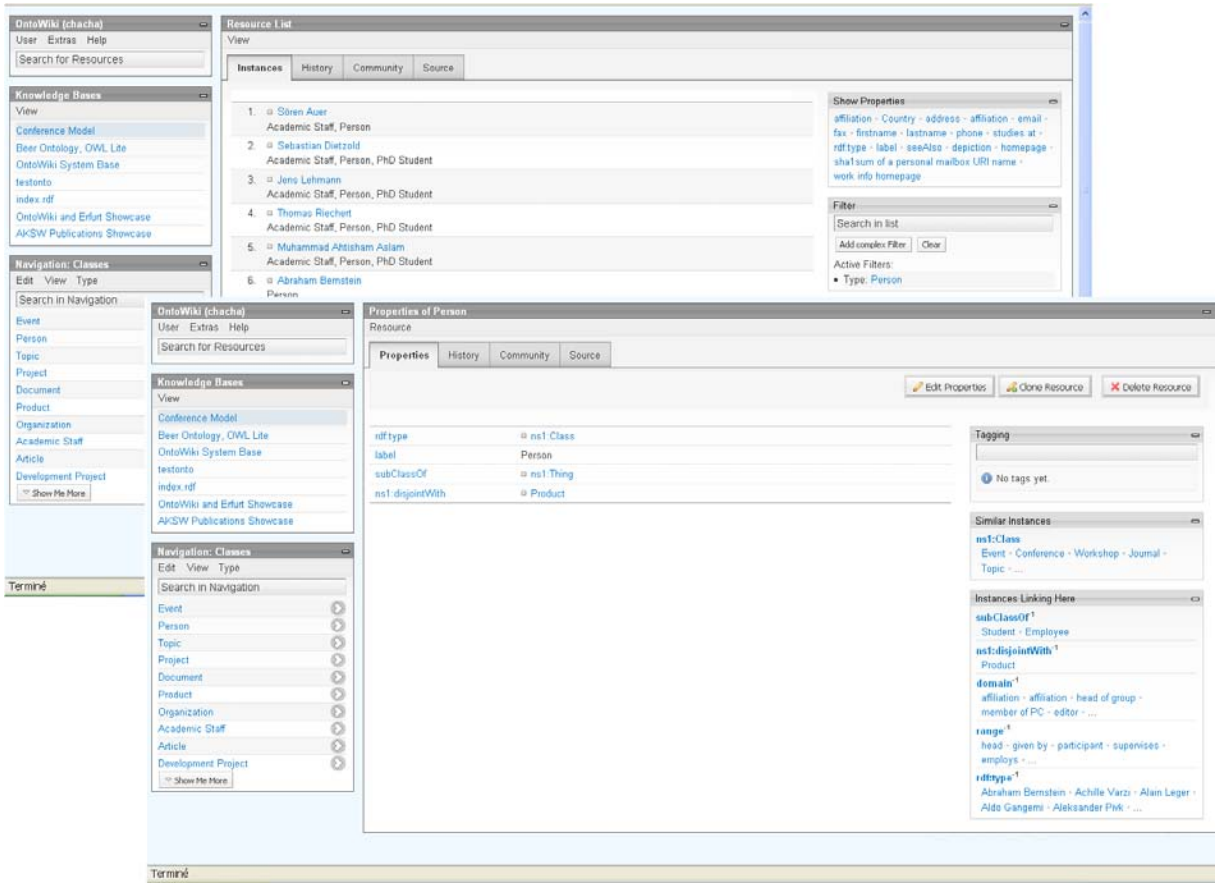

FIGURE 1.10 – Les vues d'une ressource 'Person' et de la liste de ses intances dans OntoWiki

1.3 Motivations

Dans cette section, nous présentons les motivations de notre travail. Nous présentons les problèmes de passage à l'échelle et la tolérance aux pannes et les problèmes de coordination.

1.3.1 Passage à l'échelle et tolérance aux pannes

De nos jours, les systèmes collaboratifs sont utilisés massivement. Par exemple, Wikipedia reçoit 60 000 demandes de pages par seconde. Cela pose des problèmes de passage à l'échelle et de tolérance aux pannes. Les wikis sont des systèmes distribués basés sur une architecture client-serveur. De manière intrinsèque, cette architecture supporte mal le passage à l'échelle et la tolérance aux pannes. Cependant, force est de constater que Wikipedia supporte plutôt bien la collaboration de masse. Wikipedia fonctionne grâçe à un logiciel open source MediaWiki. Si MediaWiki ne passe pas à l'échelle, Wikipedia passe à l'échelle. Comparé à MediaWiki, Wikipedia est un système distribué complexe utilisant

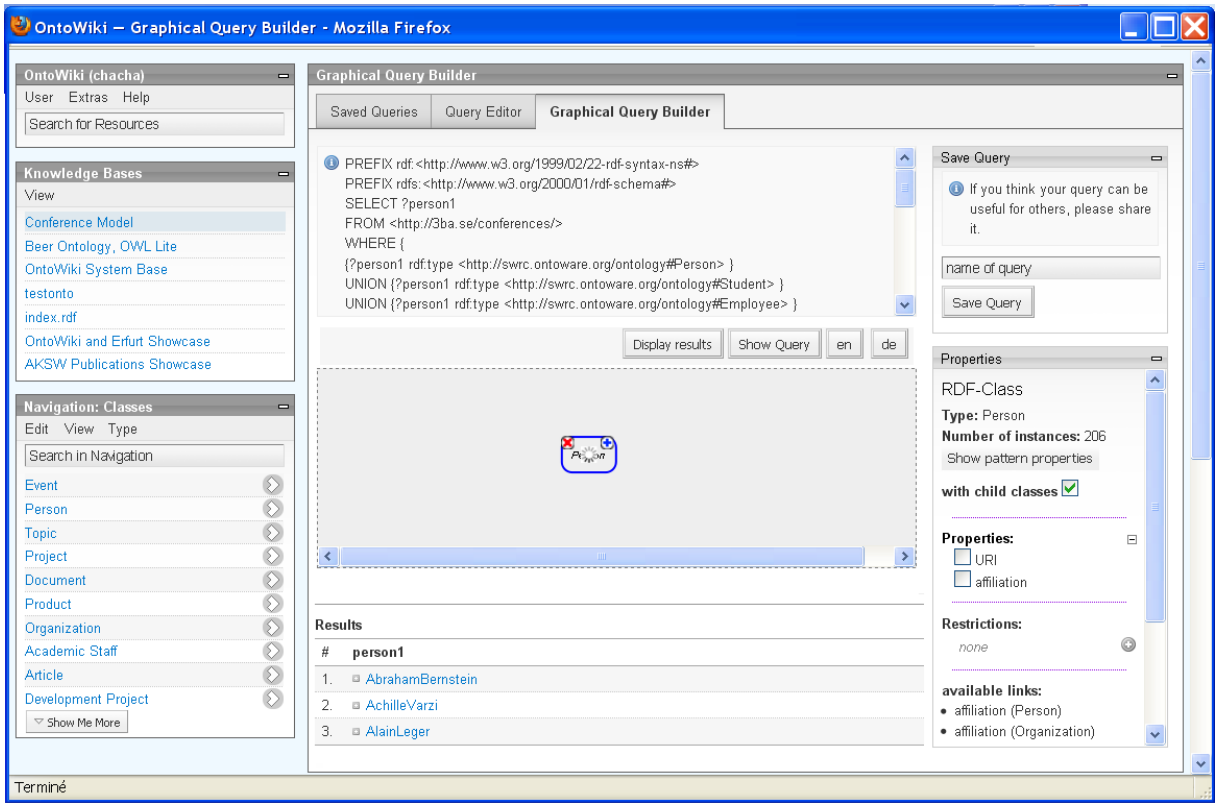

FIGURE 1.11 – L'interface graphique des requêtes dans OntoWiki

déjà massivement des techniques de réplication optimiste basées sur des caches. L'architecture actuelle de Wikipedia illustrée à la figure 1.12 comporte 370 serveurs [2] distribués sur trois continents.

La maintenance de cette architecture a un coût inhérent. Le rapport annuel de Wikipedia révèle une dépense de 3 370 200$ pour sa maintenance pour l'année 2008-2009. En dépit, du coût élevé de la maintenance, Wikipedia ne tolère pas complètement les pannes. Nous illustrons dans le tableau 1.13 les durées de pannes de Wikipedia par année.

Comme les wikis classiques, les wikis sémantiques souffrent des mêmes limitations dues à l'architecture centralisée telles que le passage à l'échelle et la disponibilité des données. Cependant, l'aspect sémantique dans les wikis sémantiques amplifie le problème de leur passage à l'échelle. Le traitement des annotations dans les wikis sémantiques a un surcoût sur leur performance. Par exemple, des mesures effectuées dans [KVV+07] montrent que pour un wiki sémantique de taille moyenne appelé 214365 fois, le temps occupé par la partie sémantique est de 6,8% du temps total dont 40% de ce temps est accordé aux requêtes sémantiques. Puisque le temps d'exécution des requêtes sémantiques est polyno-

2. http ://ganglia.wikimedia.org/

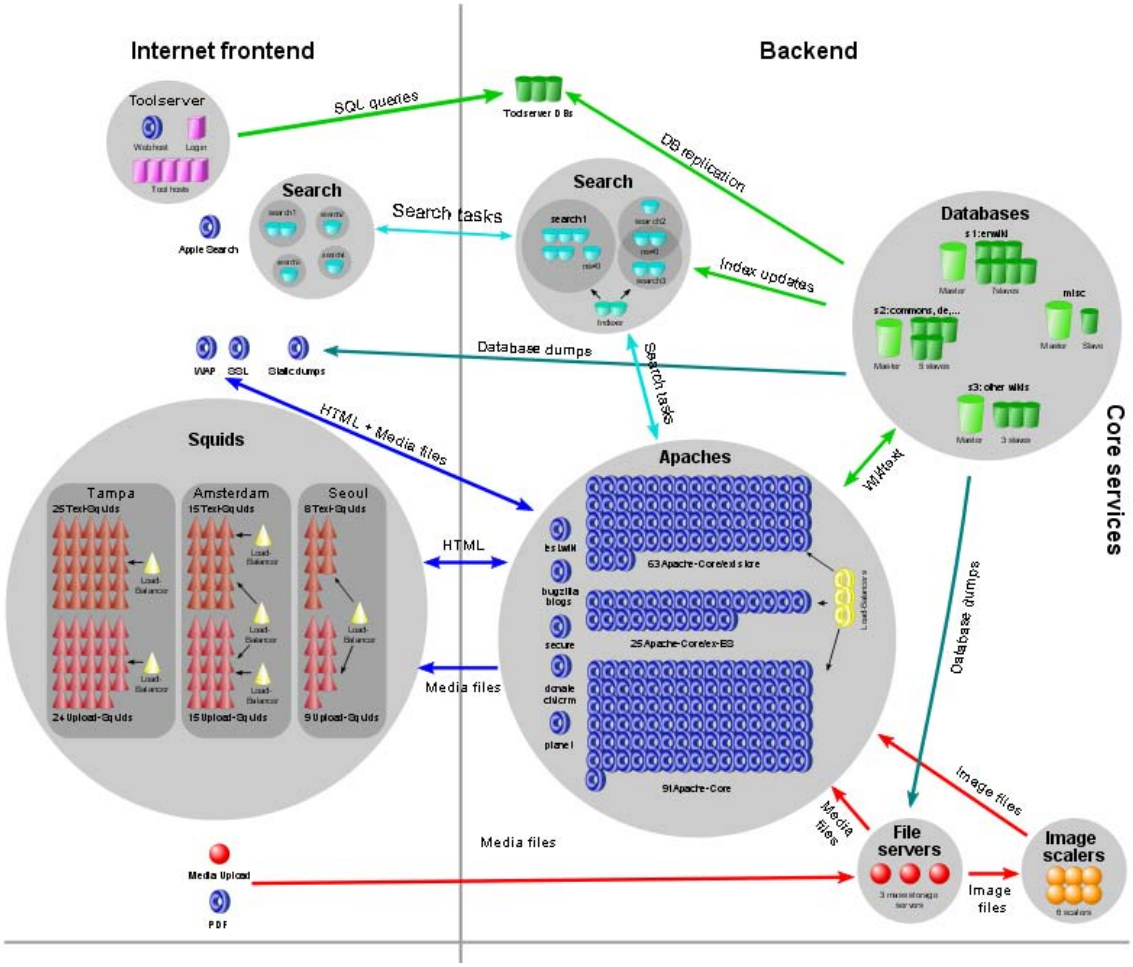

FIGURE 1.12 – L'architecture de Wikipedia

mial [KRH07], lorsque la quantité de données augmente la proportion du temps accordée aux données sémantiques va augmenter. Par conséquent, aucune garantie que l'architecture de Wikipedia présentée dans la figure 1.12 puisse supporter Semantic Wikipedia. En effet, le directeur adjoint de la fondation Wikimedia (WMF) Eric Möller qui est le responsable de la gestion et la mise en œuvre de la stratégie technique de l'organisation WMF déclare ceci [3] : "Wikipedia est confronté à des grands défis pour inclure les concepts sémantiques. Le premier défi est que personne n'a encore construit un service web sémantique à l'échelle d'un site tel que Wikipedia et il est difficile de savoir si les logiciels existants tels que Semantic MediaWiki sont à la hauteur de cette tâche". Pour conclure, le passage à l'échelle et la tolérance aux pannes dans les wikis sémantiques restent des problèmes ouverts et critiques.

3. http ://www.technologyreview.com/web/25728/page2/

Durée de panne	année
1h 13m	2010
5h 13m	2009
4h 1m	2008
20h 26m	2007
18h 56m	2006

FIGURE 1.13 – Durées de panne de Wikipedia extrait de [pin]

1.3.2 Procédés d'édition et édition multi-synchrone

Les wikis en tant qu'outil de gestion de connaissance ont besoin de maintenir la qualité de leur contenu. Par exemple, la qualité des pages de Wikipedia repose sur un procédé léger de contrôle des modifications qui est une forme de coordination [KK08]. Ce procédé peut être informel implémenté au moyen des pages de discussion, des balises ou de la conscience de groupe. Il peut être plus formalisé comme avec flaggedRevs [FlR07] implémenté dans Wikipedia allemande. FlaggedRevs est une extension du logiciel MediaWiki qui fournit un procédé de révision. Il permet d'associer à chaque révision d'une page un drapeau 'flag' indiquant sa qualité et sa validité et de contrôler les révisions visibles par défaut sur la page. FlaggedRevs définit trois échelles qui sont la "précision", la "profondeur" et la "lisibilité" d'un article dont chacun contient quatre drapeaux comme c'est illustré dans la figure 1.14. En plus, FlaggedRevs définit des marqueurs qui sont des regroupements plus abstraits de drapeaux. Ils identifient généralement les articles d'une qualité particulière. Il y a trois marqueurs disponibles "revue", "qualité" et "vierges". Par exemple, un article est qualifié de qualité s'il a le niveau 1 pour la précision et la profondeur et le niveau 2 pour la lisibilité. Ainsi chaque page possède une version 'stable' la dernière version qui a été marquée à un niveau suffisamment élevé et la version 'récente'. FlaggedRevs est un procédé de révision qui ne concerne qu'une page wiki à la fois.

En plus de ces procédés, les wikis fournissent un mécanisme d'annulation qui permet aux utilisateurs de corriger leurs propres erreurs et de défaire les actes de vandalisme. Ce mécanisme aide également à maintenir la qualité du contenu des wikis.

Le passage aux wikis sémantiques rend la tâche de maintenance des contenus plus complexe. La question est de savoir si les procédés légers utilisés dans les wikis sont toujours adaptés pour la maintenance des contenus structurés.

Les wikis sémantiques sont à la fois des systèmes collaboratifs distribués et des outils pour l'ingénierie des ontologies (légères). Dans le domaine de l'ingénierie des ontologies, de nombreux travaux existent sur les procédés de construction et de maintenance des connaissances tels que DILIGENT [VPST05], UPON [NMN09] et CO4 [Euz01].

Le procédé d'ingénierie d'ontologies DILIGENT est illustré dans la figure 1.15. Il comprend les cinq étapes suivantes : (1) *construire* initialement une ontologie partagée par

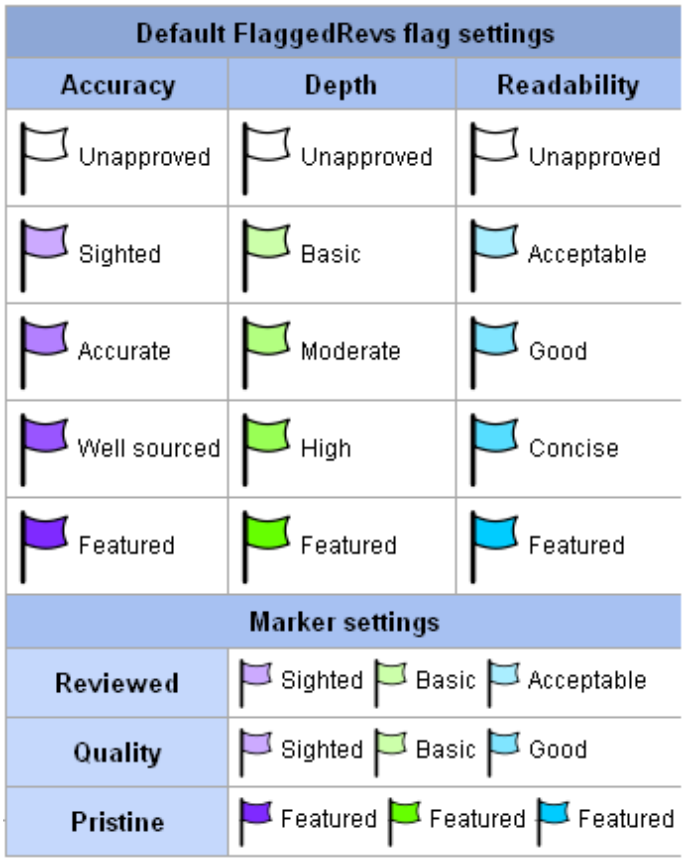

FIGURE 1.14 – Les drapeaux et marqueurs de FlaggedRevs extraits de Wikipedia

tous les participants, (2) *adaptation locale* de cette ontologie par les utilisateurs, (3) *analyse* des demandes de changements par un conseil de contrôle, (4) *révision* de l'ontologie partagée qui inclut la décision d'intégration des changements dans la nouvelle version de l'ontologie et (5) *mise à jour locale* des utilisateurs de leur ontologie locale par la nouvelle version de l'ontologie partagée.

Le procédé CO4 (pour la Construction coopérative de bases de connaissance consensuelles) [Euz01] illustré dans la figure 1.16 est composé des bases de connaissance qui sont organisées en un arbre. Les feuilles sont les bases individuelles et les nœuds représentent le consensus entre les fils. Les utilisateurs modifient leur base locale, puis ils soumettent leurs modifications en tant que propositions à la base consensuelle à laquelle ils adhèrent. Ensuite, ces modifications seront testées pour vérifier qu'elles ne violent pas la cohérence de la base de groupe. Si c'est le cas, elles seront propagées et commentées par tous les utilisateurs. Ces modifications seront intégrées dans la base consensuelle, une fois un consensus est établi.

Ces procédés d'ingénierie des ontologies sont à rapprocher des procédés utilisés en génie logiciel [NMN09]. Les gestionnaires de configuration distribués (DVCS) comme Git [git]

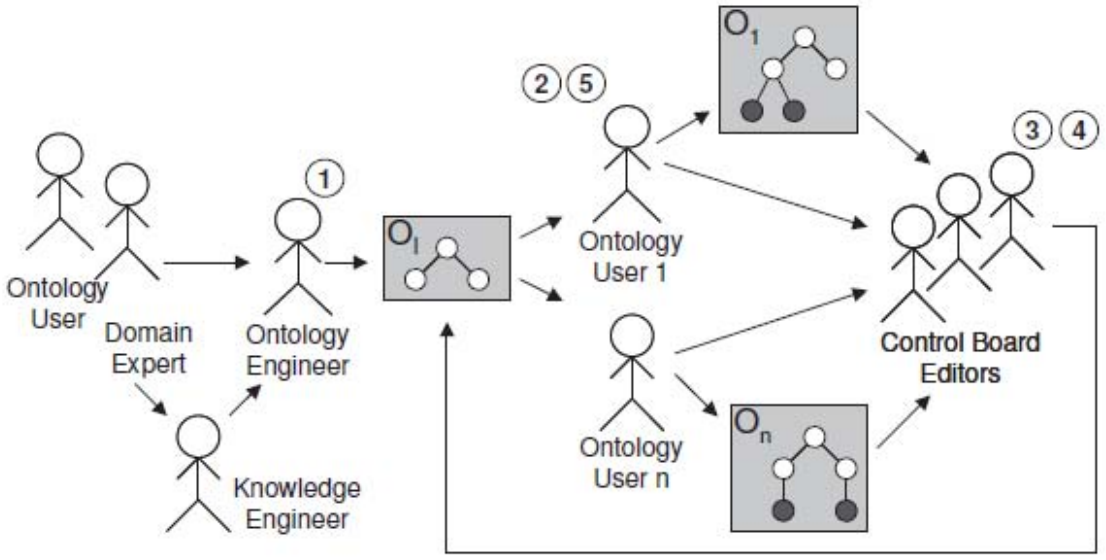

FIGURE 1.15 – Le procédé d'ingénierie d'ontologies DILIGENT extrait de [VPST05]

permettent à des communautés d'utiliser des workflows pour contrôler les changements sur la base de code. La communauté du noyau Linux utilise le worflow appelé *dictateur et lieutenant* (voir figure 1.17).

Le procédé *dictateur et lieutenant* est modélisé comme illustré dans la figure 1.17. Des utilisateurs appelés *développeurs* et *lieutenants* récupèrent le contenu d'un répertoire partagé contenant plusieurs fichiers appelé "blessed repository". Les *développeurs* modifient localement ces fichiers. Puis, les *lieutenants* récupèrent ces changements, ils les analysent et les intègrent si nécessaire. Ensuite, un utilisateur particulier appelé *dictateur* récupère les changements validés par les *lieutenants*, de même, il les analyse, les teste et les intègre si nécessaire. Finalement, le *dictateur* publie ses changements dans le répertoire "blessed repository".

De manière générale, les DVCS gèrent la propagation des changements d'un espace de travail à un autre et chaque espace de travail est associé à un niveau de confiance. Nous pouvons faire l'analogie avec CO4 où les changements sur les ontologies sont propagés d'une base ontologique à une autre base sous le contrôle des utilisateurs. Les DVCS et CO4 sont des exemples caractéristiques de l'édition multi-synchrone que nous avons décrits dans la section 1.1.1. Donc l'édition multi-synchrone permet l'expression des procédés et donc favorise la coordination dans les systèmes collaboratifs. C'est cette coordination qui garantit la qualité du contenu. Le support de l'édition multi-synchrone et des procédés est un problème ouvert dans le domaine des wikis sémantiques.

En conclusion, supporter l'édition multi-synchrone dans les wikis sémantiques permet d'une part, l'implémentation des méthodologies de la construction collaborative d'onto-

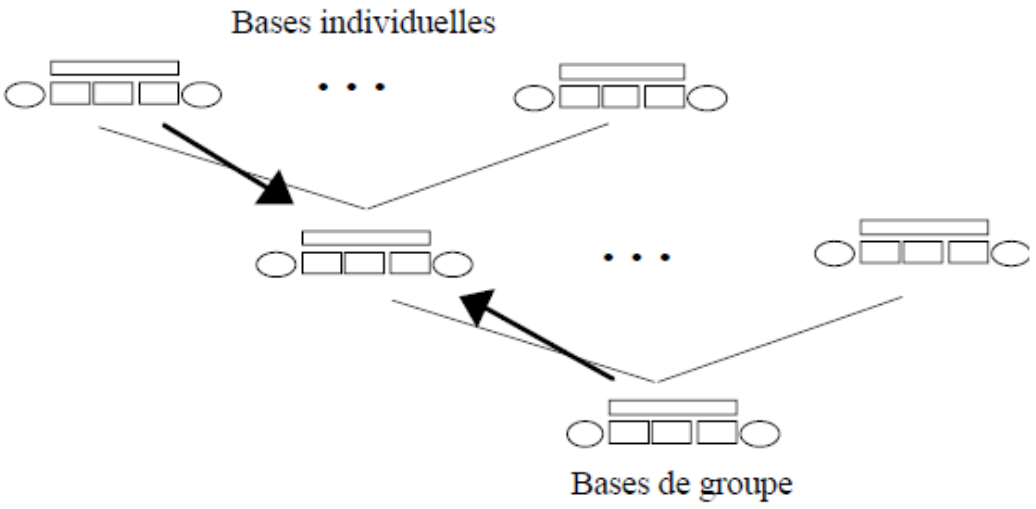

FIGURE 1.16 – Le procédé d'ingénierie d'ontologies CO4 extrait de [Euz01]

logies et d'autre part, d'améliorer la maintenance des données structurées dans ces wikis puisqu'elle offre des changements multi-pages qui peuvent être effectués en isolation avant d'être publiés.

1.4 Approche

Nous proposons d'utiliser le modèle de réplication optimiste pour construire un wiki sémantique sur réseau P2P. La réplication optimiste répond aux motivations citées ci-dessus. Elle améliore le passage à l'échelle, la disponibilité des données et la performance [SS05b]. Elle permet également de supporter facilement l'édition multi-synchrone et d'implémenter des procédés d'édition collaborative. Par exemple, les gestionnaires des contrôles de versions distribués (DVCS) tels que Git [git] sont basés sur la réplication optimiste. Ils supportent l'édition multi-synchrone et implémentent différents procédés. La réplication optimiste utilisent des mécanismes de propagation qui assurent que toutes les opérations émises arrivent sur tous les sites connectés ou non, ce qui permet de supporter une édition déconnectée.

1.4.1 Réplication optimiste

Le modèle de réplication optimiste considère des sites qui répliquent des données appelées répliques. Quand un site modifie sa copie locale, il génère une opération correspondante. Cette opération est traitée en quatre étapes :

1. Elle est exécutée localement sur la réplique locale du site,

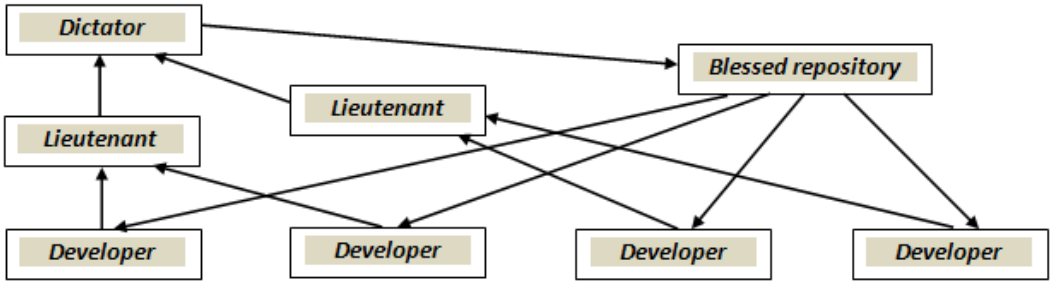

FIGURE 1.17 – Le procédé dictateur et lieutenant implémenté dans Git [git]

2. elle est propagée à travers le réseau aux autres sites répliquant cette donnée,
3. elle est reçue par ces sites,
4. elle est intégrée dans leur copie locale. Si nécessaire, le processus d'intégration fusionne cette modification avec des modifications concurrentes générées soit par le site local, soit reçues d'autres sites distants.

Un système de réplication optimiste est considéré correct s'il assure un modèle de cohérence sur les répliques. Plusieurs modèles de cohérence sont définis dans la réplication optimiste tels que la cohérence causale [Lam78], la cohérence éventuelle [SS05b] et le modèle de cohérence CCI (Causalité, Convergence, Intention) [SJZ+98]. La cohérence causale assure que les opérations ordonnées par une relation de précédence seront exécutées dans le même ordre sur tous les sites. La cohérence éventuelle assure que toutes les répliques des sites sont identiques lorsque le système est au repos. Le modèle de cohérence CCI préserve la causalité et l'intention des opérations et assure la convergence des répliques. Dans la communauté TCAO [4], un éditeur collaboratif répliquant des données est considéré correct s'il respecte le modèle de cohérence CCI [SJZ+98]. Nous détaillons le modèle de cohérence CCI dans la section 1.4.2.

La réplication optimiste permet de modifier les répliques des différents sites en parallèle, par conséquent ces répliques divergent. Ce qui nécessite d'utiliser un algorithme de réplication optimiste pour la synchronisation des répliques. Les algorithmes de synchronisation dépendent du type de données répliquées. La plupart des algorithmes existants s'intéressent aux structures linéaires i.e. texte. Par exemple, Woot [OUMI06] et Logoot [WUM09b] sont des algorithmes de réplication optimiste qui synchronisent des données linéaires.

4. L'acronyme de Travail Collaboratif Assisté par Ordinateur, qui correspond en anglais à CSCW (Computer Supported Collaborative Work)

1.4.2 Modèle de cohérence CCI

Le modèle de cohérence CCI [SJZ+98] est défini comme suit :

Préservation de la causalité : la relation d'ordre causal entre les opérations selon Lamport [Lam78] en termes de leur génération et de leurs séquences d'exécution est définie comme suit :

Définition 2. *(Relation d'ordre causal "→"). Etant donné deux opérations Oa et Ob, générées sur des sites i et j, Oa → Ob, si et seulement si :*

1. *$i = j$, et la génération de O_a précède celle de O_b ;*
2. *$i \neq j$, et l'exécution de O_a sur le site j précède la génération de O_b ;*
3. *il existe une opération O_x, telle que $O_a \rightarrow O_x$ et $O_x \rightarrow O_b$.*

Par contre elles sont dites concurrentes, si et seulement si ni $O_a \rightarrow O_b$, ni $O_b \rightarrow O_a$, cette relation est représentée par $O_a \parallel O_b$.

Définition 3. *(Préservation de la causalité). Pour chaque paire d'opérations O_a et O_b, si $O_a \rightarrow O_b$, alors O_a est exécutée avant O_b sur tous les sites.*

Exemple sur la causalité

Nous illustrons la propriété de la causalité par un exemple (voir figure 1.18). Considérons trois sites interconnectés par un réseau informatique. Trois utilisateurs éditent en parallèle un questionnaire sur ces trois sites. Sur le site $Site1$, un utilisateur U_1 insère une première question "Q1 ?" à la première ligne. Puis il sauvegarde, ce qui génère l'opération op_{11}. Cette opération est propagée vers les autres sites. Une fois reçue par le site $Site2$, op_{11} est tout de suite intégrée. L'utilisateur U_2 sur le site $Site2$ répond à la question en insérant "R1" à la deuxième ligne. Le système génère l'opération op_{21} qui est propagée aux autres sites pour y être exécutée. Puisque op_{21} est générée après op_{11}, donc $op_{11} \rightarrow op_{21}$ et l'exécution de op_{11} doit précéder l'exécution de op_{21} sur tous les sites afin de préserver la causalité. Lors de la réception de op_{21} sur le site $Site1$, elle est toute suite intégrée. Tandis que lors de sa réception sur le site $Site3$, op_{21} est mise en attente de la réception et de l'intégration de op_{11} dont elle est causalement dépendante. Le résultat final de l'exécution de toutes les opérations préserve la causalité.

Convergence : Sun [SJZ+98] définit la convergence comme suit :

Définition 4. *(Convergence). Lorsque le même ensemble d'opérations a été exécuté sur tous les sites, toutes les copies du document partagé sont identiques.*

Exemple sur la convergence

Nous illustrons par un exemple une violation de la de convergence qui peut avoir lieu sur les répliques des différents sites (voir figure 1.19). Considérons deux sites

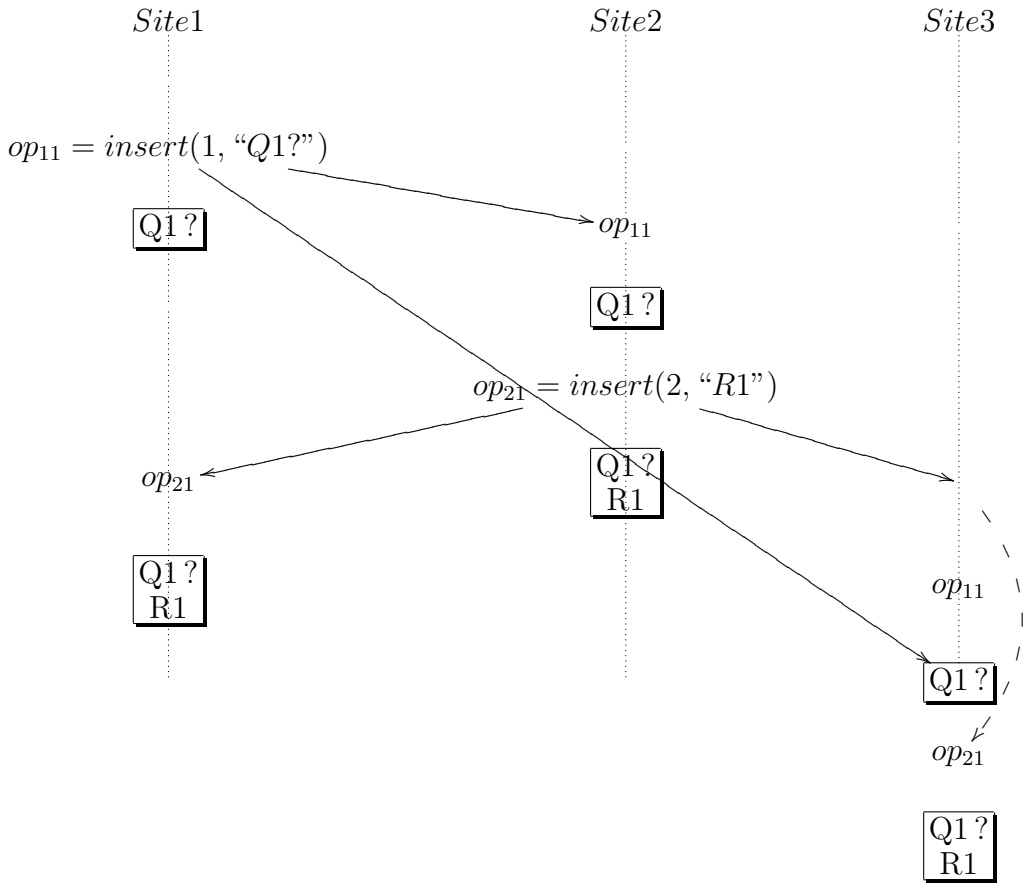

FIGURE 1.18 – Respect de la causalité des opérations

interconnectés par un réseau informatique. Chaque site réplique un document textuel contenant initialement deux lignes AB. Pour modifier sa réplique, un site génère une opération d'insertion qui est exécutée immédiatement sur le site puis propagée aux sites voisins. Considérons deux utilisateurs qui modifient parallèlement les répliques des sites. Le premier utilisateur insère X à la deuxième ligne du document sur le site $Site1$. En parallèle, le deuxième utilisateur insère Y à la même position sur le site $Site2$. Ce qui génère deux opérations concurrentes op_1 et op_2 qui modifient le document des sites $Site1$ et $Site2$ respectivement. Ensuite ces opérations sont propagées et intégrées sur les autres sites. Le résultat final est "AYXB" sur le site $Site1$ et "AXYB" sur le deuxième site, donc les répliques des deux sites divergent.

Préservation de l'intention : Sun et al. [SJZ+98] définit l'intention et la préservation de l'intention comme suit :

Définition 5. *(Intention) L'intention d'une opération op est l'effet observé lors de son exécution sur son état de génération.*

Définition 6. *(Préservation de l'intention)*

1. *Pour toute opération op, les effets de l'exécution de op sur tous les sites doivent être les mêmes que l'intention de op.*

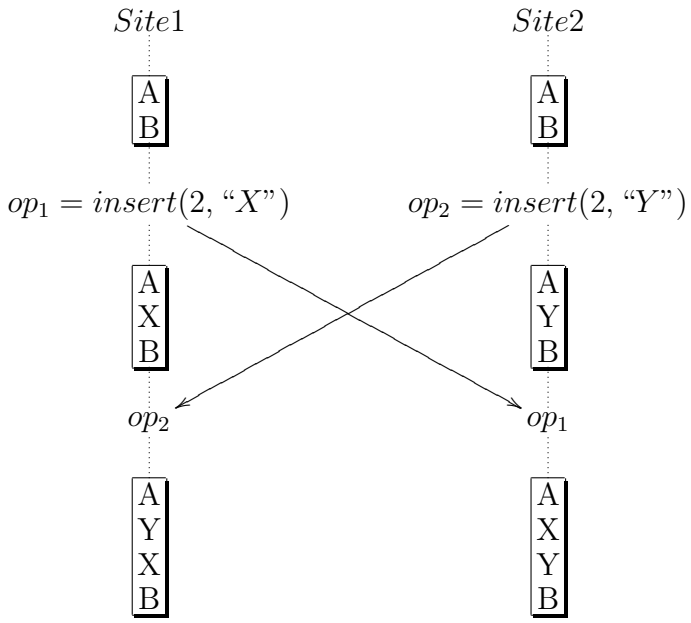

FIGURE 1.19 – Violation de la convergence après intégration des modifications concurrentes

2. *L'effet de l'exécution d'une opération op ne doit pas changer les effets des opérations concurrentes.*

Exemple sur la préservation de l'intention

En fonction de la définition des intentions des opérations, la préservation de l'intention peut être définie ou non. Par exemple, si on réplique un objet de type chaîne de caractères avec une opération d'insertion d'un caractère c à une position p : $ins(p, c)$. Si on définit l'intention de l'opération *ins* comme insérer le caractère c exactement à la position p alors la préservation de l'intention n'est pas définie. Le résultat de l'insertion de deux opérations concurrentes $ins(A, 1)$ et $ins(B, 1)$ dans une chaîne initiale "X" répliquée est nécessairement soit "XAB", soit "XBA". Par conséquent, la préservation de l'intention est violée à cause de la concurrence, un des caractères est inséré à la deuxième position. Si on définit l'intention de *ins* comme l'insertion entre le caractère précédent et le caractère suivant, alors la notion de préservation de l'intention est définie.

En effet, prenons une chaîne de caractères ayant une valeur initiale "AB" comme illustrée dans la figure 1.20. Un utilisateur sur le site $Site1$ insère 'X' entre 'A' et 'B' ; op1 = $ins(1, X)$ et l'intention de op_1 est de faire respecter la contrainte A $\prec$ X $\prec$ B, autrement dit 'X' est placé derrière 'A' et devant 'B'. Un autre utilisateur sur le site $Site2$ insère en parallèle 'Y' entre 'A' et 'B' ; op2 = $ins(1, Y)$ et l'intention de op2 est de préserver A $\prec$ Y $\prec$ B. La figure 1.20 illustre la propagation de ces deux opérations sur les deux sites. Dès qu'une opération arrive sur un site, elle est exécutée. Les états finaux "AXYB" et "AYXB" préservent les intentions de op1 et op2. En effet, les deux contraintes A $\prec$ X

$\prec$ B et A $\prec$ Y $\prec$ B sont bien respectées et l'effet de op1 n'altère pas l'effet de op2 et réciproquement. Malheureusement, les répliques ne convergent pas. Il est donc tout à fait possible de préserver les intentions sans converger les répliques.

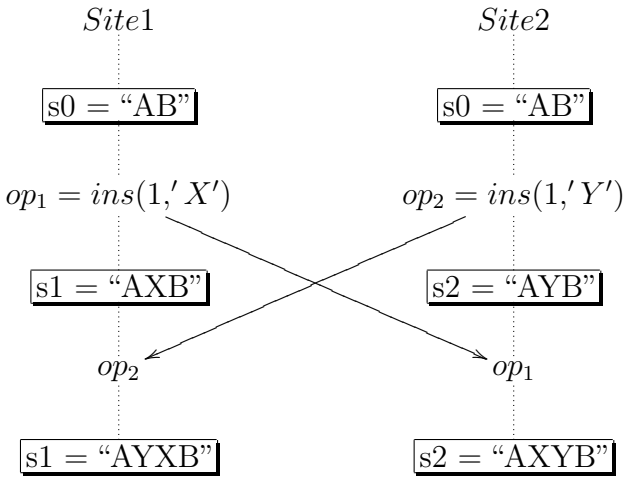

FIGURE 1.20 – Préservation de l'intention

La définition de l'intention des opérations modifiants le type de données répliquées doit être établie. Cette intention doit être préservée par les algorithmes de synchronisation.

1.5 Problèmes scientifiques

Le développement d'un wiki sémantique P2P basé sur une réplication optimiste soulève plusieurs questions telles que : comment manipuler, répliquer et synchroniser le nouveau type des données (texte et annotations) ? Et comment assurer le modèle de cohérence CCI sur ces données ? Les problèmes à résoudre dans la conception d'un wiki sémantique P2P sont les suivants :

1.5.1 Quel type de données ?

Les wikis sémantiques manipulent un nouveau type de données qui est formé des pages wikis et de leurs annotations sémantiques. La question est comment répliquer ce nouveau type de données dans le contexte d'un wiki sémantique P2P. Est-ce qu'il faut intégrer les annotations dans le contenu des pages wikis ou de les éditer séparément. Quel type de réplication faut il adopter ? Est ce qu'il faut répliquer ce nouveau type de données sur tous les pairs du réseau ou seulement sur un sous ensemble. Enfin, quelles sont les opérations à définir pour manipuler ce type de données ? Il faut définir les intentions de ces opérations. La réplication de ce type de données sur un réseau P2P n'a jamais été étudiée.

1.5.2 Quel mécanisme de propagation ?

Afin de propager les changements dans un wiki sémantique P2P, l'algorithme de propagation utilisé doit passer à l'échelle, être fiable et soutenir la dynamicité du réseau. En effet, dans un réseau P2P, les pairs joignent et quittent le réseau à tout moment. Pour cela, l'algorithme de propagation utilisé doit garantir que les changements envoyés seront éventuellement livrés à tous les pairs. Il faut traiter les cas de déconnexion ou de pannes. Un mécanisme de diffusion épidémique (epidemic broadcast) [DGH+87] est souvent utilisé dans la réplication optimiste. Il permet de propager les opérations d'une façon fiable à tous les sites même quand ils ne peuvent pas communiquer entre eux directement. Est ce que ce mécanisme de diffusion est adapté pour les wikis sémantiques sur réseau P2P ? sinon, est-ce qu'il faut développer de nouveaux mécanismes de propagation ? Il faut choisir la manière de propager les opérations qui peut être soit sous le contrôle du wiki ou soit des utilisateurs afin de supporter l'édition multi-synchrone et les procédés d'édition collaborative.

1.5.3 Quel algorithme de synchronisation ?

Les algorithmes de réplication optimiste existants sont conçus pour synchroniser des données linéaires i.e. du texte. Serait-il possible d'adapter ces algorithmes pour synchroniser le nouveau type de données dans le contexte d'un wiki sémantique P2P ? Est-il nécessaire de développer de nouveaux algorithmes ? Un wiki sémantique sur réseau pair-à-pair doit supporter un algorithme de réplication optimiste qui assure que les pairs ayant les mêmes répliques des pages wikis ainsi que leur entrepôt des annotations sémantiques finiront par converger lorsque le système est au repos. Cette convergence doit être atteinte tout en préservant l'intention des opérations définie sur le type de données répliquées. L'algorithme de réplication doit synchroniser le nouveau type de données tout en assurant le modèle CCI. Une cohérence entre les annotations dans les pages et les entrepôts sémantiques doit être maintenue.

1.5.4 Quel mécanisme d'annulation ?

Le mécanisme d'annulation doit respecter la définition de l'annulation donnée dans la section 2.9. Un wiki sémantique P2P doit supporter un algorithme approprié responsable de la génération et de l'intégration des actions d'annulation *undo* et d'annulation de l'annulation *redo*. Cet algorithme ne doit pas violer le modèle CCI sur les données du wiki sémantique. Le mécanisme d'annulation doit assurer la convergence des pages wikis et des entrepôts des annotations sémantiques sur tous les pairs. Il doit être indépendant des modifications concurrentes, de l'ordre de l'intégration des actions undo et redo et du fait que les utilisateurs peuvent éditer en mode déconnecté et joindre ou quitter le réseau à tout moment.

1.6 Synthèse

Un wiki sémantique est un éditeur collaboratif basé sur une architecture client-serveur qui passe mal à l'échelle et qui tolère difficilement les pannes. Son modèle actuel ne permet pas l'édition multi-synchrone, ni d'implémenter facilement les procédés d'édition collaborative. Pour répondre à ces limitations, nous proposons une approche qui consiste à instancier le modèle de réplication optimiste pour les wikis sémantiques. Le résultat est un wiki sémantique distribué sur un réseau P2P basé sur la réplication optimiste. La réalisation de cette instanciation soulève plusieurs questions : comment manipuler et répliquer le nouveau type de données des wikis sémantiques (pages wikis et annotations) ? comment propager et synchroniser les changements sur ces données ? Et comment assurer le modèle de cohérence CCI sur ces données répliquées ?

1.7 Systèmes collaboratifs répliqués

Dans cette section, nous nous intéressons aux approches répliquées qui pourraient être utilisées pour construire un wiki sémantique P2P. Dans un premier temps, nous nous intéressons aux approches génériques qui permettent de synchroniser des données sur le réseau P2P avec une cohérence CCI. Ensuite, nous nous intéressons aux approches qui permettent le partage et la synchronisation des données sémantiques sur réseau P2P. Enfin, nous présentons certains éditeurs collaboratifs multi-synchrones et les wikis P2P existants.

1.7.1 Maintenance de la cohérence dans les approches P2P

De nombreuses approches basées sur la réplication optimiste ont été développées, certaines sont capables de supporter le passage à l'échelle en fonction du nombre d'utilisateurs dans le réseau et d'assurer le modèle de cohérence CCI. Par conséquent, ces approches peuvent être utilisées dans le développement des wikis sur réseau pair-à-pair.

Transformations Opérationnelles (OT)

L'approche des transformées opérationnelles (OT) [SE98] utilisée par Google Wave [5] a été développée par la communauté des éditeurs collaboratifs synchrones. L'architecture générale du modèle des transformées opérationnelles distingue deux composants :

– un algorithme d'intégration. Il est responsable de la réception, de la diffusion et de l'exécution des opérations. Si nécessaire, il fait appel aux fonctions de transformation. Cet algorithme est indépendant du type des données manipulées (chaîne de caractères, page wiki, document XML, système de fichiers).

5. http ://wave.google.com

– un ensemble de fonctions de transformation. Ces fonctions ont la charge de "fusionner" les mises à jour en sérialisant deux opérations concurrentes. Ces fonctions sont spécifiques à un type de données particulier.

L'algorithme d'intégration est défini comme correct s'il assure la Causalité, la Convergence et la préservation des Intentions (CCI) [SJZY97, SE98]. Pour assurer la convergence des données répliquées dans OT, les fonctions de transformation doivent respecter deux conditions dites C1 et C2 définies comme suit :

– (Condition C1) Soient op1 et op2 deux opérations concurrentes définies sur le même état. La fonction de transformation T vérifie la condition C1 si et seulement si : op1 $\circ T(op2, op1) \equiv$ op2 $\circ T(op1, op2)$ où $\circ$ dénote un opérateur concaténant deux opérations pour former une séquence d'opérations, et où $\equiv$ signifie que les deux séquences op1 $\circ T(op2, op1)$ et $op2 \circ T(op1, op2)$ produisent le même état.
– (Condition C2) Soient trois opérations op1, op2 et op3 concurrentes définies sur le même état. La fonction de transformation T vérifie la condition C2 si et seulement si : T(T(op3, op1), T(op2, op1)) = T(T(op3, op2), T(op1, op2))

Pour détecter la causalité et donc la concurrence entre les opérations dans OT, tous les algorithmes d'intégration existants dans OT à part MOT2 [CF07] exigent l'utilisation des vecteurs d'horloge et par conséquent, ils ne passent pas à l'échelle [OUMI06].

L'algorithme d'intégration MOT2 est un algorithme de réconciliation P2P, il est le seul dans OT qui passe à l'échelle par rapport au nombre d'utilisateurs. Afin d'assurer la causalité entre les opérations, la synchronisation entre les pairs est effectuée deux à deux. Par conséquent, l'échange des messages dans le réseau est efficace mais le problème se trouve dans la taille de ces messages qui peut être non bornée. Lors de chaque synchronisation, il est nécessaire d'envoyer toute l'historique (la liste de toutes les opérations produites sur une réplique des données) pour déterminer le préfixe commun entre les deux pairs puisque l'ordre des opérations peut changer, ce qui peut poser un problème. Par conséquent, la propagation des changements entre les pairs et la synchronisation des pairs sont lentes. D'autre part, cet algorithme suppose l'existence des fonctions de transformation répondant aux contraintes C1 et C2 [IROM06]. L'écriture de telles fonctions est un problème réputé difficile et requiert des contraintes fortes telles que la vérification des contraintes C1 et C2.

L'approche CRDT

Cette approche présentée dans [PMSL09] propose un cadre générique appelé CRDT ("type de donnée répliqué commutatif") pour construire des éditeurs collaboratifs distribués assurant les critères CCI. Dans cette approche, les modifications produites localement sont ré-exécutées sur des répliques distantes. Il n'y a pas d'ordre total sur les opérations, par conséquent, elles peuvent être exécutées dans différents ordres. L'idée principale est d'utiliser un type de données où toutes les opérations commutent. Combiné avec le respect de la relation de causalité entre les opérations, la commutativité garantit la convergence.

Pour atteindre la commutativité sur une structure linéaire, les auteurs proposent une solution basée sur un ordre total entre les lignes du document. Ils définissent deux opérations pour la modification d'un document : insert(*pid* ; *text*) qui insère une ligne contenant *text* à la position désignée par l'identificateur *pid* et delete(pid) qui supprime la ligne identifiée par *pid*. Il existe plusieurs instanciations de l'approche CRDT, elles se distinguent dans la façon de générer l'identifiant unique pid et d'utiliser ou non des pierres tombales. L'utilisation d'une pierre tombale implique qu'au lieu de supprimer effectivement une ligne du modèle du document, elle sera remplacée par une pierre tombale qui maintient des informations utiles sur la ligne. Par la suite, nous présentons trois algorithmes CRDT.

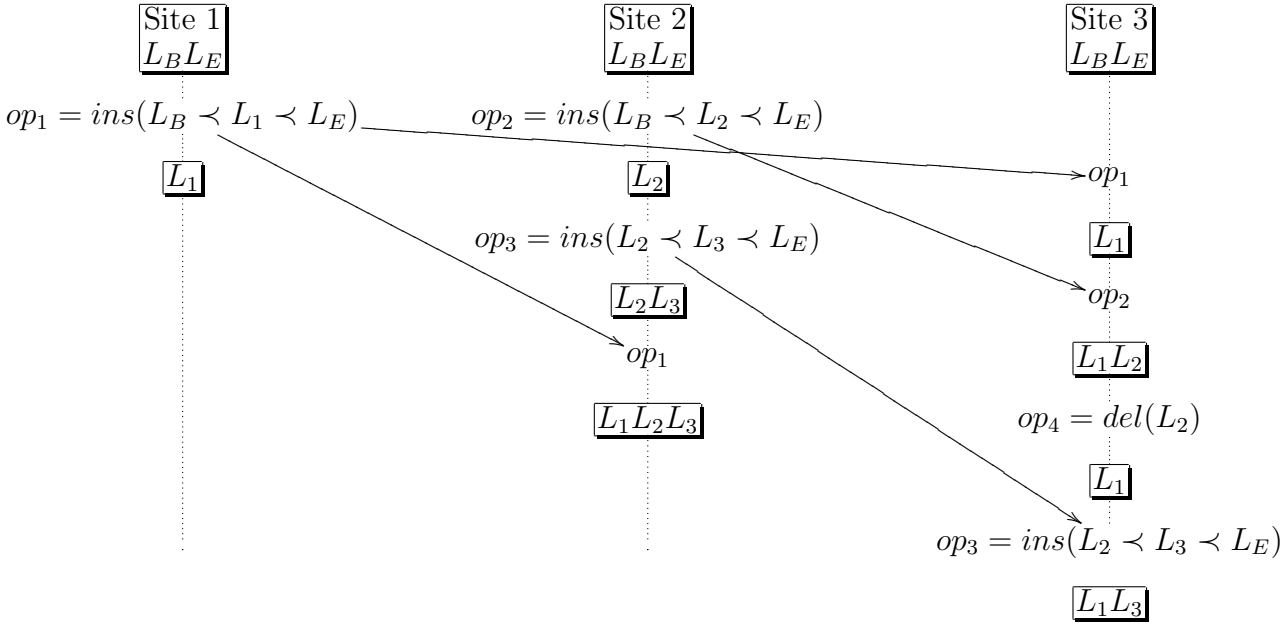

FIGURE 1.21 – Scénario d'exécution de WOOT

Woot [OUMI06] L'idée principale de Woot est de traiter un document édité collaborativement en tant qu'un diagramme de Hasse représentant l'ordre induit par les opérations d'insertion. L'algorithme Woot calcule une linéarisation de ce diagramme. Woot utilise un ordre total sur les lignes du document basé sur leur identifiant unique. Un document de Woot est une séquence ordonnée de lignes éditée par deux types d'opérations : insert(l, l_p, l_n) et delete(l). Une ligne l est un quadruplet $l = < id, \alpha, v, id_{lp}, id_{ln} >$ où id est un identifiant unique de l, α est le contenu de l, v indique si l est visible ou non, id_{lp} et id_{ln} sont les identifiants de la ligne qui précède et et celle qui suit l. Le document de Woot contient initialement deux lignes virtuelles l_B et l_E qui représentent le début et la fin du document. Dans Woot, chaque site s possède un identifiant unique $numSite_s$ et une horloge logique H_s utilisés ensemble pour identifier les lignes d'une façon unique. Lorsqu'une ligne est générée sur le site s, son identifiant est formé de la paire ($numSite_s$, H_s) et l'horloge H_s est incrémentée. Puisque $numSite_s$ est unique, alors cette paire forme un identifiant

unique pour les lignes. Les identifiants des sites sont totalement ordonnés. Dans Woot, l'identifiant des lignes est donc de taille bornée. Les lignes supprimées sont conservées comme des pierres tombales pour assurer la convergence du document sur les différents sites. Ces pierres tombales ne peuvent pas être supprimées sans compromettre la cohérence du document.

Exemple Soient trois sites Site 1, Site 2 et Site 3 possédant chacun une copie d'un document vide qui contient uniquement deux lignes virtuelles "$L_B L_E$". Nous considérons le scénario de la figure 1.21. Une fois toutes les opérations sont reçues, le scénario produit le diagramme de Hasse illustré par la figure 1.22. Nous supposons que la relation $<_{id}$ ordonne les lignes de la manière suivante : $L_1 <_{id} L_2 <_{id} L_3$. Nous supposons que le Site 3 reçoit les opérations op1, op2 et op3 dans cet ordre. L'intégration se déroule de la manière suivante :

1. Integrate(op_1) = IntegrateIns(L_1, L_B, L_E) qui implique intégrer L_1 entre L_B et L_E : WOOT commence à extraire la liste S des lignes qui sont entre L_B et L_E afin de comparer leur identifiant avec celui de L_1 avant de l'insérer. Dans ce cas $S = \emptyset$, L_1 sera donc tout suite insérée et le résultat obtenu est la chaîne L_1.
2. Integrate(op_2) = IntegrateIns(L_2, L_B, L_E) : WOOT extrait les lignes qui sont entre L_B et L_E, $S = \{L_1\}$. Donc avant d'insérer L_2, il doit comparer son identifiant avec celui de L_1 qui est en concurrence avec elle. L'identifiant de L_1 = (Id_{s1}, 1) et celui de L_2 = (Id_{s2}, 1) puisque $Id_{s1} \prec Id_{s2}$ donc $L_1 <_{id} L_2$, L_2 doit être insérée après L_1 et le résultat obtenu est la chaîne $L_1 L_2$.
3. Integrate(op_4) = IntegrateDel(L_2) : WOOT rend invisible la ligne L_2, aucune suppression physique n'aura lieu et le résultat obtenu est L_1.
4. Integrate(op_3) = IntegrateIns(L_3, L_2, L_E) : WOOT extrait les lignes entre L_2 et L_E, puisque L_2 n'a pas été physiquement supprimée, il y a aucun problème pour la trouver. Dans ce cas, $S = \emptyset$, donc L_3 est insérée entre L_2 et L_E.

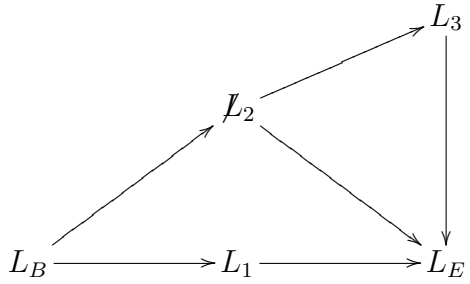

FIGURE 1.22 – Diagramme de Hasse résultant du scénario de l'exemple.

TreeDoc [LPS09] TreeDoc est un système d'édition collaborative qui utilise un arbre binaire pour représenter le document. Cette structure en arbre est introduite pour

maintenir l'ordre total entre les identifiants. Toutefois, la suppression d'un élément de cet arbre requiert l'utilisation des pierres tombales. Donc les lignes supprimées du document sont conservées comme des pierres tombales. Les auteurs proposent une procédure de "validation à deux phases" pour enlever les pierres tombales. Malheureusement, cette procédure ne peut pas être utilisée dans un réseau P2P puisqu'elle requiert l'envoi des messages à l'ensemble des pairs du réseaux. Récemment dans [PMSL09], les auteurs ont présenté une amélioration de TreeDoc permettant de supprimer les pierres tombales "feuilles". Dans TreeDoc, l'identifiant d'une ligne est égal à son chemin dans l'arbre donc il est à priori non borné.

Logoot [WUM09b] L'idée de Logoot est d'utiliser un identifiant de position sur la base d'une liste de nombres entiers pour chaque ligne. Avec une telle identification, une ligne peut être supprimée du document, sans affecter l'ordre des lignes restantes. Un document Logoot est composé de lignes définies par : $< pid; content >$ où *content* est le contenu de la ligne et *pid* un identifiant unique de position. Deux lignes virtuelles l_B et l_E représentent le début et la fin du document. Dans Logoot, un identifiant est un couple $< p_i; s_i >$ où p_i est un entier et s_i est l'identifiant du site. Une position est une liste d'identifiants et un identifiant de position généré sur une réplique d'un site s est un couple $< pos; h_i >$ où pos $= i_0.i_1...i_{n-1}. < p, s >$ est une position et h_s est la valeur de l'horloge logique du site. Ainsi, chaque identifiant de position est unique puisque la dernière position de la liste $i_0.i_1...i_{n-1}. < p, s >$ contient l'identifiant unique du site ainsi que la valeur de son horloge. L'idée principale est de générer des identifiants de position uniques, totalement ordonnés dans un espace dense c'est-à-dire pour tous identifiants de position id_1 et id_2, tels que $id_1 < id_2$, il existe un identifiant id_3 tel que $id_1 < id_3 < id_2$ et Logoot génère une ligne ayant cet identifiant calculé selon la définition ci-dessus. Les deux opérations d'édition dans Logoot sont : insert($pid, content$) et delete(pid). La dernière supprime effectivement la ligne du document. Logoot ne nécessite pas de pierres tombales. Par conséquent, le surcoût reste linéaire en fonction de la taille du document au cours de la durée d'édition et aucune procédure de purge de pierres tombales n'est nécessaire. Dans Logoot, les identifiants sont éventuellement non bornés, mais les auteurs ont montré que la taille de ces identifiants associés à chaque ligne reste acceptable dans la pratique. Ils ont conduit des expérimentations montrant que Logoot a de meilleures performances en moyenne que les algorithmes TreeDoc et Wooto [WUM07] qui est une version optimisée de Woot.

Exemple Soient trois sites Site 1, Site 2 et Site 3 possédant chacun une réplique d'une chaîne vide qui contient donc uniquement deux lignes virtuelles "$< 0, 0, 0 >, L_B < \infty, 0, 0 >, L_E$". Nous considérons le scénario de la figure 1.23.

1. Sur le Site 1, un utilisateur insère la ligne L_1 à la première position. Logoot génère l'opération $op_1 = ins(< 1, 1, 1 >, L_1)$ ayant comme identifiant de po-

sition $idp =< 1, 1, 1 >$ où le premier 1 est un entier indiquant la première position, le deuxième 1 est l'identifiant du site et le troisième 1 est la valeur de l'horloge logique du site qui a été incrémentée. L'identifiant idp est tel que $< 0, 0, 0 > \prec \text{idp} \prec < \infty, 0, 0 >$. L'opération op_1 est propagée aux autres sites pour être intégrée.

2. En parallèle sur le Site 2, un autre utilisateur insère la ligne L_2 à la première position. Logoot génère l'opération $op_2 = ins(< 1, 2, 1 >, L_1)$. De même, op_2 est propagée aux autres sites pour être intégrée.
3. Une fois reçues sur le Site 3, op_1 et op_2 sont intégrées chacune à sa position. En comparant l'identifiant des opérations $< 1, 1, 1 >$ et $< 1, 2, 1 >$, L_1 est donc insérée après L_2 et le résultat est L_1L_2.
4. Un utilisateur sur le Site 3 insère une ligne L_3 entre L_1 et L_2. Pour calculer une position entre ces deux lignes, Logoot génère une opération ayant un identifiant $idp =< 1, 1, 1 >< 6, 3, 1 >$ tel que $< 1, 1, 1 > \prec \text{idp} \prec < 1, 2, 1 >$. La valeur 6 est un entier généré aléatoirement, le 3 est l'identifiant du Site 3 et le 1 est la valeur de son horloge logique.

Synthèse

L'approche CRDT nécessite seulement une réception causale pour assurer la cohérence CCI sur des structures linéaires. Cette approche passe mieux à l'échelle par rapport à l'approche des transformées opérationnelles. Par conséquent, MOT2 et l'approche CRDT peuvent être utilisées pour la synchronisation des pages wikis dans un wiki sémantique distribué sur réseau pair-à-pair. Mais ils ne permettent pas de synchroniser des graphes RDF.

1.7.2 Réplication dans le web sémantique P2P

De nombreuses recherches ont été effectuées dans le web sémantique pair-à-pair [NWQ+02, MTEBG07, TMP+04, CF04, CIKN04, SS05c]. Ces travaux se concentrent sur le partage, le stockage, l'extraction et la synchronisation des ressources RDF distribuées sur un réseau pair-à-pair mais ils ne sont pas orientés pour le contexte d'édition collaborative. Le partage des ressources est différent d'une édition collaborative de ces ressources. Dans le contexte du partage, certains pairs publient des données tandis que les autres peuvent les récupérer, l'édition concurrente de ces données n'est pas donc gérée. Dans un wiki sémantique pair-à-pair, certains pairs publient les données, d'autres pairs peuvent les lire et les modifier collaborativement et un algorithme de synchronisation intègre les modifications concurrentes ou non tout en maintenant la cohérence des données. Par la suite, nous focalisons sur les algorithmes de synchronisation des ressources RDF. Les algorithmes de synchronisation existants des ressources RDF tels que RDFSync [MTEBG07] et RDF-Growth [TMP+04] sont utilisés dans le contexte du partage. Par conséquent, ils ne sont

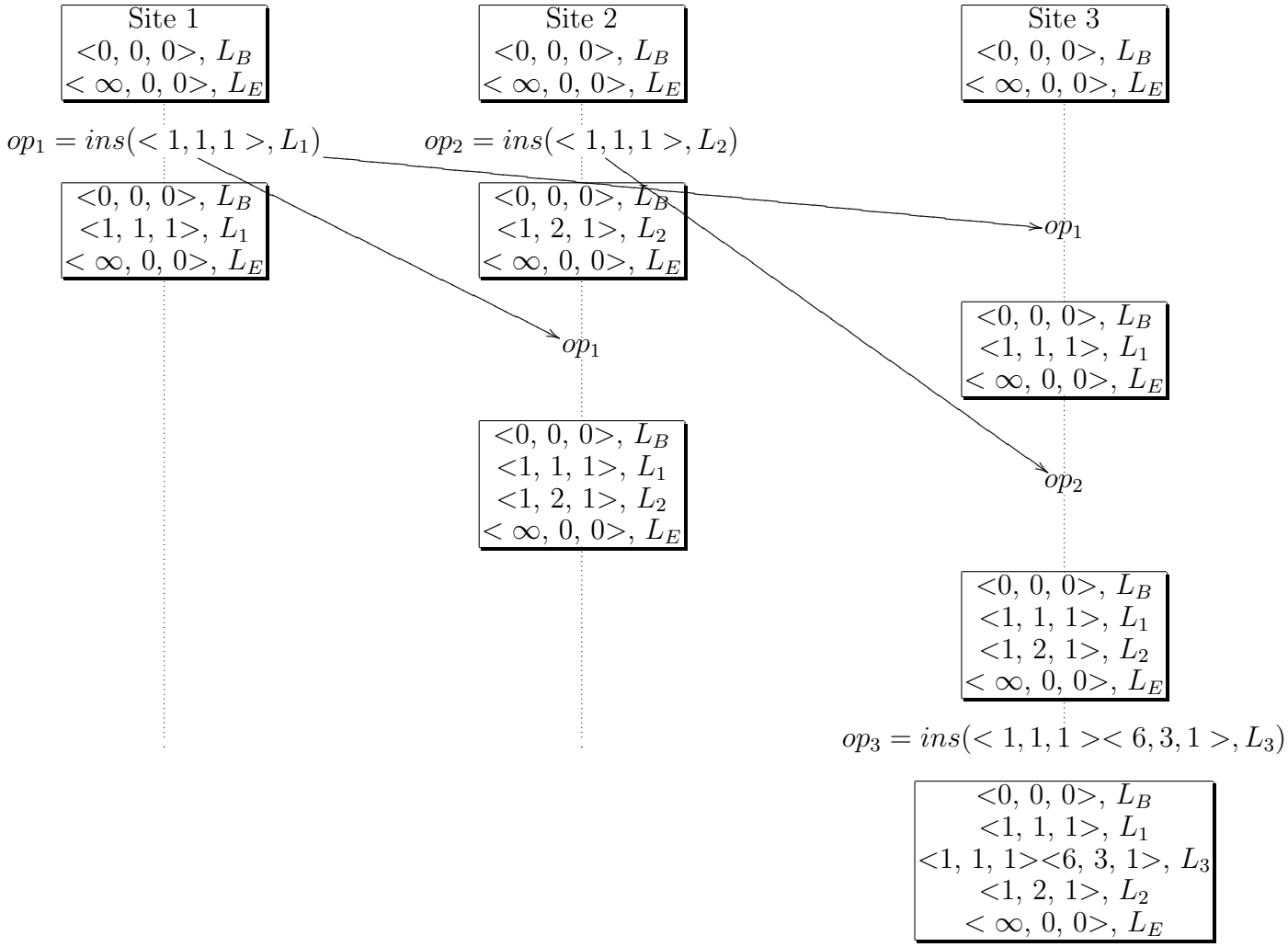

FIGURE 1.23 – Scénario d'exécution de Logoot

pas adaptés pour les wikis sémantiques pair-à-pair puisqu'ils n'assurent pas le modèle de cohérence CCI.

RDFSync

RDFSync [MTEBG07] est un algorithme qui synchronise un graphe RDF cible avec un graphe RDF source. Ces graphes sont décomposés en sous ensembles minimaux de triplets RDF (appelés Minimum Self-Contained Graphs MSGs) et représentés canoniquement par des listes ordonnées des identifiants (le hâchage) de leurs MSGs qui les composent. L'algorithme de synchronisation RDFSync calcule la différence entre la liste ordonnée des MSGs de la source et celle de la cible. RDFSync peut effectuer différents types de synchronisation :

1. Dans le premier type appelé (Target Growth Sync TGS), le graphe cible devient égal à la fusion des deux graphes ;

2. Dans le deuxième appelé (Target Erase Sync TES), le graphe cible supprime l'information inconnue par le graphe RDF source c'est-à-dire certains triplets RDF ;

3. Dans le dernier type appelé (Target Change Sync TCS), le graphe cible devient égal au graphe source.

RDFSync ne peut pas être utilisé pour synchroniser les données sémantiques dans les wikis sémantiques pair-à-pair puisqu'il a été conçu dans le contexte du partage des données sémantiques et il n'assure pas la cohérence CCI. Nous démontrons cela en déroulant le scénario suivant : Considérons deux utilisateurs sur des pairs différents qui modifient simultanément un graphe RDF initial G qui est décomposé en deux MSGs ayant une liste ordonnée de deux nombres de hâchage comme indiqué à la figure 1.24. Sur le premier site, un utilisateur insère un troisième triplet RDF, en même temps un autre utilisateur sur le deuxième site supprime le deuxième triplet RDF. Avec la stratégie TGS, le $Site_1$ a besoin de rien du $Site_2$, tandis que le $Site_2$ récupère le deuxième et le troisième triplets du premier site. Après la synchronisation, la convergence des deux sites est assurée, les deux sites convergent vers le même graphe RDF. Cependant, l'intention de l'utilisateur du site $site_2$ n'est pas préservée puisque le deuxième triplet apparaît dans le résultat final. Dans RDFSync, une suppression d'un triplet a lieu effectivement lorsque tous les pairs suppriment le même triplet au même moment (ce qui est difficile à avoir dans le contexte de l'édition collaborative surtout sur un réseau pair-à-pair). De la même façon, les autres stratégies de synchronisation produisent des résultats similaires.

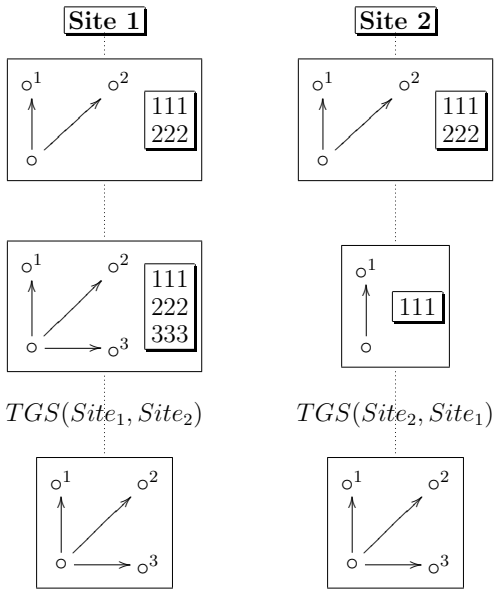

FIGURE 1.24 – Synchronisation des graphes RDF avec RDFSync

RDFGrowth

RDFGrowth [TMP+04] est un algorithme pour des applications pair-à-pair échangeant des métadonnées. Les pairs participent dans des "groupes d'intérêt" pour accroître leurs connaissances sur un ou plusieurs "sujets". Dans ce réseau pair-à-pair, chaque pair a une base de données RDF locale et utilise l'algorithme RDFGrowth pour accroître ses connaissances internes par la découverte et par l'importation des ressources d'autres pairs. RDFGrowth synchronise un ensemble d'URIs [uri] exprimant l'intérêt du groupe et les informations les concernant. Par exemple, un intérêt de groupe (IG) est de "trouver les albums d'un chanteur X". En utilisant RDFGrowth, toutes les informations concernant cette requête seront collectées et stockées sur les différents pairs intéressés par cet IG. RDFGrowth assure la convergence des bases de données RDF des pairs en cas d'absence de suppression. RDFGrowth est seulement incrémental, c'est-à-dire lorsqu'un triplet RDF est supprimé sur un pair, il ne sera pas supprimé sur les autres pairs. Par conséquent, RDFGrowth ne peut pas être utilisé pour synchroniser les données sémantiques dans un wiki sémantique pair-à-pair où les triplets RDF qui représentent les annotations sémantiques des pages wikis sont mises à jour fréquemment.

Synthèse Les recherches dans le web sémantique pair-à-pair se concentrent sur le partage et l'extraction de la connaissance. Les algorithmes d'édition collaborative des graphes RDF existants ne prennent pas en considération les modifications concurrentes des graphes RDF. Ces algorithmes ne garantissent pas le modèle de cohérence CCI nécessaire pour un éditeur collaboratif.

1.7.3 Editeurs multi-synchrones

Dans cette section, nous présentons deux exemples de systèmes de collaboration supportant l'édition multi-synchrone : SAMS et les gestionnaires des contrôles de versions distribués (DVCS). Nous les étudions et les évaluons par rapport à notre problématique.

SAMS

Synchrone-Asynchrone-MultiSynchrone (SAMS) [MSMOJ02] est le premier éditeur multi-sycnhrone. C'est un environnement collaboratif qui permet aux utilisateurs de travailler en mode synchrone, asynchrone ou multi-synchrone, tout en assurant la cohérence des données partagées. SAMS est basé sur une architecture client/serveur. Le serveur est utilisé comme un estampilleur global. Il manipule des objets typés et les opérations de modification de ces objets sont stockées dans une historique. Chaque site possède son propre historique. Lorsqu'un utilisateur modifie sa copie locale, les opérations générées sont stockées dans l'historique local. Ces opérations seront propagées vers d'autres sites après avoir intégré localement les opérations concurrentes d'autres sites qui se trouvent déjà sur le serveur. Quand une opération est reçue, l'état local de l'objet partagé peut être différent

de celui observé lors de sa génération. Le mécanisme d'intégration de SAMS transforme l'opération distante lorsque c'est nécessaire afin d'être fusionnée avec les opérations locales. SAMS utilise SOCT4 [VCFS00] un algorithme de transformation opérationnelle pour maintenir la cohérence des données partagées et il assure la cohérence CCI.

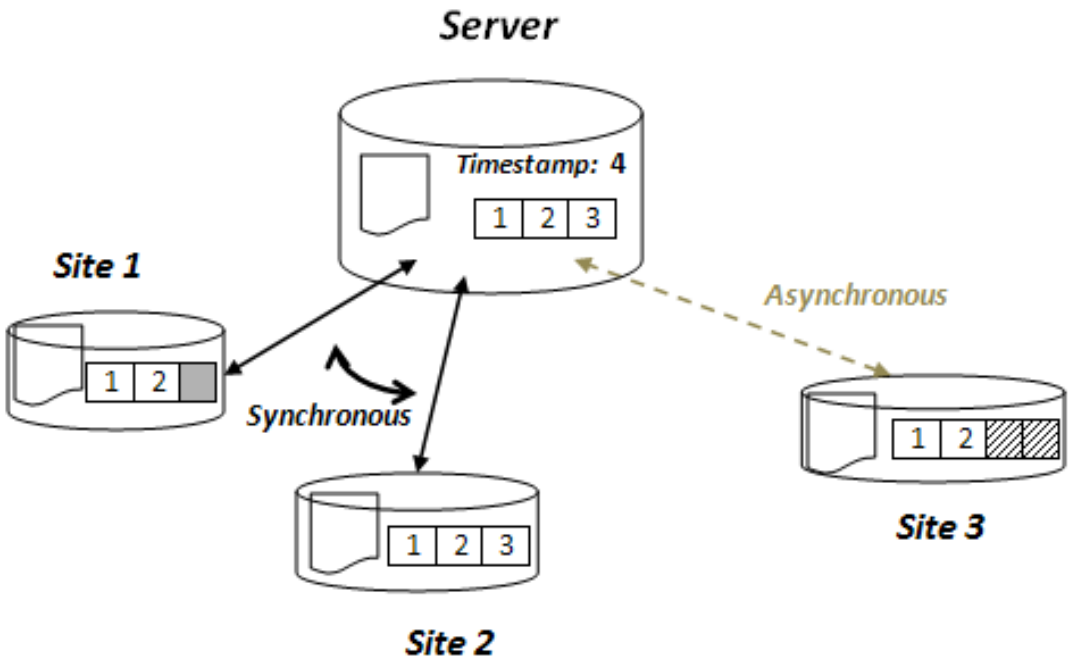

FIGURE 1.25 – Scénario de collaboration dans SAMS

Nous présentons le fonctionnement de SAMS en déroulant un scénario. Considérons trois sites Site 1, Site 2 et Site 3 qui partagent un document. Chaque site possède une copie du document et une historique locale des opérations. Supposons que les deux premiers sites décident de travailler en mode synchrone, tandis que le troisième décide de travailler en asynchrone. Supposons que l'état initial du document est le résultat de l'exécution des deux opérations op1 et op2.

- Lorsqu'un utilisateur modifie le document sur le Site 2, une opération op3 est générée. Après avoir été exécutée, op3 est rangée à la fin de l'historique à l'emplacement 3 puisqu'elle a réussi à avoir l'estampille numéro 3 du serveur. Ensuite, elle est propagée à l'ensemble des sites. Le Site 1 qui est en mode synchrone recevra cette opération directement, par contre Site 3 la recevra plus tard.
- Supposons que sur le Site 1, un utilisateur a modifié en parallèle le document, une opération op est générée. Pour que op soit placée à la fin de l'historique du Site 1, elle doit avoir l'estampille numéro 3. Puisque cette estampille a changé côté serveur, op ne peut pas être placée à cette position. Une intégration de l'opération op3 doit

précéder cela. Puisque les opérations op et op3 sont concurrentes, l'algorithme d'intégration SOCT4 utilisé dans SAMS appelle des fonctions de transformation (OT) pour transformer ces opérations. Les transformations sont effectuées pour garantir que (1) op3 sera exécutée sur le nouvel état du document intégrant déjà op et (2) assurer une cohérence de l'historique entre les deux sites Site 1 et Site 2 en plaçant op3 à la position 3 dans l'historique du Site 1. Une demande d'une nouvelle estampille sera effectuée et op sera placée dans l'historique à la position 4. Ensuite, op est propagée à tous les sites.

– Sur le Site 3, un utilisateur modifie le document en mode asynchrone, ce qui génère plusieurs opérations. Sa copie du document diverge avec celle d'autres sites. Ces opérations ne peuvent pas être envoyées aux autres sites avant que les opérations op3 et op soient intégrées.

SAMS assure la cohérence CCI. Cependant, il est basé sur une architecture client-serveur donc il souffre des problèmes liés à cette architecture. L'édition collaborative entre les différents sites que ce soit en mode synchrone, asynchrone ou multi-synchrone passe toujours par le serveur.

Gestionnaires de contrôles de versions distribués

Les systèmes de contrôles de versions distribués (DVCS) sont des systèmes de collaboration pair-à-pair. Ils permettent aux utilisateurs distribués dans le temps et dans l'espace de développer progressivement des logiciels. Ces systèmes sont conçus pour la communauté des développeurs pour la gestion du code source. Il existe actuellement plusieurs DVCS tels que Git [AFK+95], Mercurial [Mer], Darcs[Dar], Bazaar [Baz], etc. Le modèle de coopération des DVCS est basé sur trois primitives : push, pull et clone. Nous présentons ce modèle au moyen du scénario suivant :

– Dans un premier temps, un utilisateur $U1$ crée un projet dans un espace de travail "Working directory" sur sa machine (voir figure 1.26). Il décide d'utiliser un DVCS, soit Mercurial, pour suivre les états de son projet.
– Il intialise Mercurial de son espace de travail par une commande "hg init", cela a pour effet de créer un "Private repository" local dans l'espace du travail.
– $U1$ utilise la commande "hg add *" pour ajouter (ou tracker) les fichiers à gérer. Ces fichiers sont maintenant hâchés et indexés. Ensuite, il fait un commit avec la commande "hg commit -m 'commit initial'". Des blobs des fichiers indexés, un arbre de ces fichiers et un objet commit sont donc créés dans le "repository". Ils sont des objets non mutables. $U1$ continue à modifier les fichiers du projet, à faire des commits lorsqu'il veut sauvegarder une version courante du projet et à publier ses changements.
– Pour collaborer avec $U2$, $U1$ crée un deuxième repository "Public repository" sur un serveur ayant l'adresse $URL1$ en clonant le premier par la commande "hg clone URL1". Ce deuxième repository est placé sur le serveur public $URL1$ pour être

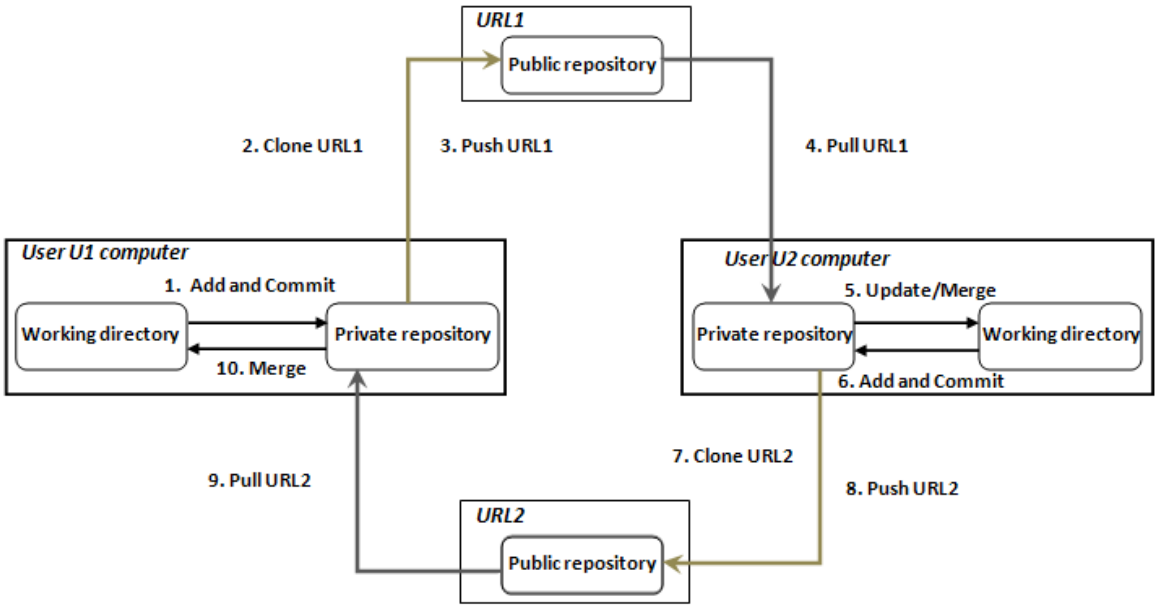

FIGURE 1.26 – Le modèle de coordination Push/Pull/Clone dans les DVCS

accédé par $U2$. $U1$ peut créer plusieurs "repository" publics et les placer dans des endroits différents.

- Ensuite, $U1$ avertit $U2$ de la disponibilité de ce repository par mail. $U2$ récupère ce repository en le clonant à son tour. Cela lui crée un repository privé sur sa machine personnelle. $U2$ commence à l'éditer localement.
- Pour renvoyer ses modifications à $U1$, $U2$ crée à son tour un repository public en clonant son repository et avertit $U1$ de la disponibilité de ce repository. Puis, il publie ces changements en utilisant la commande "hg push URL2 :public repository".
- $U1$ alors récupère les modifications de $U2$ par un "hg pull URL2" suivi éventuellement d'un merge (certain DVCS comme Git fusionnent les changements automatiquement, d'autres non). A ce stade, la coopération entre les deux utilisateurs est établie.
- Toute modification ultérieure suit un cycle de push/pull. Si $U1$ fait des modifications, il les rend publiques en faisant un push sur $URL1$. $U1$ intègre les modifications de $U2$ à tout moment en faisant un pull de URL2.

Dans les DVCS, les utilisateurs décident avec qui échanger et quand publier leur travail. Par conséquent, les répertoires des utilisateurs ne contiennent pas nécessairement les mêmes données. Pour l'intégration des changements distants dans l'espace de travail d'un utilisateur, les DVCS soit ils utilisent un algorithme de fusion intégré tel que Git, soit ils

délèguent cette tâche à des algorithmes de fusion en dehors du système. Aucune garantie sur le résultat de la fusion n'est donc fournie. Git par exemple utilise l'algorithme de fusion "3-way merge" pour la fusion des versions. Il assure donc une fusion semi-automatique. En cas de conflit d'édition, il génère des blocs de conflit. La résolution de ces conflits et la validation des nouvelles versions seront faites par l'utilisateur propriétaire de l'espace du travail. Les DVCS ne s'intéressent pas à maintenir la cohérence des données répliquées dans les différents répertoires, ils assurent seulement la cohérence causale.

Dans les DVCS, les données sont répliquées sur les machines des utilisateurs. Donc l'édition déconnectée est naturellement supportée et la connexion internet est seulement utilisée pendant l'échange des données entre les pairs. Les DVCS supportent l'édition multi-synchrone et fournissent un mécanisme d'annulation de groupe. Les DVCS implémentent plusieurs procédés d'édition collaborative. Nous présentons un de ces procédés implémenté dans Git (voir figure 1.27). Dans ce procédé, chaque développeur possède un répertoire public, par contre un seul est considéré comme le répertoire officiel. Le "blessed repository" est utilisé pour créer des packages et des fichiers binaires. Une personne ou une équipe a le droit de publier dans ce répertoire, tandis que les développeurs autorisés peuvent copier des fichiers de ce répertoire. Lorsque les développeurs modifient des fichiers et veulent les publier, ils les envoient au gestionnaire de l'intégration. Enfin, ce sont les personnes autorisées à gérer le gestionnaire d'intégration qui fusionnent les changements, les testent et les publient dans le répertoire officiel.

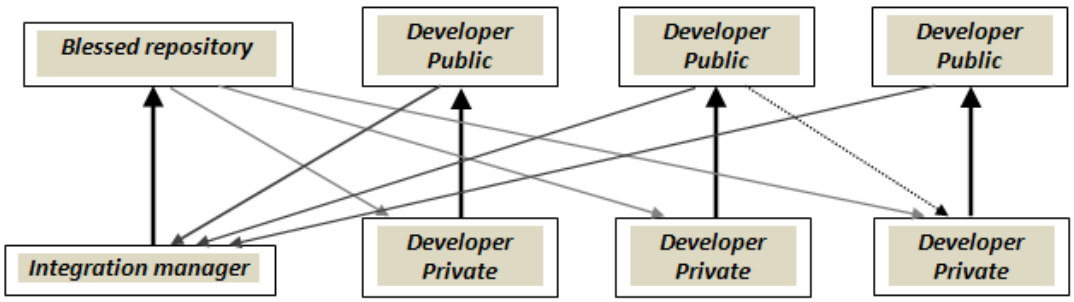

FIGURE 1.27 – Un procédé d'édition collaborative implémenté dans Git [git]

En dépit de leurs avantages, les DVCS n'assurent que la cohérence causale. Ils sont conçus pour la gestion du code et ne gèrent pas des annotations sémantiques.

1.7.4 Wikis pair-à-pair

Les wikis sur réseaux pair-à-pair (P2P) sont une autre évolution des systèmes wikis. Ils tentent de concilier les avantages de l'édition collaborative et les qualités intrinsèques des

réseaux pair-à-pair[ATS04] et d'adresser les problèmes des systèmes centralisés tels que les wikis traditionnels. Les wikis P2P répliquent des pages wikis sur des réseaux pair-à-pair structurés ou non structurés. La réplication des pages wikis peut être en un nombre limité de fois ou sur tous les pairs du réseau. Certains d'entre eux se concentrent sur les aspects de communication et de réplication, mais ils n'adressent pas proprement les mises à jour concurrentes. D'autres gèrent les modifications concurrentes d'une manière efficace. Les wikis pair-à-pair peuvent être classifiés en deux catégories :

Wikis sur réseaux P2P structurés Les wikis pair-à-pair appartenant à cette catégorie sont basés soit sur des tables de hâchage distribuées (DHT) [6] [MM02], soit sur un système de fichiers distribués [7]. Ils répliquent les pages wikis sur cette architecture en garantissant de trouver les pages wikis sur ces pairs en un certain nombre de sauts lorsqu'elles existent. En général, ces wikis se concentrent sur les aspects de communication et de réplication, mais ils n'adressent pas proprement les mises à jour concurrentes. Tous ces wikis assurent seulement la cohérence causale à part UniWiki [OMMD10] qui assure en plus la cohérence éventuelle. Il existe plusieurs wikis pair-à-pair appartenant à cette catégorie tels que DistriWiki [Mor07], Piki [MLS08], DtWiki [DB08], UniWiki et un wiki basé sur Scalaris [sca].

Wikis sur réseaux P2P non structurés Dans ces wikis, les liens entre les pairs sont construits d'une façon arbitraire. En général, ils répliquent les pages wikis sur tous les pairs. Un changement sur une page est envoyé aux autres pairs afin d'être intégré. Ils gèrent les modifications concurrentes d'une manière efficace. La plupart d'entre eux assurent la cohérence causale, seulement Wooki [WUM07] assure le modèle CCI sur les pages wikis répliquées. Des wikis appartenant à cette catégorie sont GitWiki [git], Repliwiki [KWK03] et Wooki.

Dans la suite de cette section, nous présentons le principe de fonctionnement de deux wikis P2P et nous concluons par une synthèse sur les wikis P2P.

UniWiki

L'idée de base d'UniWiki [OMMD10] est de créer un wiki P2P reposant sur une table de hâchage distribuée (DHT) combinée avec un mécanisme de réplication optimiste. Concrètement, UniWiki utilise un système DHT comme un mécanisme de stockage distribué des pages wikis et y intègre directement l'algorithme de réplication WOOT [OUMI06] pour synchroniser les pages.

UniWiki est constitué de deux composants, comme illustré dans la figure 1.28 : la DHT qui est responsable du stockage des pages wikis et des frontaux (wiki front-end) responsables de traiter les requêtes des clients. Chaque pair de la DHT est responsable d'un sous ensemble de pages wikis. Une page wiki est une séquence de lignes de Woot

6. http ://pdos.csail.mit.edu/chord/
7. http ://tier.cs.berkeley.edu/wiki/TierStore

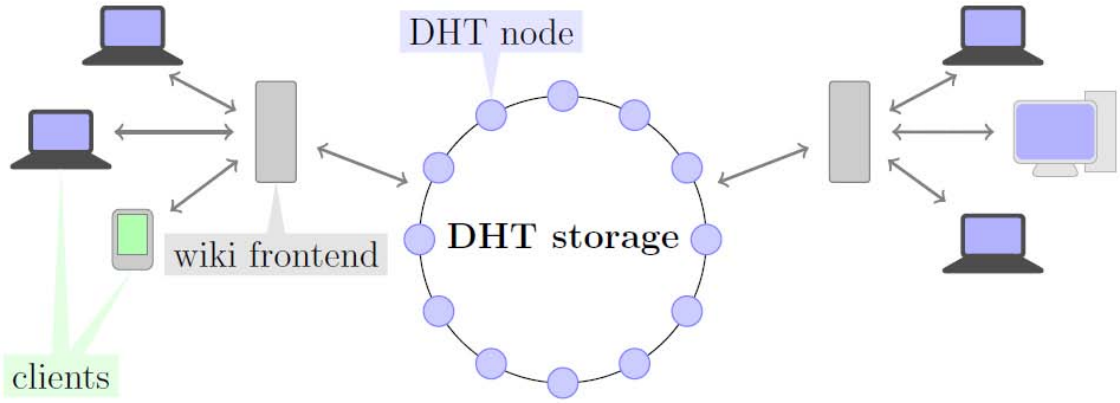

FIGURE 1.28 – L'architecture d'UniWiki extrait de [OMMD10]

(voir la section 1.7.1). Les requêtes des clients se résument en (1) une visualisation, (2) une édition et (3) une sauvegarde d'une page wiki. Ces requêtes sont traitées sur le côté frontal du wiki et sur des pairs de la DHT :

Côté serveur frontal wiki

- Quand un utilisateur (client) veut regarder une page wiki spécifique. Le serveur frontal du wiki récupère la page Woot correspondante de la DHT, en utilisant le hâchage de l'URI de la page demandée comme la clé. Ensuite, cette page est transformée en contenu wiki et rendu au format HTML si nécessaire, puis envoyée au client.
- Quand un client demande une page pour l'éditer, la page WOOT extraite de la DHT est stockée dans la session d'édition courante, de sorte que les changements effectués par l'utilisateur peuvent être correctement détectés. Ensuite, le contenu de cette page est envoyé au client pour l'éditer.
- Quand un client termine sa session d'édition et sauvegarde ses modifications, l'ancienne version du contenu wiki est extraite de la session d'édition en cours. Un algorithme de différence textuel calcule les modifications entre l'ancienne version et la nouvelle version éditée par le client. Les opérations correspondantes sont générées et envoyées à la DHT.

Côté pair DHT UniWiki modifie les méthodes **get** et **put** de la DHT pour qu'elles soient conformes à son architecture. Leur comportement diffère du comportement de base fourni par toute DHT car le type de la valeur retournée par une requête get (qui est une page Woot) n'est pas le même que celui (une liste d'opérations de Woot) stocké par la requête put.

- Requête **onGet** : quand une requête **get** est reçue par un pair de la DHT (celui qui est responsable du contenu wiki identifié par la clé ciblée), la méthode *onGet(K)*

est exécutée. Le modèle de la page Woot qui correspond à la clé K est extrait du stockage local, une page Woot simplifiée ne contenant que les lignes visibles est calculée, puis renvoyée au serveur frontal.

– Requête **onPut** : quand un pair DHT doit répondre à une requête **put**, la méthode *onPut(K, WootOps[])* est déclenchée. Premièrement, le modèle de la page Woot ciblé est récupéré du stockage local. Ensuite, chaque opération reçue par la requête est intégrée dans le modèle de la page WOOT et enregistrée dans l'histoire de cette page.

UniWiki supporte un mécanisme automatique de réplication de pages wikis. Ce mécanisme est responsable d'une part de la réplication des nouvelles pages wikis et d'autre part de répliquer certaines pages lorsque leur nombre devient critique ou lorsque de nouveaux pairs joignent la DHT. UniWiki assure le modèle de cohérence CCI des pages répliquées sur la DHT.

Repliwiki

RepliWiki [KWK03] est un wiki P2P qui utilise une réplication optimiste basée sur l'approche Summary Hash History (SHH) [Kan]. Il vise fournir un moyen efficace pour répliquer des wikis à grande échelle comme Wikipedia, afin de distribuer la charge et supporter la tolérance aux pannes. Dans Repliwiki, chaque pair héberge une réplique de toutes les pages wikis. Chaque page wiki est associée à un graphe d'histoire SSH qui enregistre les versions de la page et les dépendances causales entre elles. Dans ce graphe, une version V_i de la page est représentée par un identifiant S_i (Summary Hash) obtenu en utilisant une fonction de hâchage. L'identifiant S_i est calculé comme suit : soit $H_i = h(V_i)$, où h est une fonction de hâchage. $S_i = h(S_p||H_i)$ lorsque V_i a un seul prédécesseur où S_p est l'identifiant de la version précédente ou bien $S_i = h(S_p^*||H_i)$ lorsque V_i possède plusieurs prédécesseurs (V_i a été obtenue par fusion de plusieurs versions concurrentes). S_p^* est la concaténation des identifiants S_p ordonnés lexicographiquement. Pour supporter une propagation des changements qui passe à l'échelle, Repliwiki utilise le mécanisme anti-entropie. Durant la synchronisation, deux sites échangent leur SHH à partir duquel chacun extrait l'ensemble minimal de mises à jour qui doivent être propagées à travers le réseau et qui sont nécessaires pour rendre les deux sites dans un état mutuellement cohérent. En utilisant l'anti-entropie avec l'approche SHH, Repliwiki visait remplacer la propagation avec les vecteurs de versions qui est inefficace. Il détecte correctement la causalité et la concurrence et assure la cohérence causale. RepliWiki spécifie que tout algorithme de synchronisation à utiliser dans l'approche SHH doit préserver une ou plusieurs propriétés pour assurer la cohérence éventuelle des pages wikis répliquées. L'algorithme de fusion doit être déterministe, c'est-à-dire il doit produire le même résultat de fusion indépendant du site qui effectue la fusion, être commutative et associative. Aucune indication sur l'algorithme de synchronisation utilisé dans RepliWiki n'a été donnée.

Synthèse

Dans les wikis P2P structurés existants, les pages wikis sont répliquées partiellement sur les différents pairs du réseau structuré. Ces wikis adressent le problème du passage à l'échelle des wikis centralisés et supportent la tolérance aux pannes. Ils offrent un grand espace de stockage distribué et génèrent peu de trafic lors du transfert des changements des pages entre les pairs. En général, les wikis P2P structurés se concentrent sur les aspects de communication et de réplication. Actuellement, l'édition multi-synchrone n'est pas gérée dans ces wikis. En effet, chaque utilisateur accède à la même copie d'une page wiki. Les pages wikis que les utilisateurs souhaitent éditer ne sont pas nécessairement stockées sur leur machine. Ces wikis ne gèrent pas des procédés.

Dans les wikis P2P non structurés, le réseau est construit en suivant des relations sociales entre les utilisateurs. Ces wikis permettent donc de supporter des procédés d'édition collaborative. Les wikis P2P non structurés actuels utilisent une anti-entropie pour propager efficacement les changements des pages entre les pairs. La plupart de ces wikis répliquent les pages sur tous les pairs. Cette réplication de pages permet une édition déconnectée des pairs. Dans les wikis P2P non structurés, plusieurs copies des pages sont éditées en concurrence sur les différents pairs. Certains wikis P2P non structurés tels que Git-Wiki [git] et Repliwiki [KWK03] permettent aux utilisateurs de choisir le moment de synchronisation de leur pair. Par conséquent, ils supportent l'édition multi-synchrone. Ces wikis gèrent les modifications concurrentes des pages wikis d'une manière efficace. En conclusion, les wikis P2P structurés ne s'intéressent pas à supporter l'édition déconnectée, l'édition multi-synchrone et les procédés. Par contre, certains de ces points sont déjà supportés dans les wikis P2P non structurés.

1.8 Synthèse générale

Dans ce chapitre, nous avons présenté l'état de l'art des wikis et des wikis sémantiques. Ensuite, nous avons présenté différentes approches, modèles, systèmes et algorithmes basés sur la réplication optimiste qui sont en lien avec notre problématique.

- Les wikis et les wikis sémantiques souffrent des limitations de l'architecture centralisée telles que le passage à l'échelle et la disponibilité des données. Ils ne supportent pas les procédés de type workflows orientées données, ni l'édition déconnectée, ni l'édition multi-synchrone.
- Pour les approches répliquées, nous avons étudié les approches suivantes :
    1. Les approches qui permettent de synchroniser des données textuelles sur le réseau P2P en assurant une cohérence CCI. Ces approches peuvent être utilisées dans la synchronisation des pages wikis dans un sémantique P2P.
    2. Les approches de réplication dans le web sémantique P2P se concentrent sur le partage et l'extraction de la connaissance. Leurs algorithmes de synchronisa-

tion ne garantissent pas le modèle de cohérence CCI nécessaire pour les wikis sémantiques P2P.

3. Les éditeurs collaboratifs multi-synchrones actuels ne gèrent pas des données sémantiques. De plus, SAMS [MSMOJ02] souffre du problème de passage à l'échelle, alors que les DVCS n'assurent pas la cohérence CCI nécessaire dans un système d'édition collaborative.
4. Les wikis P2P adressent les problèmes de l'architecture centralisée des wikis, ils ne gèrent pas des données sémantiques. Les wikis P2P structurés actuels ne s'intéressent pas à supporter l'édition déconnectée, l'édition multi-synchrone et les procédés. Par contre, certains de ces points sont déjà supportés dans les wikis P2P non structurés.

Chapitre 2

Swooki : Un wiki sémantique sur réseau pair-à-pair

Sommaire

Dans ce chapitre, nous proposons Swooki une instanciation du modèle de réplication optimiste pour les wikis sémantiques. Soit N serveurs wikis sémantiques interconnectés qui répliquent des pages wikis contenant des annotations sémantiques. Quand un pair modifie sa copie locale des données répliquées, il génère une opération correspondante.

- Dans ce chapitre, nous définissons les opérations de modification du type de données (texte et annotations sémantiques). Afin d'assurer le modèle de cohérence CCI, nous définissons les intentions de ces opérations.
- Pour garantir que les changements envoyés seront éventuellement livrés à tous les pairs, nous proposons d'utiliser l'algorithme Lpbcast [EGH+03] pour envoyer les modifications à tous les pairs connectés et un mécanisme d'anti-entropie [DGH+88] pour supporter le mode déconnecté. Cette combinaison a déjà été présentée dans [WUM07].
- Pour synchroniser ce nouveau type de données, nous choisissons Woot [OUMI06] un algorithme de réplication conçu initialement pour synchroniser du texte. Nous le modifions pour synchroniser le nouveau type de données tout en préservant l'intention des opérations et la convergence des pages et des annotations des pairs.

Nous montrons que cette instanciation nommée Swooki assure le modèle de cohérence CCI sur les répliques des pairs qui sont les pages wikis contenant des annotations sémantiques.

Dans la section 2.1, nous commençons par expliquer comment le système Swooki fonctionne du point de vue utilisateur. Puis nous présentons son architecture et son implémentation dans la section 2.2. Dans les sections 2.3, 2.4 et 2.5, nous décrivons son modèle de données, les opérations d'édition et son mécanisme de propagation des changements. Ensuite, nous présentons le modèle de cohérence CCI assuré dans Swooki dans la section 2.6 et son algorithme de synchronisation des répliques dans la section 2.9.2. Enfin, nous terminons par une présentation du mécanisme d'annulation de groupe de Swooki dans la section 2.9.

2.1 Utilisation de Swooki

Swooki est un wiki sémantique distribué sur un réseau pair-à-pair non structuré. Il est composé d'un ensemble de serveurs wikis sémantiques interconnectés (voir figure 2.1). Chacun de ces serveurs appelé aussi pair ou nœud agit à la fois comme un client et serveur qui peut joindre ou quitter le réseau dynamiquement sans affecter le bon fonctionnement du wiki sémantique pair-à-pair.

Chaque pair de Swooki héberge une copie des pages wikis sémantiques et un entrepôt RDF pour stocker les données sémantiques de ces pages. Un pair de Swooki agit comme un serveur Semantic MediaWiki [VKV+06]. Il permet l'édition des annotations dans le contenu des pages wikis sémantiques. Ainsi, ces annotations seront utilisées pour améliorer la recherche et la navigation dans Swooki et pour générer dynamiquement du contenu dans les pages wikis sémantiques. Ces annotations peuvent être extraites et réutilisées par d'autres systèmes en dehors de Swooki.

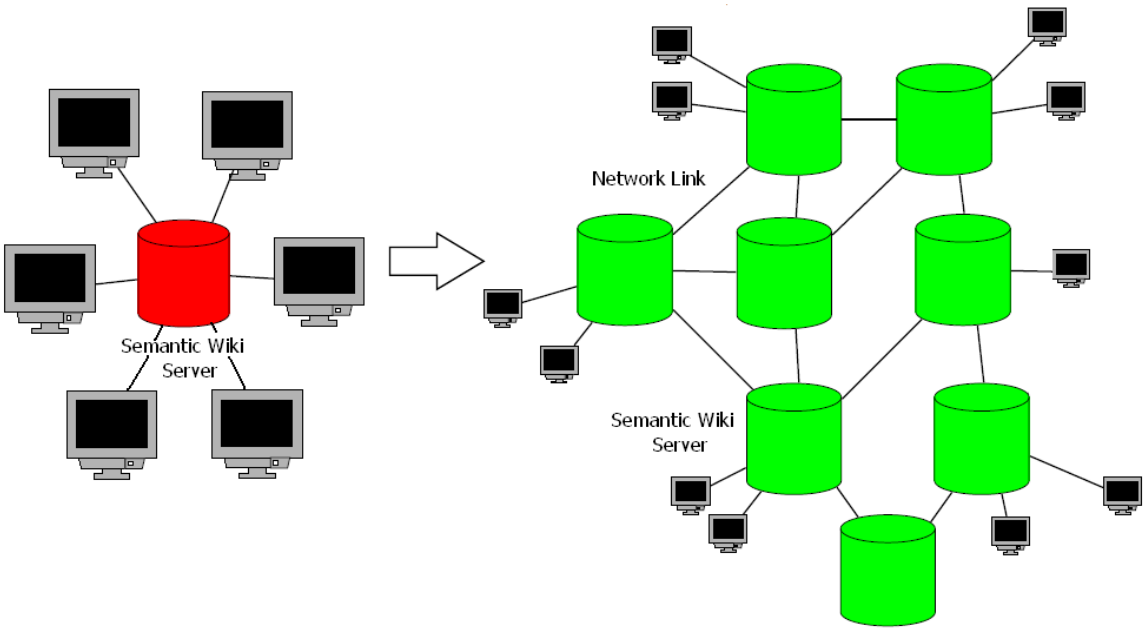

FIGURE 2.1 – Swooki : un wiki sémantique pair-à-pair

La gestion des données dans Swooki est basée sur une réplication optimiste des données (du contenu des pages wikis sémantiques et de leurs annotations) et qui assure le modèle de cohérence CCI sur ces données. Lorsqu'un pair met à jour une copie locale de données, il génère des opérations correspondantes. Une opération est effectuée en quatre étapes :

- elle est exécutée directement sur la copie locale du pair,
- propagée vers tous les autres pairs à travers le réseau,
- reçue par les pairs et
- intégrée sur chacune des répliques de ces pairs. Lorsque c'est nécessaire le procédé d'intégration fusionne une modification avec celles qui sont concurrentes générées par des pairs distants ou localement.

Grâce à son architecture pair-à-pair et à sa réplication optimiste des données, Swooki supporte la tolérance aux pannes, une meilleure performance, le passage à l'échelle, la résistance à la censure et la distribution de la charge. Il permet également de supporter des modes d'édition tels que l'édition déconnectée et l'édition multi-synchrone.

Pour joindre le réseau dans Swooki, un pair P_1 doit connaître l'identifiant d'un des pairs du réseau, par exemple P_2, et de l'ajouter dans sa liste des voisins ou bien inversement, P_1 peut être ajouté dans le réseau comme voisin par un des pairs P_2 qui connait son identifiant (voir figure 2.2). Une fois que P_1 a rejoint le réseau, les pages wikis sémantiques et les annotations du pair voisin P_2 seront répliquées chez lui. Ensuite, les deux pairs P_1 et P_2 commencent à collaborer. Nous présentons trois types d'édition possibles dans Swooki entre ces deux pairs :

Edition séquentielle Le pair P_2 édite une page insère une nouvelle ligne et sauvegarde

FIGURE 2.2 – Table de voisinage dans Swooki

(voir figure 2.3). Les changements sont directement propagés aux autres pairs du réseau. Ensuite ils sont intégrés dans le contenu de la page et dans l'entrepôt RDF si nécessaire. En actualisant la page, ces changements seront visibles sur P_1. Tous les pairs de Swooki convergent vers le même état.

Edition concurrente Les deux pairs P_1 et P_2 éditent en parallèle la même page (voir figure 2.4), ils insèrent chacun une nouvelle ligne à la même position et sauvegardent. Donc les copies de la page wiki sémantique des pairs ainsi que leur entrepôt divergent pour une courte durée, mais les changements Op_1 et Op_2 sont propagés directement après la sauvegarde rapidement à travers le réseau. Après intégration de tous les changements sur les pairs, tous les pairs de Swooki finiront par converger vers le même état.

Edition déconnectée En partant de la page initiale (voir figure 2.5), P_2 décide de se déconnecter du réseau afin de travailler en isolation. Il modifie en parallèle la page avec P_1. Les deux pairs P_1 et P_2 génèrent respectivement les modifications Op_1 et Op_2 et sauvegardent. Op_1 est propagée directement à tous les pairs connectés, ce qui exclut donc P_2, tandis que Op_2 est mise en attente sur P_2 le temps que le pair

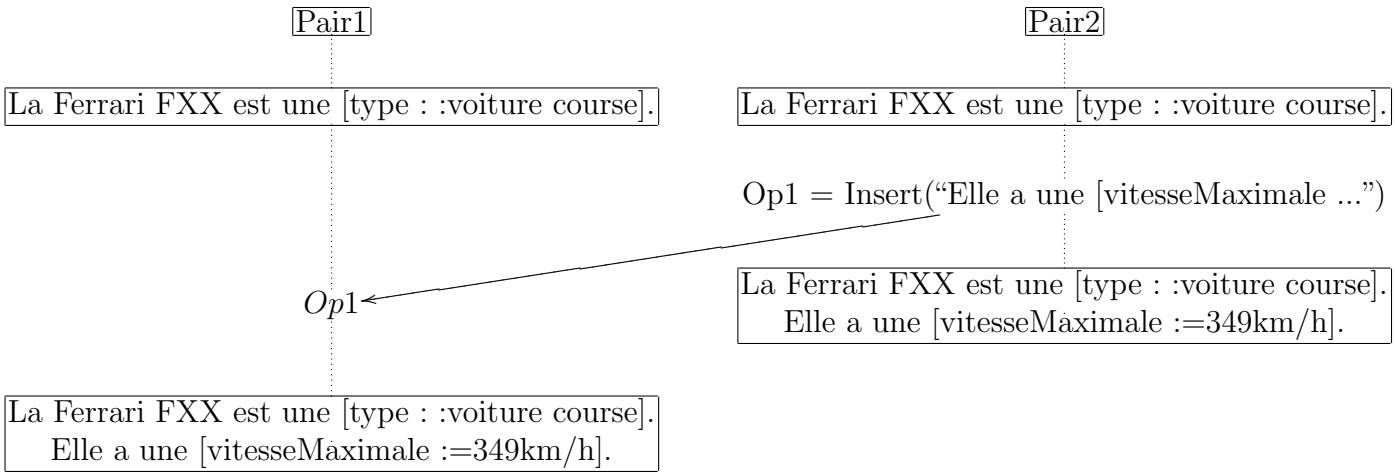

FIGURE 2.3 – Edition séquentielle

se reconnecte. La copie de la page sur P_2 et son entrepôt divergent avec ceux des autres pairs du réseau, P_2 décide le moment de synchroniser en se reconnectant. Une fois P_2 reconnecté, Op_2 sera propagée à tous les pairs. Afin de récupérer les changements des autres pairs, P_2 sélectionne un pair aléatoire dans ces voisins (par exemple P_1) et recopie les changements manquants. Après réception et intégration de tous les changements, tous les pairs de Swooki convergent.

Dans tous les cas d'édition présentés ci-dessus, Swooki assure la convergence des pages wikis sémantiques et des entrepôts des pairs tout en préservant l'intention des utilisateurs sur ces données. Les changements des lignes ainsi de leurs annotations ont eu lieu dans les pages wikis sémantiques et dans les entrepôts des pairs indépendamment de la concurrence.

2.2 Architecture et implémentation de Swooki

Dans cette section, nous présentons l'architecture de Swooki et son implémentation qui étend le wiki P2P Wooki [WUM07]. Un pair Swooki est constitué des composants suivants (voir figure 2.6) :

Interface Utilisateur. Le composant Interface Utilisateur de Swooki (voir figure 2.7) est un éditeur wiki régulier. Il permet aux utilisateurs d'éditer la vue d'une page wiki sémantique en obtenant cette page du gestionnaire de Swooki. Les utilisateurs peuvent ajouter dans leur liste des nouveaux voisins pour travailler avec eux et peuvent déconnecter leur pair afin de travailler en mode déconnecté. En plus, cette interface permet aux utilisateurs de voir l'historique d'une page qui contient l'ensemble des changements sur cette page, d'annuler des changements à partir de l'historique, d'exécuter des requêtes sémantiques et d'exporter les annotations sémantiques des pages wikis dans un format RDF.

Gestionnaire de Swooki. Ce gestionnaire est responsable de la génération et de l'intégration des patches d'édition. Un patch est l'ensemble d'opérations d'insertion et

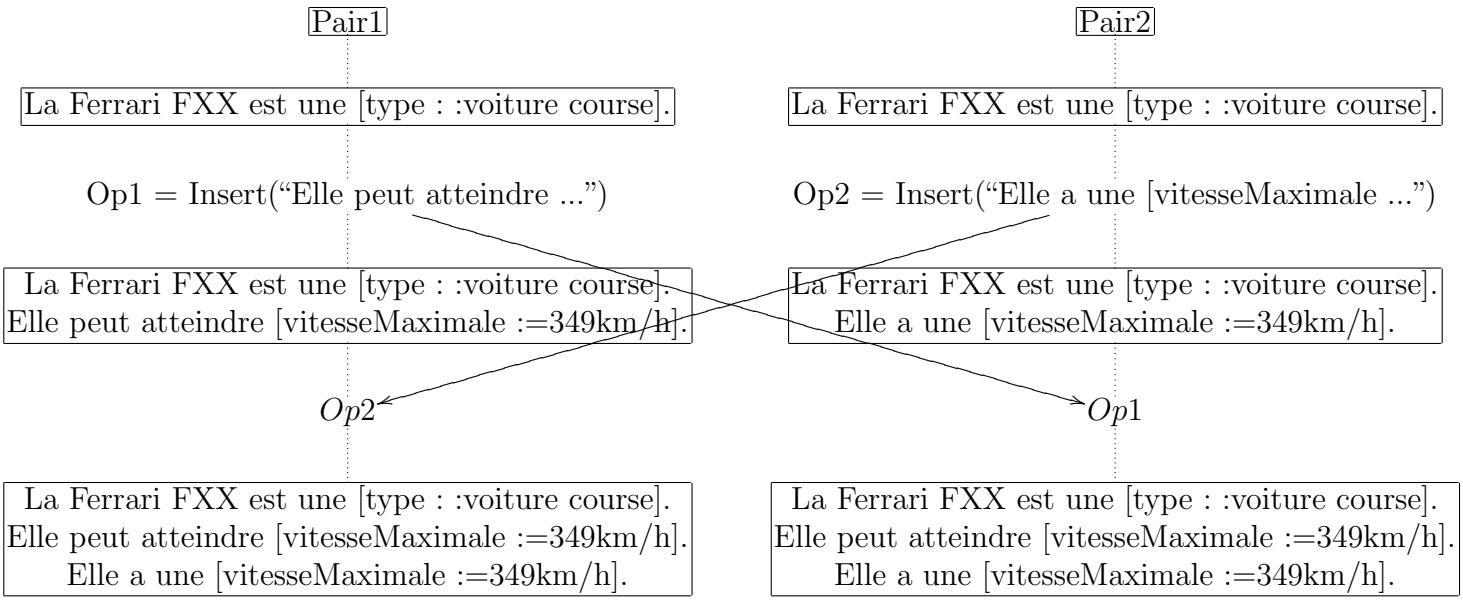

FIGURE 2.4 – Edition concurrente

de suppression sur une page wiki sémantique. Ce gestionnaire implémente l'algorithme de réplication de Swooki. La demande de modification d'une page ou la résolution d'une requête sémantique dans l'entrepôt RDF du pair passe par ce composant.

Entrepôt Sesame. L'entrepôt RDF utilisé dans Swooki est Sesame 2.0 [BKvH02]. Il est contrôlé par le gestionnaire de Swooki pour le stockage et l'extraction des triplets RDF. Nous avons utilisé une facilité de Sesame pour représenter les triplets dans cet entrepôt sous forme d'un multi-ensemble. Nous justifions le besoin de ce changement dans l'entrepôt à la section 2.6. Ce composant permet également la génération du contenu dynamique pour les pages wikis en utilisant des requêtes intégrées dans le contenu de ces pages. Il fournit également une fonctionnalité pour exécuter des requêtes sémantiques et d'exporter des graphes RDF.

Gestionnaire de Diffusion. Ce composant est responsable de la diffusion des patches d'opérations à travers le réseau. Il maintient l'adhésion des différents pairs dans le réseau pair-à-pair (P2P) non structuré. Il implémente un algorithme de diffusion probabiliste léger Lpbcast [EGH+03] fiable avec un algorithme d'anti-entropie [DGH+88]. Le premier diffuse rapidement les mises à jour sur le réseau P2P et gère l'adhésion des pairs. Le second récupère les mises à jour manquantes pour les pairs qui ont été déconnectés ou en panne. La propagation des changements dans Swooki est détaillée à la section 2.5.

Composant d'annulation de groupe. Ce composant permet aux utilisateurs d'annuler *undo* et de refaire *redo* des changements sur les pages wikis sémantiques et par conséquent sur leurs annotations dans l'entrepôt. L'annulation se fait au moyen de l'historique ou bien en utilisant une interface spéciale présentée à la figure 2.8. Ce mécanisme fournit une annulation correcte qui respecte la définition de l'annulation qui fait revenir le système à un état comme si le changement n'a pas eu lieu et respecte la cohérence CCI sur

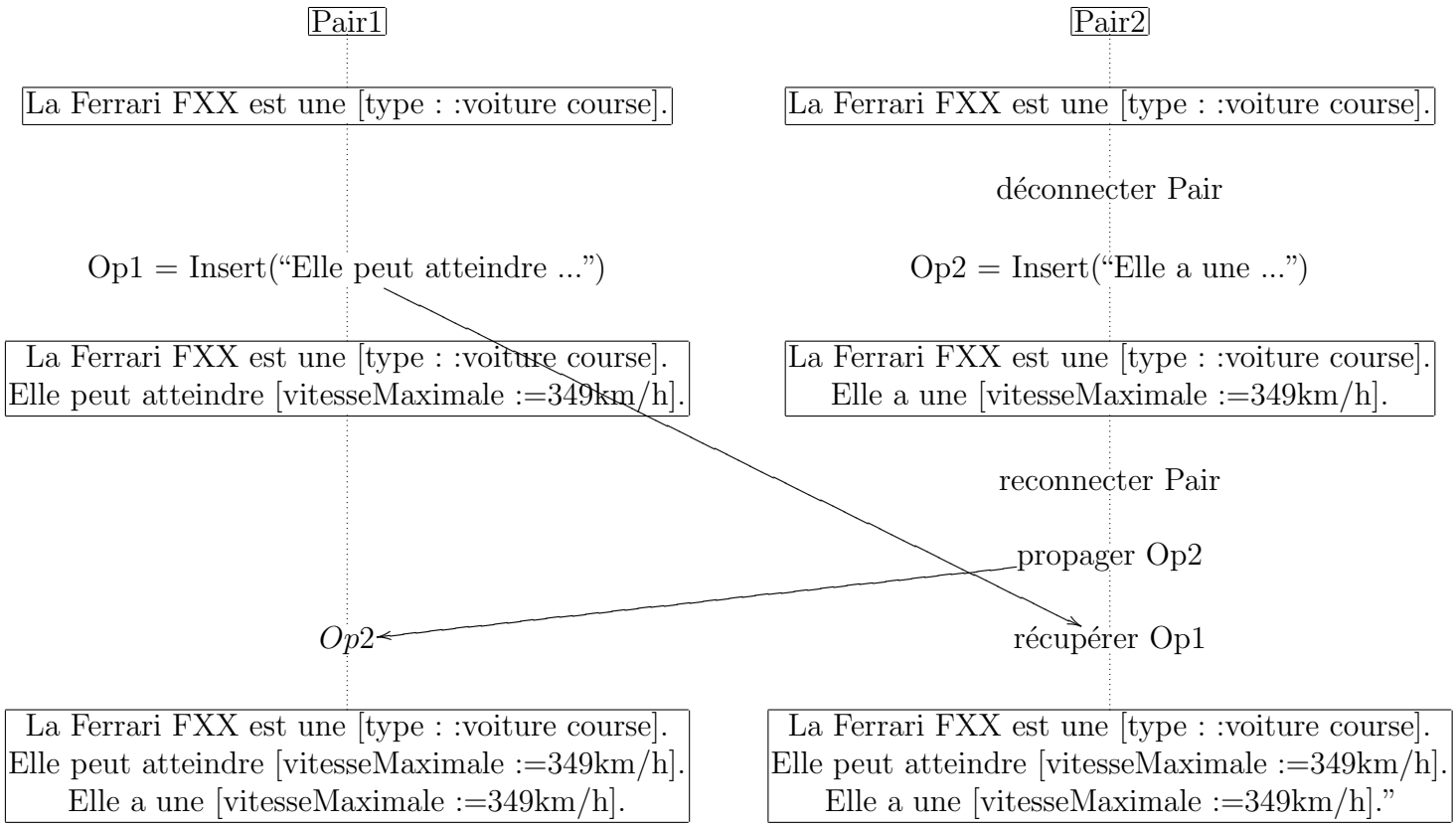

FIGURE 2.5 – Edition Déconnectée

les données répliquées. Le mécanisme d'annulation développé pour Swooki est présenté à la section 2.9. Les parties colorées dans la figure 2.6 concernent des éléments conçus ou étendus de Swooki nécessaires pour supporter le mécanisme d'annulation.

Nous avons implémenté SWooki dans des servlets Java sur un serveur Tomcat. Ce prototype est disponible avec une licence GPL sur sourceforge à l'adresse `http://sourceforge.net/projects/wooki`. La possibilité de tester SWooki est également disponible en visitant un pair de SWooki sur `http://wooki.loria.fr/wooki1`.

Swooki instancie le modèle de réplication optimiste sur les pages wikis sémantiques et leurs annotations sémantiques. La propagation des données à tous les pairs est assurée au moyen de deux algorithmes Lpbcast [EGH+03] et une anti-entropie [DGH+88]. L'intégration des données respecte le modèle de cohérence CCI, elle est produite par Wooto [WUM07] un algorithme de réplication optimiste. Dans la suite de ce chapitre, nous détaillons cette instanciation et le mécanisme d'annulation de groupe de Swooki.

2.3 Modèle de données de Swooki

A chaque pair de Swooki est attribué un identifiant unique global $IDnoeud$ dans le réseau. Ces identifiants sont totalement ordonnés. Les pages wikis sémantiques elles aussi

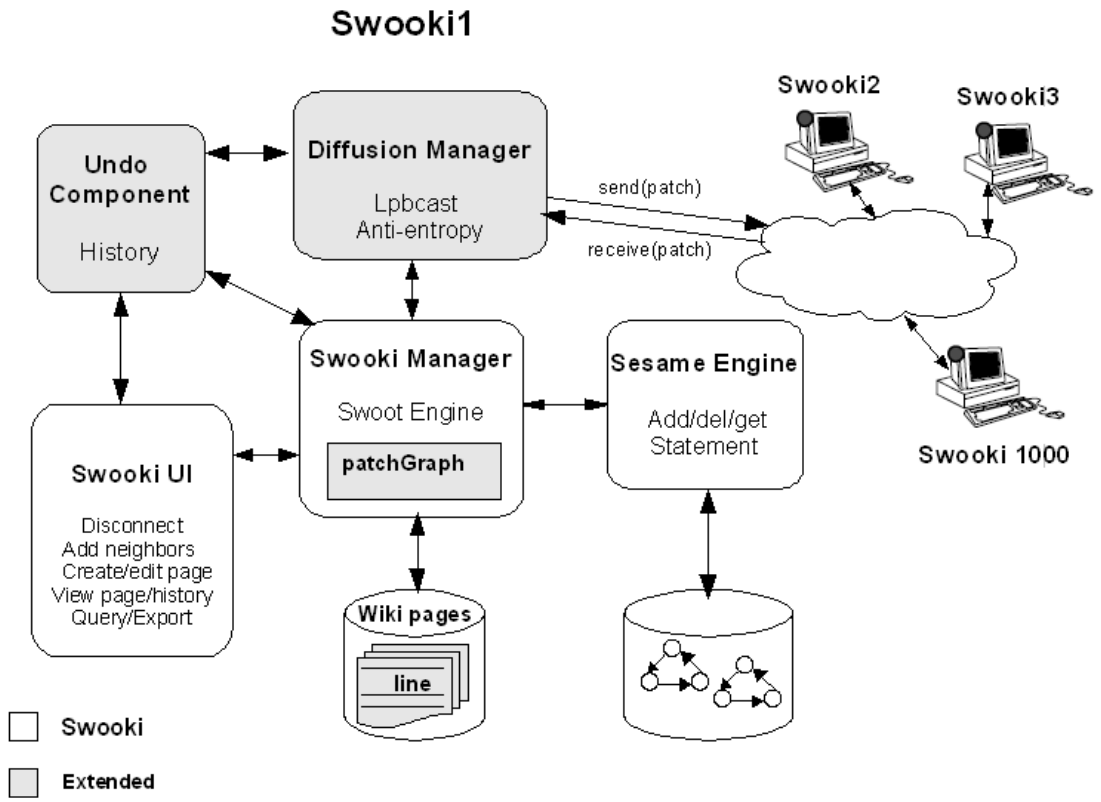

FIGURE 2.6 – L'architecture de Swooki

sont attribuées un identifiant unique $IDPage$ qui est le nom de la page. Ce nom est fixé lors de la création de la page. Si plusieurs pairs créent concurremment des pages ayant le même nom, le contenu de ces pages sera directement fusionné par l'algorithme de synchronisation. Un Identifiant de Ressource Uniforme URI peut être utilisé afin d'identifier sans ambiguïté le concept décrit par une page. Cet URI doit être global et indépendant de son emplacement afin d'assurer la distribution de la charge sur les pairs. Dans les deux sections suivantes nous définissons les modèles de stockage des pages wikis sémantiques et de leurs annotations respectivement.

2.3.1 Modèle de stockage des pages

Définition 7. *Une page wiki sémantique Page est une séquence de lignes wikis sémantiques $L_B, L_1, L_2, L_3...L_n, L_E$ où L_B et L_E sont des lignes spéciales indiquant le début et la fin de la page. Ce modèle est une extension du modèle Wooto [WUM07].*

Définition 8. *Une ligne wiki sémantique L est un quadruplet < IDLigne, contenu, degré, visibilité > où*

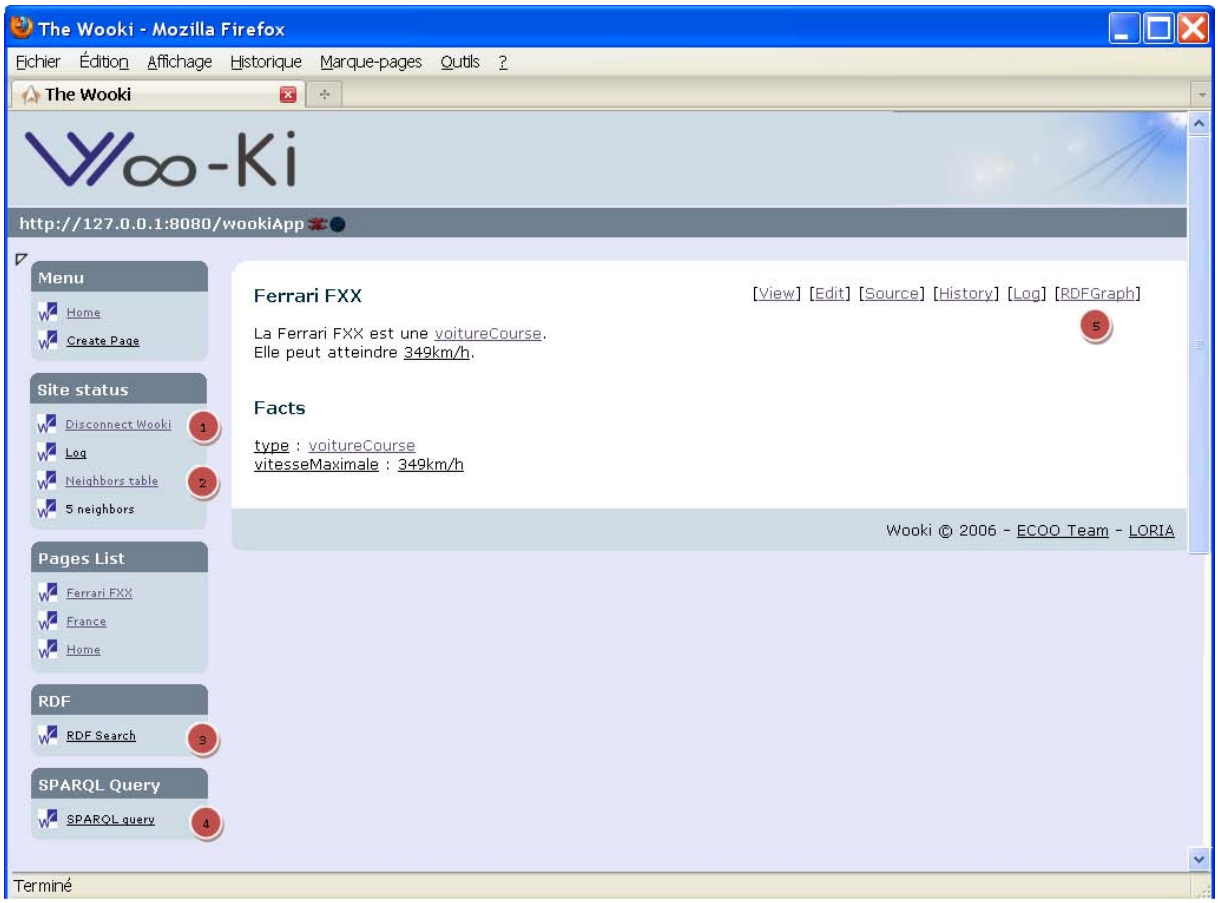

FIGURE 2.7 – L'interface utilisateur de SWooki

- *IDLigne est l'identifiant de la ligne, c'est un pair de (IDNoeud, horlogeLogique) où IDNoeud est l'identifiant du serveur wiki sémantique et horlogeLogique est une horloge logique associée à ce pair. Chaque pair maintient une horloge logique qui est incrémentée de un à chaque fois qu'une opération est produite sur ce pair. Les identifiants des lignes sont totalement ordonnés. Considérons deux lignes différentes* $IDligne_1$ *et* $IDligne_2$ *ayant les valeurs* $(IDNoeud_1, IDligne_1)$ *et* $(IDNoeud_2, IDligne_2)$ *respectives.* $IDligne_1 < IDligne_2$ *si et seulement si (1)* $IDNoeud_1 < IDNoeud_2$ *ou (2)* $IDNoeud_1 = IDNoeud_2$ *et* $IDligne_1 < IDligne_2$.
- *contenu est une chaîne de caractères représentant le contenu d'une ligne wiki sémantique. Il est constitué du texte et des annotations sémantiques intégrées dans la ligne.*
- *Le degré est un entier utilisé par l'algorithme de synchronisation. Le degré d'une ligne est fixé lors de la génération de la ligne, il représente une relation hiérarchique entre les lignes. Les lignes ayant un degré inférieur ont été générées avant celles ayant un degré supérieur. Par définition le degré de* L_B *et* L_E *est égal à zéro.*
- *La visibilité est un booléen qui spécifie si la ligne est visible ou non. Les lignes ne sont jamais supprimées afin d'assurer la convergence des pages wikis sémantiques, elles*

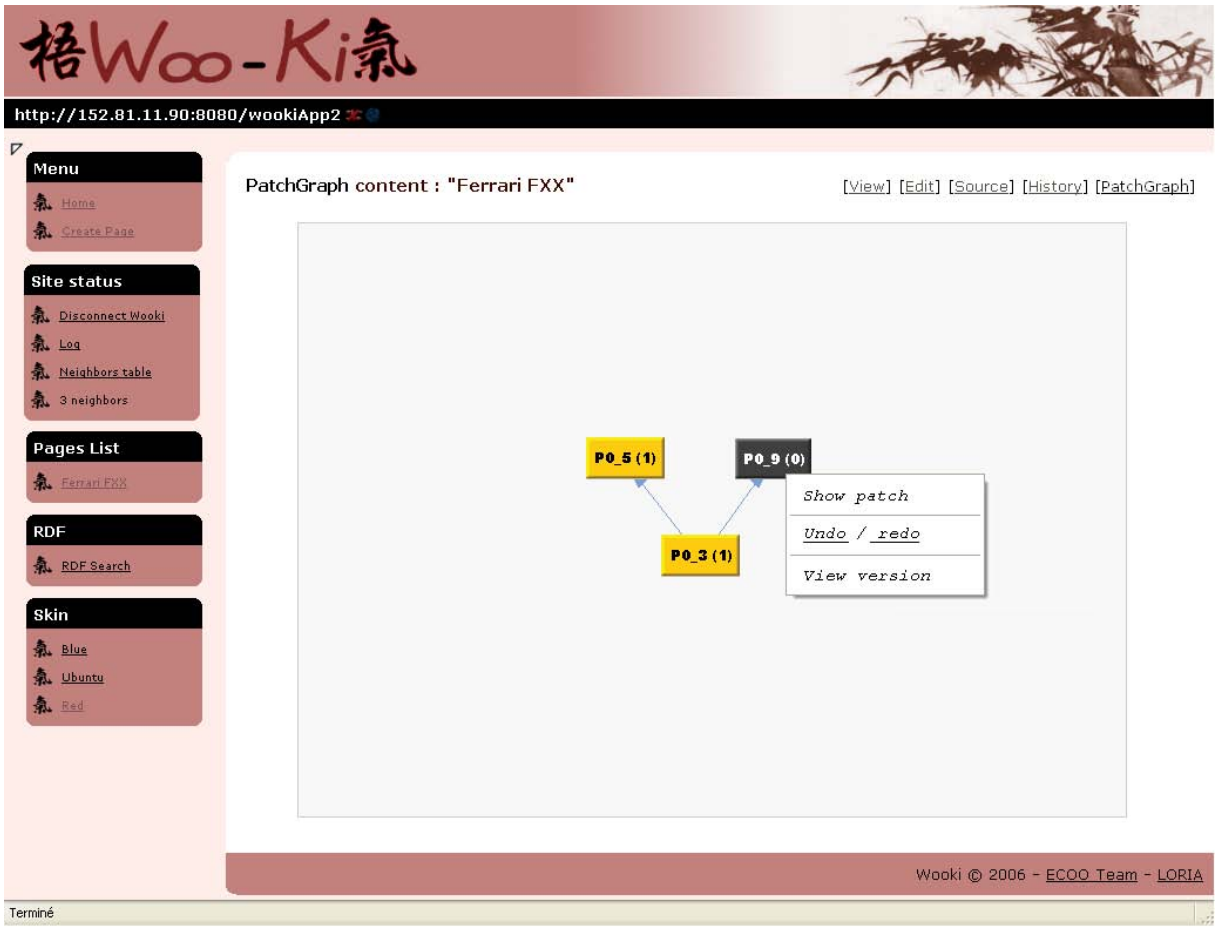

FIGURE 2.8 – L'interface d'annulation dans Swooki

sont marquées comme invisibles. Par contre, ces lignes seront toujours maintenues dans le modèle de la page ; elles sont tout simplement invisibles dans la vue de la page par l'utilisateur.

Par exemple, nous considérons une page wiki sémantique contenant deux lignes concernant la "France" ; "France" est l'identifiant de cette page.

France se situe en [située en::Europe]
La capitale de la France est [capitale::Paris]

Le texte de cette page est "France se situe en [située en : :Europe] La capitale de la France est [capitale : :Paris]" et ses annotations sémantiques sont "[située en : :Europe]" et "[capitale : :Paris]". Nous manipulons les annotations sémantiques intégrées dans le texte par les opérations d'édition de texte sans le besoin de définir de nouvelles opérations spécifiques pour l'édition des annotations sémantiques. Supposons que les deux lignes de la page "France" sont générées dans le même ordre sur le serveur wiki sémantique identifié par $IDNoeud = 1$ sans avoir des lignes invisibles. Par conséquent, la page wiki sémantique sera stockée comme suit :

L_B
((1,1), France se situe en [située en :: Europe], 1, true)
((1,2), La capitale de la France est [capitale :: Paris], 2, true)
L_E

Le texte et les données sémantiques sont stockés dans des espaces de stockage persistants séparés. Le texte peut être stocké dans des fichiers ou dans une base de données et les données sémantiques peuvent être stockées dans des entrepôts spécifiques.

2.3.2 Modèle de stockage des données sémantiques

Un wiki sémantique pair-à-pair doit posséder un espace de stockage pour les annotations sémantiques sur chaque pair. Cet espace est conçu pour contenir les annotations extraites des pages wikis sémantiques. Dans Swooki, chaque pair est associé à un entrepôt RDF local contenant les triplets RDF extraits des pages wikis sémantiques. Par exemple, l'entrepôt RDF du serveur de l'exemple précédent contient les triplets suivants : R = {<France, située en, Europe>, <France, capitale, Paris>}. Similairement à l'identifiant d'une page, un URI global peut être attribué à chaque prédicat et objet. Afin de manipuler l'entrepôt RDF, nous définissons deux opérations :

- insèreRDF(R, t) : insère un triplet t dans l'entrepôt RDF local R.
- supprimeRDF(R, t) : supprime un triplet t dans l'entrepôt RDF local R.

Ces opérations ne sont pas manipulées directement par les utilisateurs, elles sont appelées implicitement par les opérations d'édition lors des changements des lignes contenant des annotations sémantiques.

2.4 Opérations d'édition dans Swooki

Nous considérons deux types d'opérations d'édition, celles qui sont définies pour manipuler le modèle de stockage du wiki et d'autres qui sont utilisées par les utilisateurs.

2.4.1 Opérations d'édition du modèle de stockage

Nous utilisons deux opérations pour l'édition du texte wiki : *insère* et *supprime*. Une mise à jour est considérée comme une suppression de l'ancienne valeur suivie par une insertion d'une nouvelle valeur. La granularité d'un changement est une ligne. Il n'y a pas d'opérations spéciales pour l'édition des données sémantiques. Puisqu'elles sont intégrées dans le texte, elles sont éditées par les opérations du texte. De ce fait, les entrepôts RDF sont répliqués et synchronisés comme un effet de bord de la réplication et de la synchronisation du texte.

- *Insère(IDPage, ligne, lp, ln)* où IDPage est l'identifiant de la page pour la ligne insérée, ligne = < IDligne, contenu, degré, visibilité > est la ligne à insérer. *lp* est

l'identifiant de la ligne qui précède la ligne insérée et *ln* est celui de la ligne suivante. Ces deux lignes présentent une précondition d'insertion de la ligne à insérer, elles représentent l'intention de l'utilisateur comme c'est défini dans [WUM07]. Durant l'opération d'insertion, les données sémantiques intégrées dans la ligne sont extraites, les triplets RDF sont ainsi construits ayant le nom de la page comme sujet. Ensuite, ces triplets sont insérés dans l'entrepôt RDF local au moyen de la fonction *insèreRDF(R, t)*.

– L'opération $Supprime(IDPage, IDLigne)$ change la visibilité de la ligne identifiée par IDLigne de la page IDPage qui devient fausse. La ligne n'est pas supprimée physiquement mais seulement marquée comme supprimée. Les identifiants de ces lignes doivent être gardés dans le modèle de la page, ils sont nommés pierres tombales. Les pierres tombales sont largement utilisées dans la réplication optimiste et tout spécialement dans Usenet [SL98]. Elles sont utilisées pour laisser une trace de tous les changements. Semantic Media Wiki [KVV06] adopte un choix similaire en gardant toutes les versions de toutes les pages wikis. Durant une opération de suppression, les annotations contenues dans la ligne supprimée sont transformées en des triplets RDF et sont supprimées de l'entrepôt RDF local au moyen de la fonction $supprimeRDF(R, t)$.

Prenons par exemple un pair identifié par IDnoeud = 1 ayant une horloge logique égale à zéro qui génère l'opération : op = insère("France", < (1,1), "France se situe en [située en : :Europe]", 1, true, L_B, L_E). Durant l'insertion de cette opération, l'annotation sémantique [située en : :Europe] est extraite et insérée dans l'entrepôt RDF local du $pair1$. En d'autres termes, durant l'opération d'insertion l'entrepôt RDF est augmenté par les annotations sémantiques insérées. De même, si le $pair1$ génère l'opération de suppression op = supprime("France", (1,1)), alors la visibilité de la ligne (1,1) devient fausse et le triplet (France, située en, Europe) est supprimé de l'entrepôt RDF local.

2.4.2 Opérations d'édition des utilisateurs

Les utilisateurs d'un wiki sémantique pair-à-pair n'éditent pas directement le modèle des données. Afin d'éditer une page wiki sémantique, l'utilisateur ouvre l'interface d'édition de cette page alors une vue du modèle est affichée. Dans cette vue, seulement les lignes visibles sont affichées c'est-à-dire celles qui ne sont pas supprimées. Comme dans l'édition d'un wiki sémantique, l'utilisateur génère des modifications en ajoutant des nouvelles lignes ou supprimant des lignes existantes et sauvegarde la page. La détection des opérations de l'utilisateur est faite par un algorithme de $diff$ qui calcule la différence entre la page initialement demandée et la page sauvegardée. Ensuite ces opérations sont transformées en opérations d'édition du modèle. Une suppression de la ligne numéro n est transformée en une suppression de la $n^{\text{ème}}$ ligne visible. De même, une insertion à la position n est transformée en une insertion entre les lignes visibles $(n-1)^{\text{ème}}$ et $n^{\text{ème}}$. Ces opérations sont intégrées localement, ensuite propagées aux autres pairs afin d'être inté-

grées. En résumé, une opération d'insertion ou de suppression d'un utilisateur est effectuée en quatre étapes :

1. Transformer l'opération de l'utilisateur en une opération sur le modèle de stockage.

2. Exécuter cette opération d'édition localement. Durant son intégration, les annotations sémantiques sont extraites de la ligne. Ces annotations sont transformées en des triplets et stockées dans l'entrepôt RDF. Une mise à jour de l'entrepôt RDF local est ainsi effectuée.

3. Propager l'opération de l'édition transformée à travers le réseau pair-à-pair.

4. Recevoir l'opération par les autres pairs et finalement l'intégrer dans leur copie locale.

2.5 Propagation des opérations

Un modèle de réplication optimiste fait l'hypothèse que toutes les opérations émises arrivent éventuellement sur tous les pairs connectés ou non. Pour réaliser cela, le gestionnaire de diffusion de Swooki combine un algorithme de propagation probabiliste avec un algorithme anti-entropie [DGH+88]. L'algorithme de propagation diffuse rapidement les mises à jour à travers le réseau, il permet de maintenir l'adhésion des pairs dans Swooki. L'algorithme anti-entropie est responsable de récupérer les mises à jour manquantes pour les pairs qui ont été déconnectés ou en panne. Afin d'être déployé sur un réseau pair-à-pair (P2P), le protocole de diffusion doit passer à l'échelle, être fiable et soutenir la dynamicité du réseau. En effet, dans un réseau P2P, les pairs joignent et quittent le réseau à tout moment. Pour cela, nous utilisons un algorithme de propagation probabiliste leger Lpbcast [EGH+03] qui donne des garanties probabilistes que les opérations envoyées seront livrées à tous les pairs connectés. Dans Swooki, chaque pair gère une liste des voisins de taille fixe qui contient une sous partie des pairs du réseau P2P. Lpbcast met à jour cette liste durant la propagation des opérations. Lpbcast donne des fortes garanties probabilistes de ne pas avoir des partitions dans le réseau P2P, il assure une diffusion fiable et qui passe à l'échelle des opérations sur les pairs connectés. Pour gérer l'édition déconnectée, nous combinons l'algorithme Lpbcast avec un algorithme d'anti-entropie. Nous utilisons l'algorithme anti-entropie [DGH+88]. Lorsqu'un pair commence une anti-entropie, il choisit un voisin au hasard dans sa liste locale de voisins et lui envoie un résumé de ses propres opérations reçues. Le pair sélectionné retourne les opérations manquantes au pair appelant. L'utilisation d'une anti-entropie implique que chaque pair conserve les opérations reçues dans un journal.

2.6 Modèle de cohérence de Swooki

Dans cette section nous présentons le modèle de cohérence assuré par notre approche Swooki. Il est basé sur le modèle de cohérence CCI défini dans [OUMI06] mais appliqué dans le contexte du wiki sémantique pair-à-pair répliquant du texte wiki et des annotations sémantiques.

Notre mécanisme de réplication optimiste garantit le modèle de cohérence CCI défini par les trois critères suivants :

Respect de la Causalité Les opérations reçues sont intégrées si leurs pré-conditions sont satisfaites. Les pré-conditions d'une opération expriment sous quelles conditions cette opération peut s'exécuter.

Convergence des Copies Lorsque tous les sites ont exécuté les mêmes opérations, les répliques sont identiques.

Respect de l'Intention Pour toute opération op, les effets de l'exécution de op sur tous les sites sont les mêmes que les effets de l'exécution de l'opération op sur son état de génération.

2.6.1 Respect de la Causalité

La propriété de la causalité assure que les opérations ordonnées par une relation de précédence vont être exécutées dans le même ordre sur chaque pair. Nous adoptons la définition de la relation de précédence comme définie dans [LLSG92]. La relation de précédence est basée sur une dépendance sémantique causale. Cette dépendance est explicitement exprimée comme des préconditions sur les opérations. Pour cela, les opérations sont exécutées sur un état dans lequel les préconditions sont satisfaites. Nous présentons la causalité pour des opérations d'édition qui manipulent du texte avec des données RDF comme suit :

Préconditions de l'insertion soit la page $Page$ identifiée par $IDPage$ et l'opération *op= Insère(IDPage, nouvelleLigne, p , n)*, où la ligne *nouvelleLigne* =< IDLigne, c, d, v> est générée sur un nœud $IDnoeud$ et R est son entrepôt RDF associé. La ligne *nouvelleLigne* peut être insérée dans la page $Page$ si la ligne qui la précède et celle qui la suit sont présentes dans le modèle de données de la page $Page$.

Préconditions de la suppression soit la page $Page$ identifiée par $IDPage$ et l'opération *op = Supprime(IDPage, dl)* générée sur un nœud $IDnoeud$ et R est son entrepôt RDF associé. La ligne identifiée par *dl* peut être supprimée (marquée comme invisible) si cette ligne existe déjà dans le modèle de la page $Page$.

Par exemple, il existe une dépendance causale entre l'opération op_1=*Insère("France", < (1,1), "France se situe en [située en : :Europe]", 1, vraie>, L_B, L_E)* et l'opération op_2=*Insère("France", < (1,2), "La capitale de la France est [capitale : :Paris]", 1, vraie>,*

(1,1), L_E). Cette dépendance peut être déduite des préconditions de l'opération op_2. La précondition de op_2 nécessite l'existence de la ligne (1,1), par conséquent la relation $op_1 \prec op_2$ (op_1 précède op_2) doit être respectée sur chaque pair. Lorsqu'un pair reçoit une opération, cette opération est intégrée immédiatement si ses préconditions sont correctes sinon elle est ajoutée dans une queue d'attente. Elle sera intégrée lorsque ses préconditions seront satisfaites.

2.6.2 Convergence des données répliquées

Le wiki sémantique pair-à-pair doit assurer que lorsque les mêmes opérations ont été exécutées sur toutes les répliques :

- toutes les répliques des pages wikis sémantiques sont identiques,
- toutes les répliques des entrepôts RDF sont identiques.

De nombreux algorithmes ont été proposés dans le domaine de la réplication optimiste des structures linéaires. Peu d'entre eux permettent aux pairs de joindre et de quitter le réseau à tout moment. Ces algorithmes assurent souvent la cohérence éventuelle seulement sans la préservation de l'intention. Dans ce cas, le système peut lui même générer des pertes de mise à jour ce qui est inacceptable dans le cas de l'édition collaborative. L'algorithme de réplication WOOTO [OUMI06] présenté dans la section 1.7.1 supporte l'édition collaborative massive et respecte le modèle de cohérence CCI pour des structures linéaires. WOOTO a été conçu pour synchroniser des données textuelles mais il n'est pas adapté pour synchroniser des graphes RDF. Nous utilisons l'algorithme de réplication optimiste WOOTO [OUMI06, WUM07] pour l'intégration des opérations dans Swooki. Cet algorithme garantit la convergence des pages wikis sémantiques puisque les annotations sémantiques sont traitées comme du texte dans le contenu de ces pages.

Une fois la convergence des pages wikis sémantiques est assurée, le problème qui reste est d'assurer la convergence des entrepôts RDF des différents pairs. Un entrepôt RDF dans les wikis sémantiques actuels est défini comme étant un ensemble de triplets RDF c'est-à-dire il ne peut avoir qu'une seule occurrence d'un même triplet dans l'entrepôt RDF. La définition actuelle des entrepôts RDF ne permet pas leur convergence, nous présentons ce problème au moyen de l'exemple suivant :

Considérons trois nœuds wikis sémantiques pair-à-pair $Pair_1$, $Pair_2$ et $Pair_3$ qui partagent une page wiki sémantique à propos de la "France". Ce scénario est illustré à la figure 2.9. Chaque pair possède sa propre copie de la page "France" et son entrepôt RDF. Initialement, la page "France" et l'entrepôt RDF sont vides sur tous les pairs. Sur le pair $Pair_1$, l'utilisateur U_1 insère la ligne "France se situe [située en : :Europe]" à la première position dans sa copie locale de la page "France". D'une façon concurrente, l'utilisateur U_2 insère la ligne "France est un pays situé en [située en : :Europe]" dans sa propre copie de la page "France" à la même position sur le pair $Pair_2$ et finalement, sur le pair $Pair_3$ l'utilisateur U_2 supprime la ligne insérée par U_1 comme c'est illustré à la figure 2.9.

Sur le pair $Pair_1$, l'opération op_1 est intégrée localement, la page wiki sémantique contient la ligne : "France se situe [située en : :Europe]" et l'annotation sémantique "[située en : :Europe]" est ajoutée dans l'entrepôt RDF local sous la forme d'un triplet RDF. Ensuite, op_2 est reçue et intégrée par l'algorithme WOOTO [OUMI06], le résultat est une intégration du contenu de l'opération op_2 à la deuxième ligne. op_1 est intégrée avant op_2, l'algorithme WOOTO utilise la relation d'ordre sur les identifiants des sites pour sérialiser les opérations concurrentes. La propriété "[située en : :Europe]" est maintenant présente deux fois dans la page wiki et une seule fois dans l'entrepôt RDF en respectant la définition de l'entrepôt RDF. Finalement, op_3 est reçue et intégrée, la ligne insérée par l'opération op_1 est supprimée ainsi que son annotation sémantique. Par conséquent, l'entrepôt RDF du pair $Pair_1$ devient vide. Un résultat similaire est obtenu sur le pair $Pair_2$.

Sur le pair $Pair_3$, op_1 est reçue et intégrée. l'utilisateur U_3 décide de supprimer la ligne insérée par op_1. op_3 est localement intégrée, donc la propriété "[située en : :Europe]" est supprimée de l'entrepôt RDF du pair $Pair_3$. Ensuite l'opération op_3 est diffusée vers les autres pairs pour être intégrée. Finalement, op_2 est reçue et intégrée. Après l'exécution des trois opérations sur le pair $Pair_3$, la propriété "[située en : :Europe]" est présente dans la page wiki sémantique et dans l'entrepôt RDF. Toutefois, cette propriété est présente dans la page wiki sémantique des pairs $Pair_1$ et $Pair_2$ mais absente de leur entrepôt RDF. Comme nous pouvons remarquer qu'à l'état final le contenu de la page wiki est le même sur tous les pairs, par contre leurs entrepôts RDF divergent c'est-à-dire ils ne possèdent pas le même ensemble de triplets RDF.

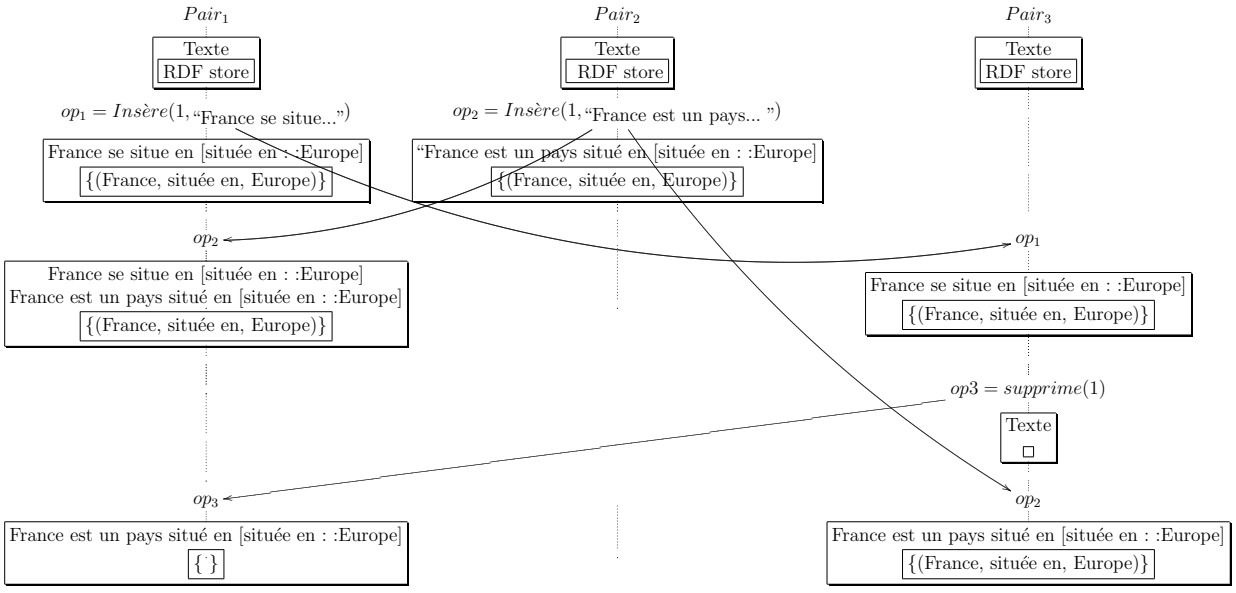

FIGURE 2.9 – Divergence après intégration des modifications concurrentes

La solution que nous proposons à ce problème est de définir l'entrepôt RDF comme un multi-ensemble de triplets afin d'assurer leur convergence.

Définition 9. *Un entrepôt RDF est un multi-ensemble de triplets. Il est défini comme*

un ensemble de pairs (T, m) *où* T *est un ensemble de triplets RDF et* m *est la fonction de multiplicité* $m : T \rightarrow \mathcal{N}$ *où* $\mathcal{N} = 1, 2, ...$

Par exemple, le multi-ensemble R = { ("France", "située en", "Europe"), ("France", "située en", "Europe"), ("France", "capitale", "Paris") } peut être présenté par R = { ("France", "située en", "Europe")^2, ("France", "capitale", "Paris")^1 } où 2 est le nombre d'occurrences du premier triplet et 1 est celui du deuxième.

Nous reprenons le même scénario mais cette fois ci nous l'exécutons avec notre solution. Sur le pair $Pair_1$, l'opération op_1 est intégrée localement, son entrepôt RDF contient une seule occurrence de l'annotation sémantique "[située en : :Europe]". Après la réception et l'intégration de op_2, la propriété "[située en : :Europe]" est maintenant présente deux fois dans la page wiki, donc sa multiplicité est égale à deux. Finalement, l'opération op_3 est reçue et intégrée, le résultat est la suppression de la ligne insérée par l'opération op_1 et la multiplicité de la propriété est décrémentée de un. Un résultat similaire est obtenu sur le pair $Pair_2$.

Après l'intégration de l'opération op_1 sur le pair $Pair_3$, la page wiki sémantique contient "France se situe [située en : :Europe]" et l'annotation sémantique "[située en : :Europe]" est ajoutée dans l'entrepôt RDF ayant une multiplicité de un. L'opération op_3 supprime cette ligne de la page wiki sémantique, ce qui décrémente le nombre d'occurrences de l'annotation sémantique. Son occurrence devient zéro, donc le triplet est supprimé de l'entrepôt RDF. Finalement, l'opération op_2 est reçue et intégrée et le résultat est l'insertion du contenu de la ligne dans la page wiki et l'annotation sémantique "[située en : :Europe]" est ajoutée dans l'entrepôt RDF avec une multiplicité de un.

En comparant le résultat final de l'exécution des trois opérations sur les trois pairs, nous constatons que les pages wikis sémantiques et les entrepôts RDF de ces pairs convergent vers le même état. Cela est illustré à la figure 2.11. Par conséquent, la propriété de la convergence des données répliquées est respectée.

Une autre solution pour assurer la convergence des entrepôts RDF est de garder l'ancienne définition des entrepôts RDF intacte, c'est-à-dire l'entrepôt RDF est défini comme un ensemble de triplets mais de traiter les opérations comme suit. Initialement, un test peut être utilisé pour réduire l'accès à l'entrepôt RDF. Il consiste à tester lors de l'intégration d'une opération, si elle contient des annotations sémantiques. Si c'est le cas, il faut parser le contenu de la page wiki sémantique obtenu après l'intégration de cette opération, supprimer toutes les annotations sémantiques correspondantes à cette page de l'entrepôt RDF et ajouter les annotations du contenu de la page dans l'entrepôt RDF. Dans cette solution, la granularité du changement des annotations sémantiques dans l'entrepôt RDF est égale au nombre d'annotations dans la page wiki sémantique et non pas aux annotations inclues dans les opérations. Une solution plus coûteuse que celle adoptée surtout lorsque le nombre des annotations devient très grand dans la page wiki sémantique.

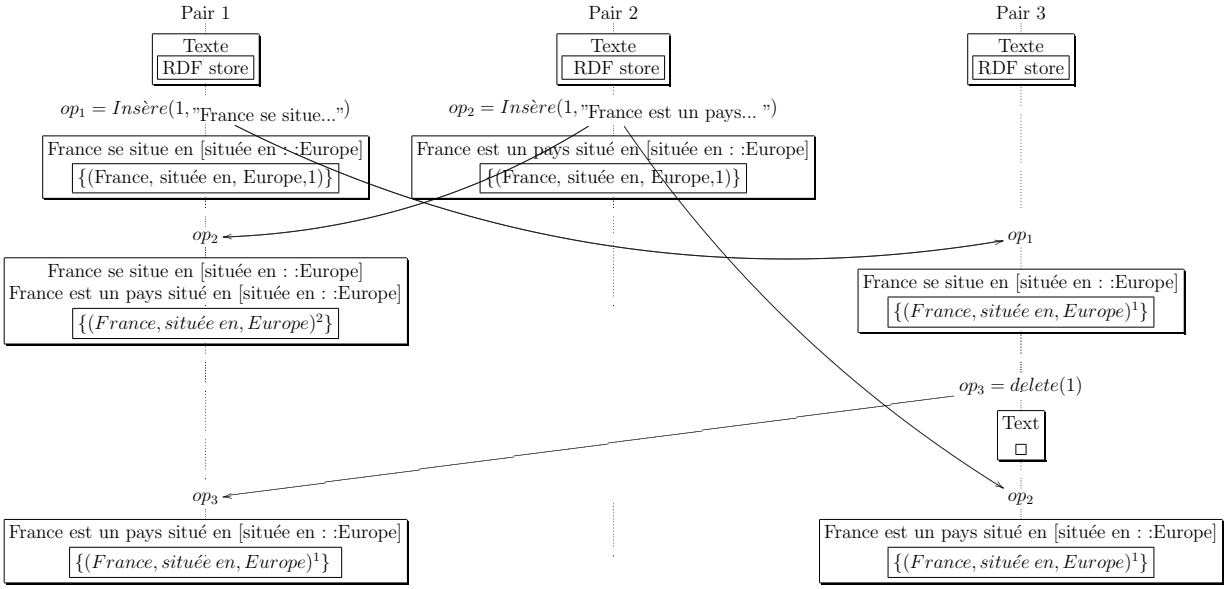

FIGURE 2.10 – Convergence après intégration des modifications concurrentes

2.6.3 Respect de l'Intention

L'intention d'une opération est l'effet visible observé lors d'un changement généré sur un pair, la préservation de l'intention signifie que l'opération va être observable sur tous les pairs malgré les opérations concurrentes. L'intention d'une opération dépend du type des données manipulées, c'est une sorte de postcondition de l'opération. La définition de l'intention n'est pas une tâche facile, selon cette définition il est possible de préserver ou de violer l'intention. L'intention des opérations d'insertion et de suppression de texte peut être définie comme une relation de précédence sur les lignes à leur génération comme c'est le cas dans [OUMI06] et cet ordre doit être préservé lors de l'intégration. Par exemple, si deux sites insèrent d'une façon concurrente deux lignes 'x' et 'y' entre les mêmes lignes 'a' et 'b', deux résultats peuvent être obtenus soit "axyb" ou bien "ayxb". Le résultat choisi dépend des identifiants des lignes qui sont comparables comme c'est défini à la section 2.3.1, soit "axyb" par exemple qui doit être le même sur tous les sites.

L'intention des opérations sur des données textuelles et sémantiques n'a jamais été définie. D'abord, nous pouvons définir l'intention des opérations d'insertion et de suppression d'une façon simple. Supposons par exemple que l'intention de l'opération d'insertion est d'insérer le contenu de la ligne entre la ligne qui la précède et celle qui la suit comme spécifié dans l'opération d'insertion et que les annotations sémantiques dans la ligne sont ajoutées dans l'entrepôt RDF du pair. Supposons aussi que l'intention de l'opération de suppression est de rendre le contenu de la ligne invisible et que les annotations sémantiques dans le contenu de la ligne sont supprimées de l'entrepôt RDF du site.

Nous remarquons que cette définition n'est pas respectée dans le scénario présenté à la figure 2.9. Selon cette définition, l'intention de l'opération d'insertion op_2 produite par l'utilisateur U_2 sur le pair $Pair_2$ doit insérer la ligne dans la page et ses annotations

sémantiques dans l'entrepôt RDF, ce qui n'est pas le cas dans le résultat final sur les différents pairs. Cela est obtenu à cause de la concurrence et de la définition de l'entrepôt RDF. Pour surmonter l'incohérence, nous avons besoin de définir une sorte de "contraintes d'intégrité de références" entre les annotations sémantiques et l'entrepôt RDF. Une façon de le faire est de transformer l'entrepôt RDF en un multi-ensemble de triplets dans le but de refléter l'effet de l'insertion de chaque triplet.

D'après ce qui précède, nous définissons l'intention de la manière suivante :

Définition 10. ***L'intention de l'opération d'insertion*** *soit S un pair du wiki sémantique pair-à-pair, R son entrepôt RDF et la page $Page$ une page wiki sémantique sur ce pair. L'intention de l'opération d'insertion op=*Insère($IDPage$, $nouvelleLigne$, p , n) *lorsqu'elle a été générée sur le site S est définie comme suit : (1) le contenu de la ligne est insérée entre les deux lignes p et n et (2) les annotations sémantiques dans le contenu de la ligne sont ajoutées dans R.*

$$\exists i \quad \wedge \exists\, i_P < i \; LineID(Page[i_P]) = p \tag{2.1}$$
$$\wedge \exists\, i \leq i_N \; LineID(Page[i_N]) = n \tag{2.2}$$
$$\wedge\, Page'[i] = newline \tag{2.3}$$
$$\wedge\, \forall j < i \; Page'[j] = Page[j] \tag{2.4}$$
$$\wedge\, \forall j \geq i \; Page'[j] = Page[j-1] \tag{2.5}$$
$$\wedge\, R' \leftarrow R \uplus T \tag{2.6}$$

Où $Page'$ et R' sont les nouvelles valeurs de la page et de l'entrepôt RDF respectivement après l'intégration de l'opération d'insertion sur le site S et $\uplus$ est l'opérateur union de multi-ensembles. Si un triplet de T existe déjà dans R donc sa multiplicité est incrémentée sinon il est ajouté dans R avec un multiplicité de un.

Définition 11. ***L'intention de l'opération de suppression*** *soit S un pair du wiki sémantique pair-à-pair, R son entrepôt RDF et la page $Page$ une page wiki sémantique sur ce pair. L'intention de l'opération de suppression op =* Supprime($IDPage$, dl) *lorsqu'elle a été générée sur un site S est définie comme suit : (1) le contenu de la ligne de cette opération est rendu invisible et (2) le nombre d'occurrences des annotations sémantiques intégrée dans dl est décrémenté de un, si son occurrence est égal à zéro ce qui signifie que cette annotation n'est plus référencée dans la page alors cette annotation sémantique est physiquement supprimée de R.*

$$\exists i \qquad \wedge\, LineID(Page'[i]) = ld \tag{2.7}$$
$$\wedge\, visibility(Page'[i]) \leftarrow false \tag{2.8}$$
$$\wedge\, R' \leftarrow R - T \tag{2.9}$$

Où $Page'$ et R' sont les nouvelles valeurs de la page et de l'entrepôt RDF respectivement après l'intégration de l'opération de suppression sur le site S et $-$ est l'opérateur de

soustraction de multi-ensembles. Si un triplet de T existe déjà dans R donc sa multiplicité est décrémentée et supprimé de R si cette multiplicité est égale à zéro.

Considérons de nouveau le scénario précédent déroulé à la figure 2.9. Nous le déroulons maintenant à la figure 2.11. Lorsque op_2 est intégrée sur $Pair_1$, la multiplicité du triplet *("France","située en","Europe")* est incrémentée et prend la valeur 2. Lorsque op_3 est intégrée sur le pair $Pair_1$, la multiplicité du triplet correspondant est décrémentée et la cohérence entre le texte et l'entrepôt RDF est assurée. Nous pouvons observer maintenant que les pairs $Pair_1$ et $Pair_2$ convergent et que l'intention est préservée.

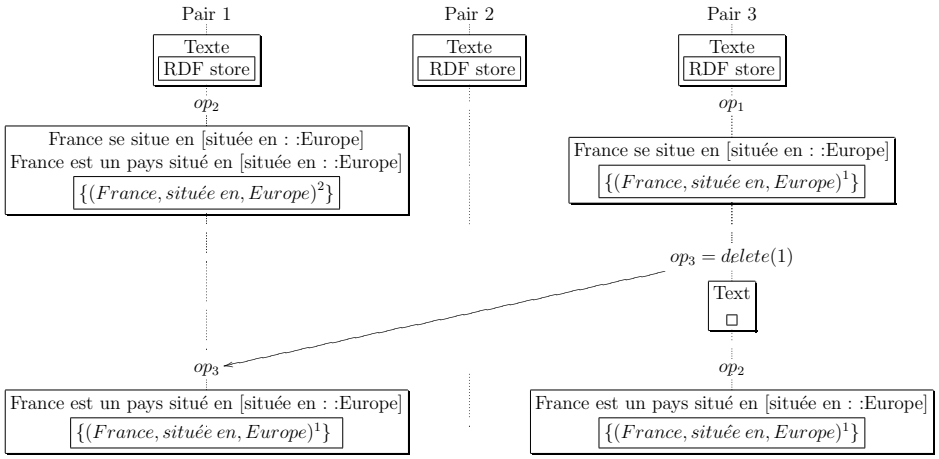

FIGURE 2.11 – Préservation de l'intention après intégration des modifications concurrentes

2.7 Réconciliation des données dans Swooki

La réplication des pages wikis imbriquant des annotations sémantiques dans un wiki sémantique pair-à-pair nécessite un algorithme de réplication optimiste pour synchroniser les modifications dans ces pages et de leurs annotations et pour assurer leur convergence sur tous les pairs. Cet algorithme doit supporter ce nouveau type de données : une page wiki sémantique et ses annotations. Dans Swooki, nous étendons l'algorithme WOOTO pour synchroniser et assurer le modèle de cohérence pour ce nouveau type de données. La mise à jour des triplets dans un entrepôt RDF est faite par la mise à jour du contenu des pages wikis sémantiques c'est-à-dire en utilisant seulement les opérations d'insertion et de suppression des lignes de texte.

Lorsqu'un utilisateur modifie une réplique d'une page wiki sémantique, l'opération correspondante est générée. Cette opération est (i) immédiatement intégrée en local, puis (ii) diffusée aux autres sites. Elle est ensuite (iii) reçue par les autres sites, pour y être enfin (iv) intégrée. L'opération doit être intégrée sur la copie locale afin que la ligne insérée soit positionnée par rapport aux lignes invisibles qui ont été supprimées. De même, elle sera intégrée sur les autres sites.

Durant la sauvegarde d'une page (voir figure 2.12) :
Lors de la sauvegarde d'une page wiki sémantique, un algorithme de Diff [Mye86] calcule la différence entre la version sauvegardée et la version précédente de la page et génère un patch. Un patch est un ensemble d'opérations d'insertion et de suppression de lignes sur la page (Op = Insère(IDPage, ligne, L_P, L_N) ou Op = Supprime(IDPage, IDLigne)). Le patch est intégré localement et propagé vers d'autres sites afin d'être intégré. Durant la génération d'une opération d'insertion, l'horloge logique du pair est incrémentée de un et les lignes précédente et suivante visibles sont extraites.

Dans Swooki, la diffusion des patchs est faite comme dans [WUM07]. Il utilise un algorithme de diffusion probabiliste léger Lpbcast [EGH+03] avec un algorithme d'anti-entropie [DGH+88]. Le premier diffuse rapidement les mises à jour sur le réseau P2P et gère l'adhésion des pairs. Le second récupère les mises à jour manquantes pour les pairs qui ont été déconnectés ou en panne.

```
1 Upon Save(page, oldPage)
2 begin
3     let P ← Diff(page, oldPage)
4   for each op ∈ P do
5     Receive(op)
6   endfor
7 Broadcast(P)
8 end
```

FIGURE 2.12 – La sauvegarde d'une page

Durant la réception d'une opération (voir figure 2.13) :
L'ordre de réception des opérations peut être différent de l'ordre de génération. Ainsi, une opération peut être reçue sur un site sans être exécutée directement si ses préconditions ne sont pas satisfaites. Lors de la réception d'une opération, elle est ajoutée à une queue d'attente. Ensuite pour chaque opération dans cette queue, l'algorithme teste si elle satisfait ses préconditions. Si c'est le cas, l'opération sera intégrée selon son type et elle sera retirée de la queue. Lors de l'intégration d'une opération, deux types d'intégration sont produites, une intégration textuelle et une autre pour les annotations sémantiques. Cet algorithme boucle tant qu'il y a des opérations exécutables dans la queue.

Les opérations d'insertion et de suppression des lignes sont intégrées comme dans WOOTO en respectant leurs préconditions (voir figure 2.14). Une opération de suppression est intégrée lorsque la ligne à supprimer existe dans la page et si cette ligne est visible. Nous ajoutons le test de visibilité de la ligne pour garantir que les annotations de cette ligne sont supprimées une seule fois de l'entrepôt RDF en cas de suppression concurrente de la même ligne sur différents sites. Par contre, l'opération d'insertion est intégrée lorsque les lignes entre lesquelles la ligne doit être insérée existent dans la page. La fonction containsL() teste l'existence d'une ligne dans une page et retourne une valeur booléenne.

```
1  uponReceive(op):
2    addOpWaitingQueue(op)
3    stop := false
4    while(stop = false) do
5        stop := true
6        for op in OpWaitingQueue do
7            if isExecutable(op)= true then
8                stop := false
9                if type(op) = insert
10                   IntegrateInsT(PageID, line, l_P ,l_N)
11                   IntegrateInsRDF(line)
12               else
13                   IntegrateDel(op)
14               endif
15               RemoveOpWaitingQueue(op)
16           endif
17       done
18   done
```

FIGURE 2.13 – La réception d'une opération

Durant l'intégration d'une insertion textuelle (voir figure 2.15) :
Pour intégrer une opération d'insertion op = insert(PageID, line, lP, lN), la ligne doit être placée parmi toutes les lignes qui sont entre lP et lN. Certaines de ces lignes ont pu être déjà insérées ou supprimées simultanément avec cette ligne et leurs annotations sémantiques ont été intégrées. Cette intégration d'une ligne dans une page wiki sémantique utilise l'algorithme d'intégration défini dans [WUM07]. Cet algorithme sélectionne d'abord une sous-séquence S' de lignes entre les lignes précédente et suivante de la ligne insérée. Dans le cas d'une séquence vide, la ligne est insérée avant la ligne suivante. Sinon, la sous séquence S' est filtrée en gardant seulement les lignes ayant le degré minimum de S'. Ces lignes sont ainsi ordonnées selon $<_{id}$, une relation d'ordre sur les identifiants des lignes définie dans [OUMI06]. La ligne va être intégrée à sa place selon $<_{id}$ parmi ces lignes filtrées. Cette procédure est appelée récursivement pour placer la ligne parmi les lignes ayant un degré supérieur dans S'.

Durant l'intégration des annotations sémantiques (voir figure 2.16) :
Nous assurons la définition de l'entrepôt RDF comme un multi-ensemble en associant un compteur à chaque triplet. La valeur du compteur correspond au nombre de fois le triplet figure dans l'entrepôt RDF. L'intégration des annotations sémantiques de la ligne dans l'entrepôt commence par leur extraction au moyen de la fonction ExtractRDF(). Cette fonction transforme ces annotations en des triplets RDF ayant comme sujet l'identifiant de la page, elle retourne un ensemble de triplets. Ensuite l'algorithme teste l'existence du

```
isExecutable(op)
 let pageid := pageID(op), l := ligne(op),
     lP := previousLine(op), lN := nextLine(op)
 if type(op) = del then
    return containsL(pageid, idl(l)) and isVisible(LineID)
 else
    return containsL(pageid, idl(lP))
        and containsL(pageid, idl(lN))
 endif
```

FIGURE 2.14 – La précondition d'une opération

```
 IntegrateInsT(PageID, line, l_P, l_N)
 begin
 let S' ← subseq(Page[PageID]), l_P, l_N)
 if S = ∅ then
   insert(PageID, line, l_N)
 else
       let i ← 0
       let d_min ← min(degree(S'))
       let F ← filter(S', degree = d_min)
       while (i < |F| − 1) and (F[i] <_id l) do
          i ← i +1
        IntegrateInsT(PageID, line, F[i−1], F[i])
end
```

FIGURE 2.15 – Intégration textuelle [WUM07]

triplet RDF dans le store avant de l'insérer, si c'est le cas il incrémente son compteur de un dans l'entrepôt RDF ; sinon il insère le triplet dans l'entrepôt RDF avec un compteur égal à un. La fonction Contains() teste la présence d'un triplet dans l'entrepôt RDF, elle retourne une valeur booléenne.

Durant l'intégration de la suppression (voir figure 2.17) :
L'intégration d'une opération de suppression rend la ligne invisible. La suppression des triplets RDF de la ligne dans l'entrepôt RDF est faite par la méthode DeleteRDF(). Cette méthode décrémente de un la valeur du compteur du triplet dans l'entrepôt. Les triplets ayant un compteur égal à zéro sont supprimés de l'entrepôt RDF.

```
1  IntegrateInsRDF(line)
2  begin
3   let   S ← ExtractRDF(line)
4   if  S ≠ ∅ then
5    for each  triple  ∈ S do
6      if   Contains(triple)  then
7           triple .counter++
8      else
9         insertRDF(R,triple)
10     endif
11   endif
12   end
```

FIGURE 2.16 – Intégration sémantique

```
1  IntegrateDel(LineID)
2  begin
3  Page[LineID]. visibility    ← false
4  let   S ← ExtractRDF(LineID)
5   if  S ≠ ∅ then
6    for each  triple  ∈ S do
7        triple .counter−−
8      if    triple .counter == 0 then
9        deleteRDF(R,triple)
10     endif
11   endif
12 end
```

FIGURE 2.17 – La suppression dans l'entrepôt RDF

2.8 Correction de l'approche

Dans cette section, nous montrons que notre algorithme de réplication garantit notre modèle de cohérence CCI sur des données textuelles et sémantiques.

Théorème 1. *Puisque notre algorithme est basé sur WOOTO, donc il assure les mêmes propriétés. L'algorithme d'intégration termine tout en respectant les préconditions. Il respecte le modèle de cohérence CCI pour les données textuelles.*

Théorème 2. *Notre algorithme d'intégration préserve l'intention.*

Démonstration. La préservation de l'intention pour le texte est démontrée dans [OUMI06]. Nous sommes concernés par la préservation de l'intention sur les annotations sémantiques comme définie à la section 2.6.3. L'intention d'une opération d'insertion est trivialement

préservée par l'algorithme IntegrateInsRDF puisqu'une façon d'implémenter un multi-ensemble est d'associer un compteur à chaque élément. De la même façon, l'algorithme IntegrateDelRDF préserve l'intention de l'opération de suppression.

□

Théorème 3. *Notre algorithme assure la convergence des entrepôts RDF.*

Démonstration. Un entrepôt RDF défini comme un multi-ensemble est un CRDT puisque les opérations concurrentes dans cet entrepôt commutent. Considérons tous les cas concurrents possibles :

- Insertion - Insertion : Deux insertions concurrentes de deux triplets identiques ou non dans l'entrepôt RDF évidemment commutent.
- Insertion - Suppression : Il est de même pour des opérations d'insertion et de suppression de triplets concurrentes. L'opération de suppression d'un triplet nécessite la suppression de la ligne contenant l'annotation sémantique qui correspond au triplet.
- Suppression - Suppression : La précondition d'intégration de la suppression d'une ligne empêche la suppression d'une ligne déjà supprimée. Donc les opérations de suppression de triplets commutent.

Par conséquent les entrepôts RDF sont un CRDT, donc selon [PMSL09] ils convergent nécessairement sur les différents pairs.

□

2.9 Mécanisme d'annulation dans Swooki

Similairement aux éditeurs collaboratifs classiques, Swooki a besoin de supporter le mécanisme d'annulation pour plusieurs raisons :

- L'annulation est une fonctionnalité nécessaire pour l'utilisateur. En effet, les utilisateurs peuvent utiliser cette fonctionnalité comme un moyen pour corriger leurs propres erreurs.
- Dans les éditeurs collaboratifs, quand deux ou plusieurs utilisateurs modifient en concurrence les mêmes données partagées, le système propose un document incluant toutes leurs modifications ou parfois les modifications de l'un sans celles de l'autre. Dans les éditeurs collaboratifs distribués, l'algorithme de réplication utilisé dans la fusion des changements sert à offrir un résultat le plus correct possible mais pas nécessairement celui attendu par les utilisateurs. Le mécanisme d'annulation dans ce cas est utile pour défaire des résultats inattendus ou non souhaités. Ce mécanisme d'annulation appelé *annulation de groupe* (group undo) supporte les annulations concurrentes.
- Nous avons conçu Swooki comme un wiki sémantique pair-à-pair ouvert où tout le monde peut joindre le wiki. Puisque tout le monde peut joindre, des utilisateurs malveillants peuvent également joindre ce système. Par conséquent, le mécanisme d'annulation peut être utilisé pour éliminer l'impact des actes de vandalisme.

Dans cette section, nous proposons un mécanisme d'annulation de groupe dans Swooki. Ce mécanisme respecte la définition de l'annulation et assure la cohérence CCI sur les données répliquées composées des pages wikis sémantiques et de leurs annotations. Nous procédons à des changements dans le modèle de données de Swooki. Nous étendons les opérations des utilisateurs par des actions *undo* et *redo* qui annulent et réappliquent respectivement les changements, ces actions sont propagées dans le réseau via le gestionnaire de diffusion de Swooki. Nous développons l'algorithme approprié responsable de la génération et de l'intégration des ces actions. L'extension de Swooki assure la convergence des pages wikis sémantiques et des entrepôts RDF sur tous les pairs. Cette convergence est indépendante des modifications concurrentes, de l'ordre de l'intégration des actions *undo* et *redo* et du fait que les utilisateurs peuvent éditer en mode déconnecté et joindre ou quitter à tout moment.

2.9.1 Réaliser l'annulation

Pour réaliser ce mécanisme, la fonction *revert* qui permet de retourner à une ancienne version semble adéquate : nous pouvons effacer tout le contenu et ajouter le contenu précédemment créé. Malheureusement, cette fonction ne permet pas d'annuler n'importe quelle modification. En effet, des opérations concurrentes peuvent être reçues dans des ordres différents. Prenons cet exemple, à partir d'une version initiale V_0 (voir la figure 2.18), un utilisateur U_1 sur le site $Site1$ insère une deuxième ligne, donc il génère une nouvelle version V_1 incluant la modification M_1. Un autre utilisateur U_2 sur le site $Site2$ insère simultanément une deuxième ligne, il génère une nouvelle version V_2 et la modification M_2. Malheureusement, U_2 a commis une erreur concernant la vitesse maximale de la voiture. L'intégration des deux modifications M_1 et M_2 sur les deux sites résulte en une version V_3 contenant les deux. Afin de corriger l'erreur, chaque site possède différentes alternatives. Le site $Site1$ qui a intégré premièrement M_1, peut restaurer les versions V_1 ou V_0. La version V_1 est le résultat attendu puisqu'elle contient toutes les modifications à l'exception de la modification incorrecte. Contrairement, le site $Site2$ qui a reçu premièrement M_2, peut retourner aux versions V_2 ou V_0. La version V_2 contient la modification incorrecte et la version V_0 ne contient pas M_1. Comme résultat, le site $Site2$ est incapable de supprimer seulement la modification incorrecte en utilisant la fonction *revert* sans une perte de la modification M_1 du site $Site1$.

Une autre idée est d'annuler les modifications en faisant une modification inverse. Malheureusement, cela ne respecte pas la définition de l'annulation qui dit "annuler une modification fait retourner le système à l'état qu'il aurait atteint si cette modification n'a jamais été produite". Nous illustrons ce cas à la figure 2.19.

A partir de la version V_3 résultat de l'exemple précédent, un utilisateur U_3 sur le site $Site2$ génère une modification de vandalisme M_4 en supprimant le contenu entier du document. M_4 supprime toutes les lignes dans le document. Simultanément, U_1 sur le site $Site1$ supprime la troisième ligne qui contient une erreur. Ensuite M_4 est reçue

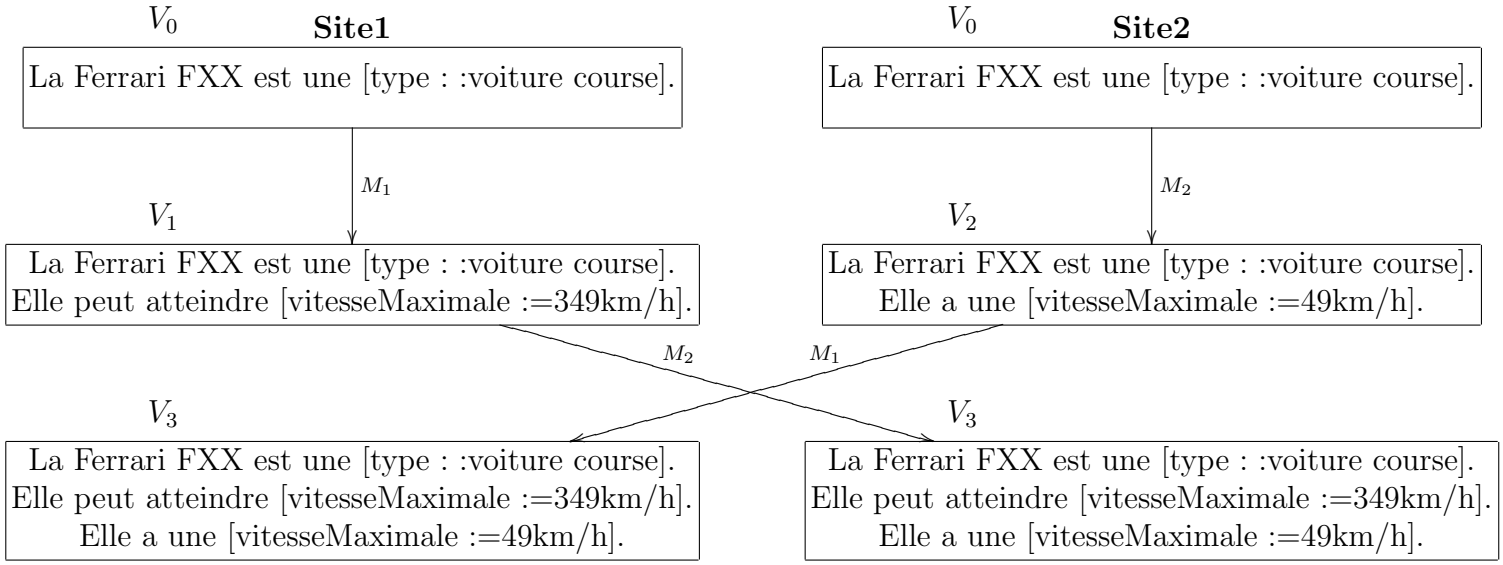

FIGURE 2.18 – Scénario d'édition concurrente

et intégrée sur le site $Site1$. La modification inverse de M_4 insère les trois lignes. Par conséquent, lorsque l'utilisateur U_1 défait M_4 sur le site $Site1$, il réintroduit les trois lignes et perd sa propre modification M_3 qui vise à supprimer la troisième ligne. Si M_4 n'a jamais été produite, le document sera corrigé pour "La Ferrari FXX est une [type : :voiture course]. Elle peut atteindre [vitesseMaximale :=349km/h].". Une annulation correcte doit retourner le document à cet état. Notre approche consiste à fournir une telle fonctionnalité d'annulation.

Quand un utilisateur souhaite annuler une modification dans Swooki c'est-à-dire un patch, le document et l'entrepôt RDF doivent retourner à un état dans lequel la modification n'a jamais été exécutée conformément à la définition de l'annulation. La définition implique deux cas :

- Le patch est déjà annulé et le document et l'entrepôt ne doivent pas être modifiés,
- le patch doit être désactivé et son effet doit être supprimé du document et de l'entrepôt RDF.

En outre, puisque l'action défaire d'un patch est aussi une modification du document, les utilisateurs doivent être en mesure d'annuler cette action, ce que nous appelons l'action refaire. De même selon la définition de l'annulation, si un patch est déjà refait, l'action refaire n'a pas un effet, sinon il faut réappliquer son effet. En conséquence, le système doit savoir si un patch est activé ou non. De plus, le système doit savoir combien de fois un patch a été défait ou refait. Nous expliquons ce besoin au moyen d'un exemple illustré à la figure 2.20.

Supposons que deux sites $Pair1$ et $Pair2$ ont reçu le même patch $P1$. Parallèlement, les deux sites décident d'annuler ce patch. En conséquence, $Pair1$ génère une modification

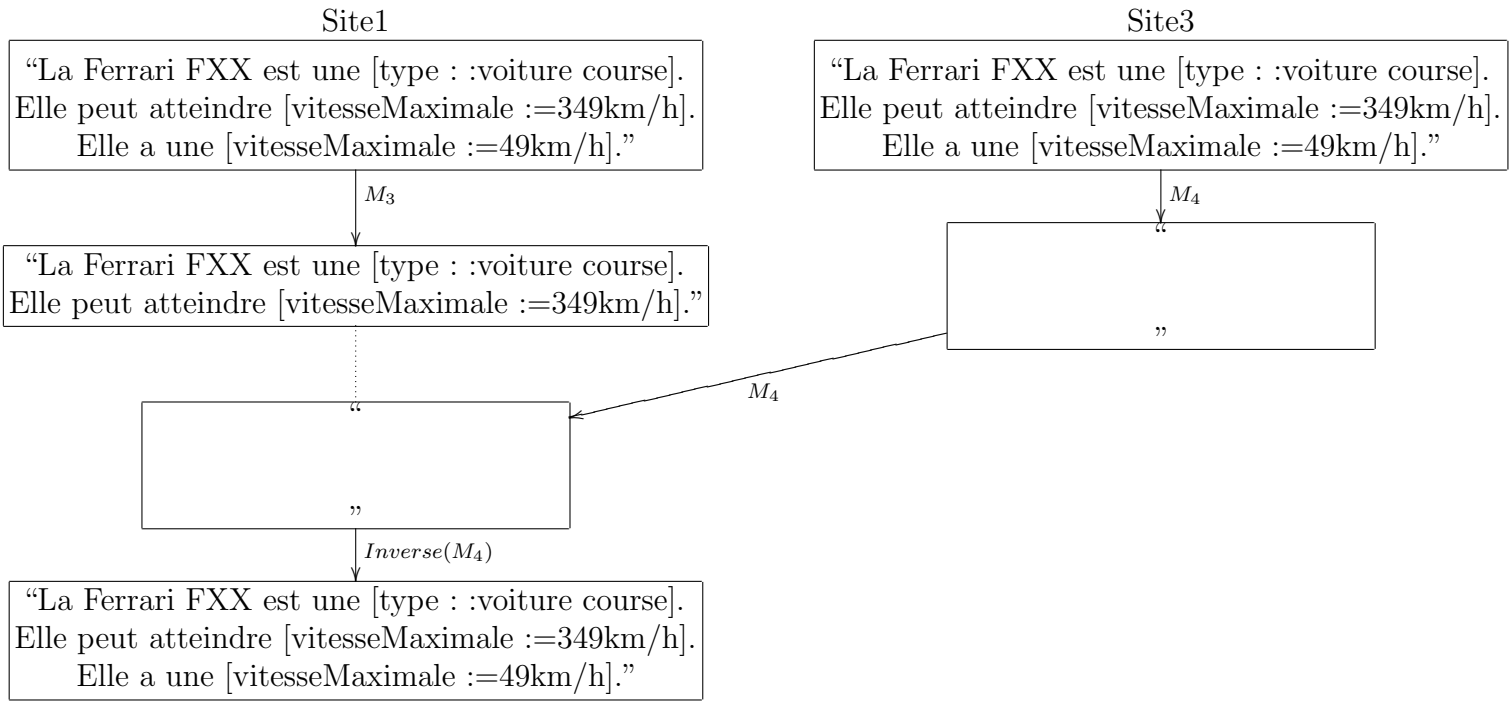

FIGURE 2.19 – Annuler une modification en utilisant la modification inverse

M_1 tandis que $Pair2$ génère M_2. Ensuite, $Pair2$ choisit de refaire le patch $P1$ pour une certaine raison, ce qui génère la modification M_3. Enfin, chaque pair reçoit les modifications de l'autre. Le site $Pair1$ a reçu deux modifications M_2 et M_3 . Si le système ne détecte pas que le patch $P1$ est déjà annulé, il réappliquera M_3. Par conséquent, le patch $P1$ est activé et son effet est réappliqué sur le site $Pair1$, ce qui n'est pas le cas sur le site $Pair2$. Ce scénario montre une violation de la définition de l'annulation puisque le système n'est pas capable de savoir l'état du patch à tout moment et comment les actions défaire et refaire changent son état. Comme solution, nous proposons d'étendre certains éléments de Swooki. L'idée de base est de fournir au système un moyen de détecter le nombre de fois qu'un patch est défait ou refait.

2.9.2 Modification dans le modèle de Swooki

L'intégration de ce mécanisme d'annulation dans Swooki nécessite l'ajout du composant d'annulation (voir figure 2.21), avec une légère extension de certains éléments. Ce mécanisme est conçu pour permettre aux utilisateurs de supprimer ou de réinsérer l'effet de certains changements dans les pages wikis et par conséquent de mettre à jour leurs annotations sémantiques dans l'entrepôt RDF.

Dans Swooki, la sauvegarde des changements d'un utilisateur sur une page wiki sémantique produit des opérations d'insertion et de suppression de lignes. Ces opérations composent un patch. Ce patch est reçu localement et propagé vers les autres pairs pour y

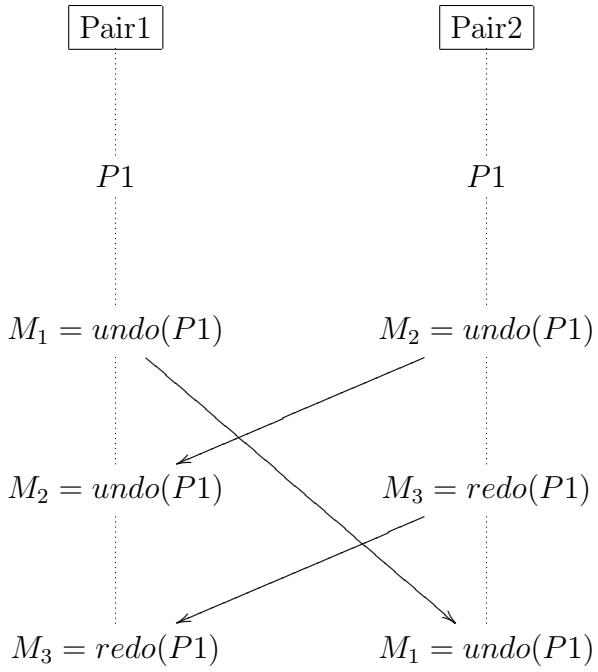

FIGURE 2.20 – Des messages Undo et Redo concurrents

être intégré. Un patch intégré est stocké dans une liste de patchs. Pour fournir à Swooki un mécanisme d'annulation, nous proposons d'étendre sa liste des patchs en associant un *patchDegree* à chaque patch de cette liste. La liste des patchs devient un ensemble de pairs : $< patch, patchDegree >$ où le *patchDegree* indique si ce patch a un effet ou non dans l'état courant de la page wiki sémantique. Nous avons choisi arbitrairement d'affecter le *patchDegree* de un durant l'insertion d'un patch dans cette liste, de le décrémenter de un durant l'exécution d'une action défaire et de l'incrémenter de un durant une action refaire de ce patch. Par conséquent, un patch est annulé et n'a aucun effet si son *patchDegree* est strictement inférieur à un, il n'est jamais supprimé de la liste des patchs. En fait, les patchs ne sont jamais supprimés de la liste des patchs, cependant leur effet est supprimé du modèle de la page wiki sémantique comme s'il n'existait pas. Si nous reprenons le scénario présenté à la figure 2.20, le site $Pair1$ calcule un degré de zéro et le patch $P1$ ne sera pas restauré. Un même résultat est obtenu sur les deux sites tout en respectant la définition de l'annulation.

Dans Swooki, une ligne dans le modèle de la page wiki sémantique est définie comme suit : < IdLigne, contenu, degré, visibilité >. Elle n'est jamais supprimée, elle est seulement rendue invisible dans la page wiki sémantique. La visibilité d'une ligne est déterminée par l'attribut de visibilité qui est un booléen. Nous changeons la valeur de l'attribut *visibilité* d'une ligne en une valeur entière afin de permettre au système de détecter si une ligne est visible ou non après plusieurs actions défaire et refaire des patchs. Dans notre cas, une ligne est visible si son attribut de visibilité est strictement positif. Une suppression d'une ligne met son attribut de visibilité à zéro, par contre une action d'annulation (ou refaire)

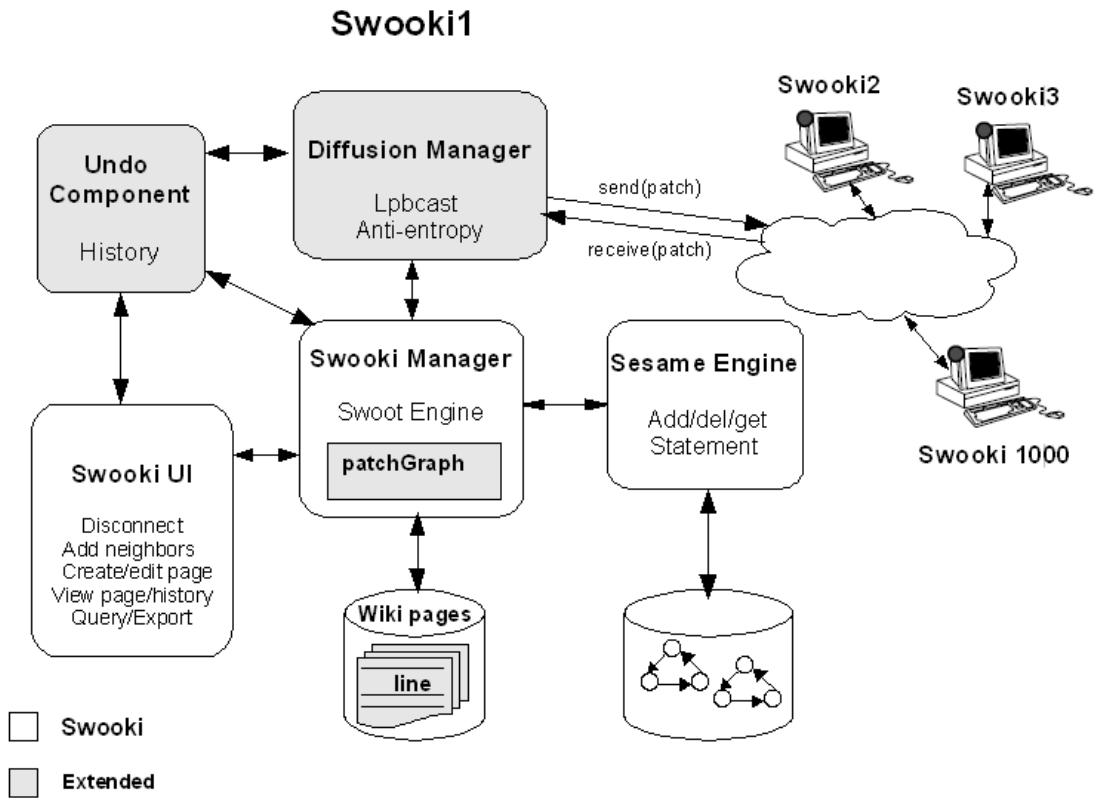

FIGURE 2.21 – L'architecture de Swooki supportant un mécanisme d'annulation

le décrémente (ou l'incrémente de un).

Messages

Comme pour les modifications habituelles, les actions défaire et de refaire doivent être propagées à tous les autres sites. Par conséquent, nous étendons le gestionnaire de diffusion de Swooki afin de tenir compte des messages d'*undo* et de *redo*. Nous définissons trois types de messages :

Do message : Ce message contient un patch, les opérations contenues dans ce patch sont les opérations d'insertion et de suppression de lignes définies dans Swooki. Pour un message exécutable, le patch est ajouté dans le *patchGraph* avec un *patchDegree* égal à un. Un message *Do* est exécutable si ses opérations satisfont leurs préconditions comme défini dans Swooki, les opérations de ce patch sont ensuite intégrées selon leur type.

Undo message : Ce message contient l'identifiant du patch sur lequel l'annulation sera appliquée. Un message *Undo* est exécutable si le patch sur lequel il est appliqué existe dans le *patchGraph*.

Redo message : De même pour le message Redo, il contient l'identifiant du patch sur lequel le redo sera appliqué. Il a la même précondition d'exécution que celle du message Undo.

FIGURE 2.22 – Annulation des changements à partir de l'historique

Dans Swooki, les opérations d'édition des utilisateurs se limitent à des insertions et des suppressions de lignes dans les pages wikis sémantiques. Dans le but d'intégrer le mécanisme d'annulation dans Swooki, nous définissons deux nouvelles actions que nous appelons défaire et refaire. La première vise à annuler l'effet du patch de l'état actuel de la page wiki sémantique, tandis que la deuxième réapplique son effet. L'action défaire d'un patch génère un Undo message, par contre l'action refaire génère un Redo message. Ces actions sont offertes soit au moyen de l'historique des pages comme dans les wikis traditionnels (voir figure 2.22), soit à partir d'une interface spéciale (voir figure 2.23) que nous avons conçu pour visualiser la liste des patches.

Algorithmes

Cette section décrit les différents algorithmes développés pour la réception et l'intégration des messages. Quand un message est reçu, il est ajouté dans une file d'attente.

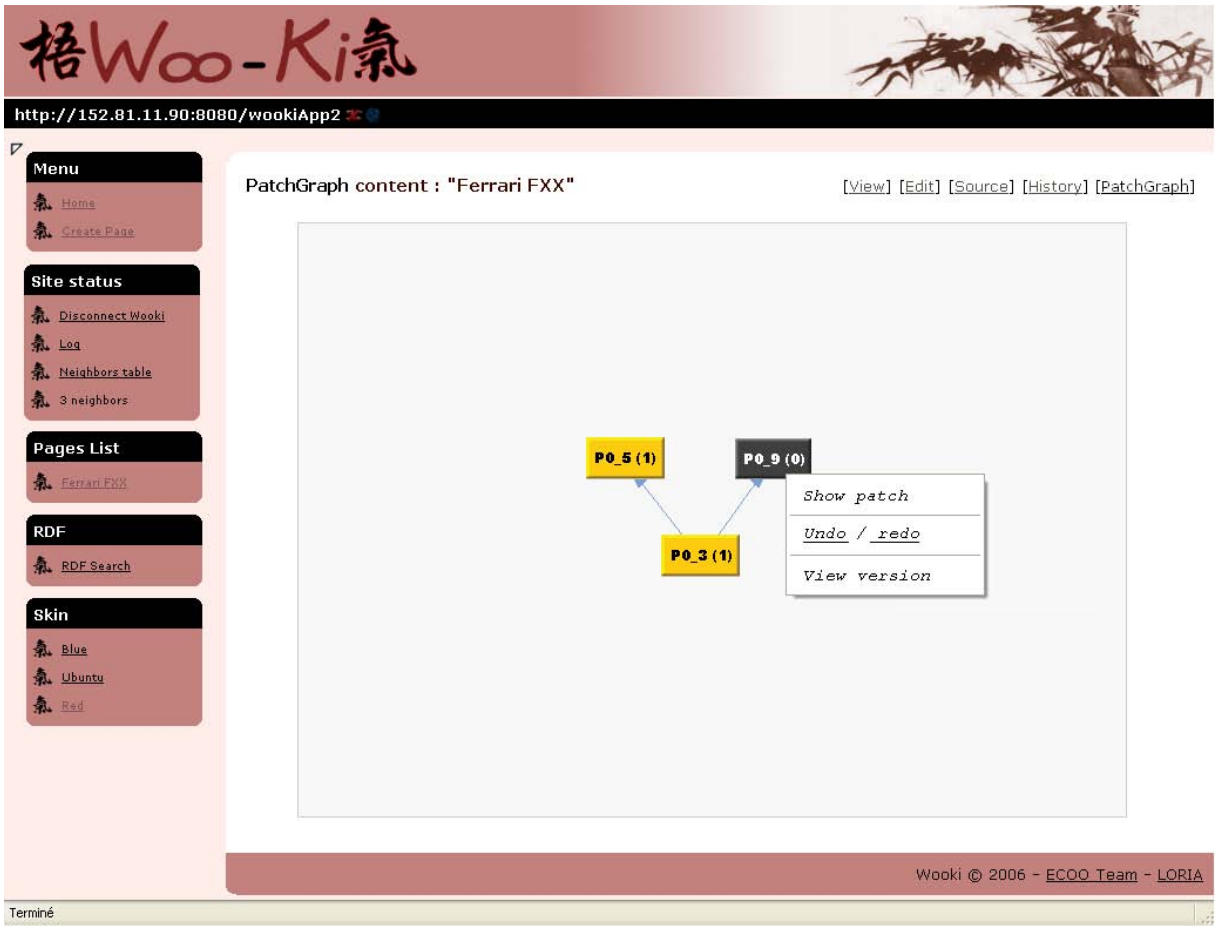

FIGURE 2.23 – Annulation des changements dans l'interface de la liste des patches

Ensuite, chaque message de cette file est testé afin de déterminer s'il est exécutable ou non. Un message Do est exécutable si ses opérations satisfont leurs préconditions comme c'est défini à la section 2.6.1. Cependant un message Undo ou Redo est exécutable si le patch sur lequel le message est appliqué existe dans le patchGraph. Dans le cas d'un message exécutable, son message d'information *mInfo* qui contient le commentaire d'un utilisateur est ajouté dans l'historique de la page wiki sémantique. Ensuite, le message est intégré selon son type. L'algorithme de réception s'arrête lorsque la file d'attente ne contient aucun message exécutable.

```
1 uponReceive(message):
2   addMsgWaitingQueue(message)
3   stop := false
4   while(stop = false) do
5       stop := true
6       for msg in MsgWaitingQueue do
7           if isExecutable(msg)= true then
8               stop := false
```

```
9                writeHistoryEvent(mInfo)
10               switch(type(msg))
11                   “Do”:
12                       addInPatchGraph(getPatch(msg))
13                       for op in patch do
14                           if type(op) = insert
15                               integrateIns(op)
16                           else
17                               integrateDel(op)
18                       done
19                   “Undo”:
20                       integrateUndo(getPatchID(msg))
21                   “Redo”:
22                       integrateRedo(getPatchID(msg))
23           endif
24   endif
```

L'intégration des opérations d'insertion et de suppression sont faites par l'algorithme de Swooki présenté à la section 2.7. Par la suite, nous détaillons l'intégration des messages Undo et Redo. Lors de l'intégration d'un message Undo, le patch considéré est extrait de la liste des patchs. Ensuite, son *patchDegree* est décrémenté de un. Si sa valeur devient égale à zéro, l'effet du patch est désactivé de la page wiki sémantique et de l'entrepôt des annotations sémantiques si nécessaire.

```
1 integrateUndo(patchId):
2     patch := getPatch(patchId)
3     decreasePatchDegree(patch)
4     if getPatchDegree(patch) = 0 then
5         disable(patch)
```

De même, lors de l'intégration d'un message Redo sur un patch, ce patch est extrait et son *patchDegree* est incrémenté de un. Si sa valeur est égale à un, l'effet du patch est activé pour la page wiki sémantique et dans l'entrepôt des annotations sémantiques si nécessaire.

```
1 integrateRedo(patchId):
2     patch := getPatch(patchId)
3     increasePatchDegree(patch)
4     if getPatchDegree(patch) = 1 then
5         enable(patch)
```

L'activation et la désactivation des patchs sont traitées comme suit : Durant l'activation d'un patch, la ligne de chacune de ses opérations est récupérée. Selon le type de l'opération, cette ligne sera ainsi traitée. L'entrepôt des annotations sémantiques est affecté si cette ligne contient des annotations. Dans le cas d'une opération d'insertion, la

ligne est réinsérée dans le contenu de la page wiki sémantique au moyen de la méthode reinsertLine(). Tandis que pour une opération de suppression, cette ligne est supprimée par la méthode delLine().

```
enablePatch(patch):
  for op in patch do
      line := getLine(op)
          switch(type(op))
              " insert ":  reinsertLine(line)
              " delete ":  delLine(line)
```

La désactivation d'un patch débute par l'extraction de la ligne de chaque opération du patch. Pour une opération d'insertion, la ligne est supprimée. L'opération de suppression est traitée comme une réinsertion de la ligne.

```
disablePatch(patch):
  for op in patch do
      line := getLine(op)
      switch(type(op))
          " insert ":  delLine(line)
          " delete ":  reinsertLine(line)
```

D'une façon générale, l'intégration d'une opération est traitée en deux étapes : (1) l'intégration du texte et (2) l'intégration des annotations sémantiques. L'insertion d'une ligne est exécutée à la position NextIDentifier dans la page wiki sémantique, après avoir trouvé cette position. La ligne est insérée avec un degré de visibilité égal à un. Ensuite, les annotations sémantiques contenues dans cette ligne sont intégrées dans l'entrepôt RDF.

```
insertLine(line):
  insert(PageID, line, NextIdentifier)
  integrateInsRDF(line)
```

Afin de supprimer une ligne, son degré de visibilité est décrémenté de un. Ce degré permet au système de détecter si une ligne est visible ou non après plusieurs actions défaire et refaire des patchs. Si la valeur de ce degré devient égale à zéro, c'est-à-dire que la ligne est devenue invisible dans le contenu de la page wiki sémantique alors une intégration de la suppression des annotations de cette ligne est exécutée. Une ligne dans Swooki n'est jamais supprimée du modèle de la page mais rendue invisible dans la page wiki sémantique.

```
delLine(lineID):
  line := getLine(lineID)
  decreaseVisibilityDegreeOfLine(line)
  if visibilityDegree(line) = 0 then
      integrateDelRDF(getContent(line))
```

L'opération reinsertLine() change la visibilité des lignes dans le modèle de la page wiki sémantique sans vraiment insérer des nouvelles lignes. Elle est l'opération inverse de delLine(). Cette opération incrémente le degré de visibilité de la ligne. Si sa valeur devient égale à un, c'est-à-dire la ligne est visible alors les annotations sémantiques de cette ligne sont intégrées dans l'entrepôt RDF.

```
9  reinsertLine(lineID):
10   line := getLine(lineID)
11   increaseVisibilityDegreeOfLine(line)
12   if visibilityDegree(line) = 1 then
13       integrateInsRDF(getContent(line))
```

2.9.3 Correction du mécanisme d'annulation

Dans cette section, nous présentons la correction de notre mécanisme d'annulation. Nous montrons qu'il respecte le modèle CCI défini à la section 2.6.

Théorème 4. *Tous les messages commutent, i.e. si un message $M1$ ne dépend pas causalement d'un autre message $M2$, ni $M1 \rightarrow M2$, ni $M2 \rightarrow M1$ alors $M1$ et $M2$ commutent.*

Démonstration. Les messages $M1$ et $M2$ peuvent être soit de type message Do qui contient un patch, soit un message Undo/Redo qui incrémente ou décrémente de un le degré d'un patch. Dans le cas d'un message Do, les auteurs dans [OUMI06, WUM07] ont montré que les opérations d'insertion commutent. Pour le couple d'opérations "insert-delete", les opérations ne peuvent pas référencer la même ligne sinon nous aurons *insert* $\rightarrow$ *delete* ou l'inverse. Puisque ces deux opérations ont un effet sur des lignes différentes, par conséquent elles commutent. Alors les messages Do commutent. Comme les messages Undo décrémentent le degré du patch tandis que les messages Redo l'incrémentent, il est évident qu'ils commutent. Finalement, les couples Do-Undo et Do-Redo commutent pour les mêmes raisons. □

Théorème 5. *Lorsque tous les sites ont reçu tous les messages, ils possèdent les mêmes patchGraphs.*

Démonstration. Chaque pair a reçu les mêmes messages Do, donc chaque patchGraph contient les mêmes patchs. Nous devons également assurer que chaque patch a le même degré dans le patchGraph de chaque pair. Puisque le gestionnaire de diffusion de messages dans Swooki garantit qu'un message Undo/Redo ne peut pas être livré avant le message Do correspondant et grâce au théorème 4, tous les patchs ont le même degré sur tous les sites. □

Définition 12. *Le degré Degré(l_i, S) d'une ligne l_i est :*

$$D(l_i, S) = A(P_0) - A(P_1) - ... - A(P_n)$$

où

$$A(P_i) = \begin{cases} 1 & \textit{si le degré de } P_i \textit{ est supérieur à 1} \\ 0 & \textit{sinon} \end{cases}$$

et P_0 est le patch qui a inséré la ligne l_i et $P_1, ...P_n$ sont les patches qui contiennent une opération "delete" de cette ligne.

Théorème 6. *Le degré d'une ligne est exprimé en utilisant le patchGraph.*

Démonstration. Selon notre définition, P_0, $P_1, ..., P_n$ sont les seuls patchs qui réfèrent à la ligne l_i. Chacun de ces patchs contient exactement une opération qui affecte l_i. L'exécution de P_0 insère l_i avec un degré de un. Le patch P_0 incrémente de un le degré de l_i si cette ligne est visible, i.e. si le degré de P_0 est égal au moins un. Similairement, chaque patch $P_1, ..., P_n$ le décrémente de un si elle est visible. Par conséquent, nous pouvons facilement exprimer le degré de l_i en utilisant le patchGraph.

□

Théorème 7 (Cohérence éventuelle)**.** *Swooki assure la cohérence éventuelle, i.e. tous les sites après avoir reçu tous les messages, chaque page a le même contenu et chaque entrepôt RDF possède les mêmes triplets.*

Démonstration. Grâce à [OUMI06, WUM07], nous connaissons que chaque pair héberge les mêmes lignes dans le même ordre. Puisque chaque pair a le même patchGraph (Theorem 5), et le degré de la ligne dépend seulement du patchGraph (Theorem 6) alors chaque ligne possède le même degré sur tous les pairs. Dans [SMRM09], nous avons montré que si le document assure la cohérence éventuelle, l'entrepôt RDF assure aussi cette propriété. □

Nous présentons la définition de l'annulation comme définie dans [Sun02] :

Définition 13. *Annuler une modification fait retourner le système à l'état qu'il aurait atteint si cette modification n'a jamais été produite.*

Théorème 8. *Le mécanisme d'annulation respecte la définition de l'Undo (Définition 13).*

Démonstration. Le respect de la définition de l'Undo se traduit par le respect de l'équation suivante pour toute ligne : Degré($l_i, S \circ P1 \circ ... \circ Pi \circ ... \circ Pn \circ undo(Pi)$) =Degré($l_i, S \circ P1 \circ ... \circ Pi \circ undo(Pi) \circ ... \circ Pn$). Cette équation est vérifiée si le message Undo(Pi) commute avec tous les messages qui suivent Pi. En effet, il y a deux cas : soit ces messages concernent un autre patch dans ce cas Undo(Pi) commute, soit ils concernent le même patch dans ce cas ils sont forcément des messages undo et redo qui commutent nécessairement. □

2.10 Synthèse

Dans ce chapitre, nous avons proposé Swooki une instanciation du modèle de réplication optimiste pour les wikis sémantiques. Swooki est le premier wiki sémantique sur réseau P2P. Il est composé d'un ensemble de serveurs wikis sémantiques interconnectés. Chaque pair réplique un nouveau type de données composé de pages wikis sémantiques et d'annotations. Le problème principal est d'assurer la cohérence des répliques dans un environnement P2P. Dans les éditeurs collaboratifs, le modèle de cohérence utilisé est CCI. Ce modèle est basé sur trois critères : la préservation de la Causalité, de la Convergence et de l'Intention. Malheureusement, la définition de l'intention n'a pas été définie pour le type de données des wikis sémantiques. Dans ce chapitre, nous avons proposé une définition de l'intention pour les wikis sémantiques. Nous avons proposé également un algorithme de réplication optimiste qui assure la cohérence CCI. Nous avons étendu l'algorithme Woot [WUM08] pour synchroniser les annotations par effet de bord. De plus, nous avons développé un mécanisme d'annulation de groupe dans Swooki qui permet d'annuler toute modification.

Swooki combine les avantages des wikis sémantiques et ceux des wikis pair-à-pair. D'un point de vue système distribué, Swooki permet une édition collaborative qui s'auto-organise, qui passe à l'échelle, qui supporte la tolérance aux pannes et qui résiste à la censure. Il permet l'édition et le partage de la connaissance d'une façon distribuée et supporte le mode d'édition déconnectée. D'un point système collaboratif, Swooki supporte partiellement l'édition multi-synchrone puisqu'elle est contrôlée par le système. La propagation des changements et la synchronisation des répliques sont sous le contrôle de Swooki. Malheureusement, Swooki ne permet pas de supporter les procédés d'édition collaborative. Pour supporter l'édition multi-synchrone et les procédés d'édition collaborative, nous proposons une autre instanciation du modèle de réplication optimiste dans le chapitre 3. Nous développons un autre wiki sémantique P2P qui est basé sur un réseau ami-à-ami (friend-to-friend en anglais) dans lequel les utilisateurs ne collaborent qu'avec leurs amis.

Chapitre 3

DSMW : Wikis sémantiques multi-synchrones distribués

Sommaire

Dans ce chapitre, nous proposons Distributed Semantic MediaWiki (DSMW) une autre instanciation du modèle de réplication optimiste pour les wikis sémantiques. Contrairement à Swooki, DSMW supporte l'édition multi-synchrone et l'implémentation des procédés d'édition collaborative. DSMW est composé de N serveurs wikis sémantiques interconnectés répliquant des pages wikis contenant des annotations sémantiques.

- Dans DSMW, nous préservons les mêmes opérations d'édition des pages wikis et les définitions de leur intention présentées dans Swooki.
- Nous changeons le mécanisme de propagation de changements. Nous développons un modèle de collaboration Publier/Souscrire basé sur des flux similaires aux flux de syndication RSS. Nous montrons que ce mécanisme de propagation préserve la causalité. Nous définissons ce modèle sous forme d'une ontologie pour exploiter l'historique des changements par des requêtes sémantiques. D'autre part, la propagation des changements dans ce modèle est sous le contrôle des utilisateurs. Par conséquent, l'édition multi-synchrone et l'implémentation des procédés d'édition sont supportées.
- Nous remplaçons l'algorithme de synchronisation Woot par Logoot [WUM09b] puisque le dernier réduit la complexité de la génération, de l'intégration des opérations et du stockage des pages wikis sémantiques.

Nous montrons que DSMW assure le modèle de cohérence CCI en préservant les mêmes définitions de l'intention définies dans Swooki. Dans la section 3.1, nous commençons par une introduction de DSMW. Nous décrivons son modèle de collaboration Publier/Souscrire dans la section 3.1.1 et son fonctionnement du point de vue utilisateur dans la section 3.1.2. Dans la section 3.2, nous déroulons un scénario de collaboration dans DSMW. Nous présentons l'architecture et l'implémentation de DSMW dans la section 3.3. Ensuite, nous détaillons dans la section 3.4.1 le modèle de données et les algorithmes nécessaires pour maintenir les flux et synchroniser les différents pairs du DMSW. Enfin, nous terminons par une synthèse de DSMW.

3.1 Introduction

Un wiki sémantique multi-synchrone distribué (DSMW) est composé d'un ensemble de serveurs wikis sémantiques interconnectés. Il permet aux utilisateurs de construire leurs propres réseaux de coopération. Les pairs du réseau forment un réseau ami-à-ami sémantique dans lequel ils sont connectés les un aux autres au moyen d'un modèle publier/-souscrire des flux. Dans ce modèle, les utilisateurs d'un pair publient lorsqu'ils souhaitent leurs changements dans des flux. Ils invitent ensuite leurs amis à souscrire à ces flux en leur communiquant leurs identifiants afin de récupérer leurs contenus. Les flux ressemblent aux flux de syndication (RSS) [RSS], ils contiennent effectivement les changements des utilisateurs. Ils sont utilisés pour publier localement les changements et de les propager entre les pairs amis du réseau. Un wiki sémantique multi-synchrone distribué propose un nouveau modèle de collaboration qui supporte l'édition multi-synchrone d'où vient son

nom. Plusieurs sous-réseaux d'amis peuvent avoir lieu (voir figure 3.1).

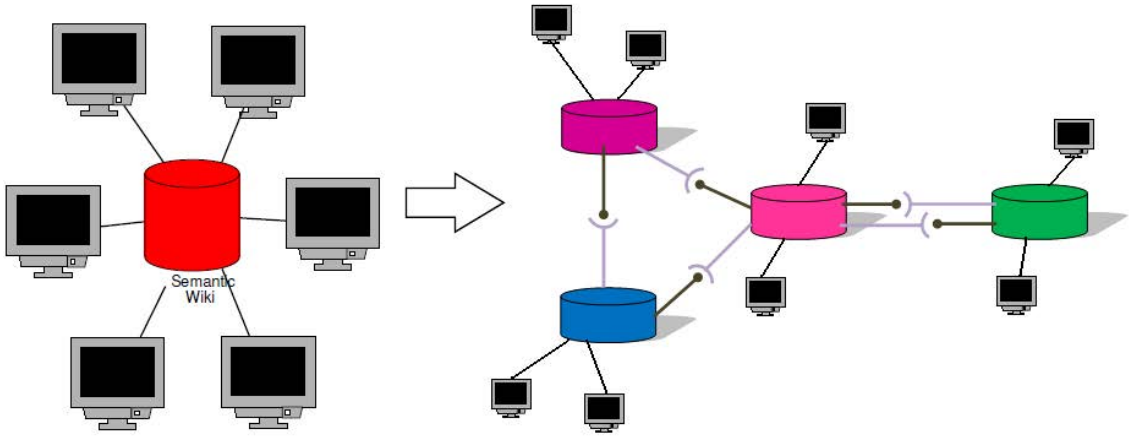

FIGURE 3.1 – Wiki sémantique multi-synchrone distribué

Un utilisateur est libre de choisir quand et quoi publier et surtout à qui. La propagation des changements dans DSMW est basée sur le modèle de collaboration Publier/Souscrire présentée dans la section 3.1.1.

3.1.1 Modèle de collaboration Publier/Souscrire

Nous proposons un modèle de collaboration nommé *Publier/Souscrire* dans DSMW. Dans ce modèle, les liens entre les différents pairs sont établis par les utilisateurs et le résultat est la construction d'un réseau ami-à-ami sémantique. Dans DSMW, la réplication des données et la communication entre les serveurs sémantiques se font à travers des *flux* similaires aux flux de syndication RSS [RSS]. Les flux RSS sont souvent utilisés par les sites d'actualité ou les blogs pour présenter les titres des dernières informations consultables en ligne. Généralement, un flux RSS contient un titre (souvent celui d'un article), une description de l'article et un lien vers le site concerné. Son contenu est produit automatiquement en fonction des mises à jour d'un site web, il est généré périodiquement pour que le sommaire soit toujours à jour. Par contre, les flux d'un DSMW contiennent les changements sur les pages wikis sémantiques des pairs i.e. les opérations d'insertion et de suppression des lignes. Ces flux sont utilisés pour publier les changements sur une ou plusieurs pages dans un pair DSMW et récupérer des changements distants d'autres pairs amis.

3.1.2 Fonctionnement de DSMW

Dans cette section, nous présentons le fonctionnement de DSMW. Nous décrivons l'édition d'une page wiki sémantique, la publication des changements d'un pair DSMW dans un flux et la souscription à ce flux par un autre pair.

La sauvegarde des pages génère des patchs La figure 3.2 montre l'édition de la page 'Hello'. La sauvegarde de cette page sur un pair du DSMW génère un patch. Un patch est représenté par une page wiki sémantique spéciale (voir figure 3.3). Cette page affiche la liste des opérations d'insertion et de suppression de lignes contenues dans ce patch.

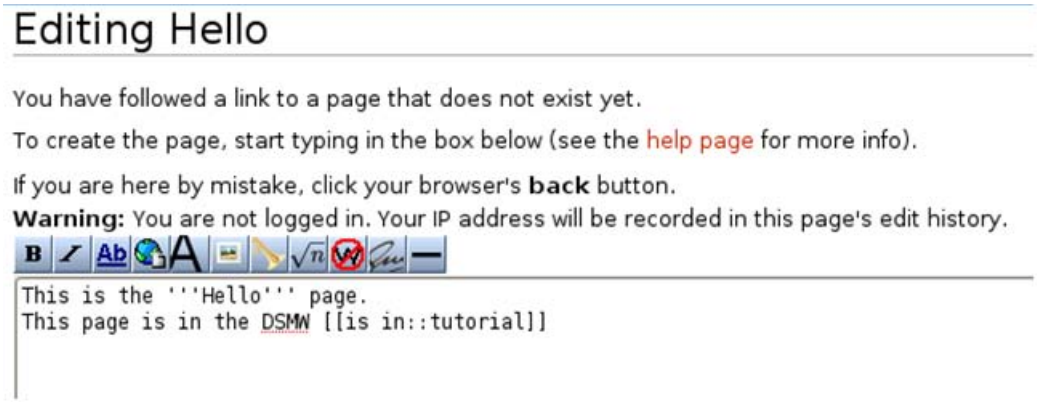

FIGURE 3.2 – Editer une page DSMW

Un pair DSMW publie des flux de patchs

Les patchs peuvent être sélectionnés en utilisant des requêtes sémantiques et publiés dans des flux *PushFeed*. Ces flux sont également représentés par des pages wikis sémantiques spéciales (voir figure 3.5). Un flux *PushFeed* publie localement les patchs d'une ou plusieurs pages wikis sémantiques. A chaque publication dans un flux, une page wiki sémantique spéciale appelée *ChangeSet* est créée (voir figure 3.4). Elle contient les patchs générés sur les pages du flux depuis la dernière publication. Afin de permettre à un autre pair de récupérer le contenu d'un flux *PushFeed*, l'utilisateur doit communiquer l'url de ce flux et son nom à l'autre pair.

Un pair DSMW souscrit à des flux de patchs Chaque pair peut souscrire à n'importe quel nombre de flux et commencer à synchroniser leur contenu. Il doit être autorisé par les pairs propriétaires de ces flux. La souscription à un flux *PushFeed* est faite par un flux *PullFeed* qui est lui aussi représenté par une page wiki sémantique spéciale (voir figure 3.6). A chaque *Pull* sur un flux *PullFeed* d'un pair, les derniers patchs distants publiés dans le flux *PushFeed* associé sont récupérés et intégrés localement. Des pages spéciales qui sont une page *ChangeSet* et des pages *Patchs* sont créés et les patchs sont intégrés automatiquement dans les pages wikis sémantiques correspondantes.

DSMW assure le modèle de cohérence CCI, les pairs DSMW qui intègrent les mêmes

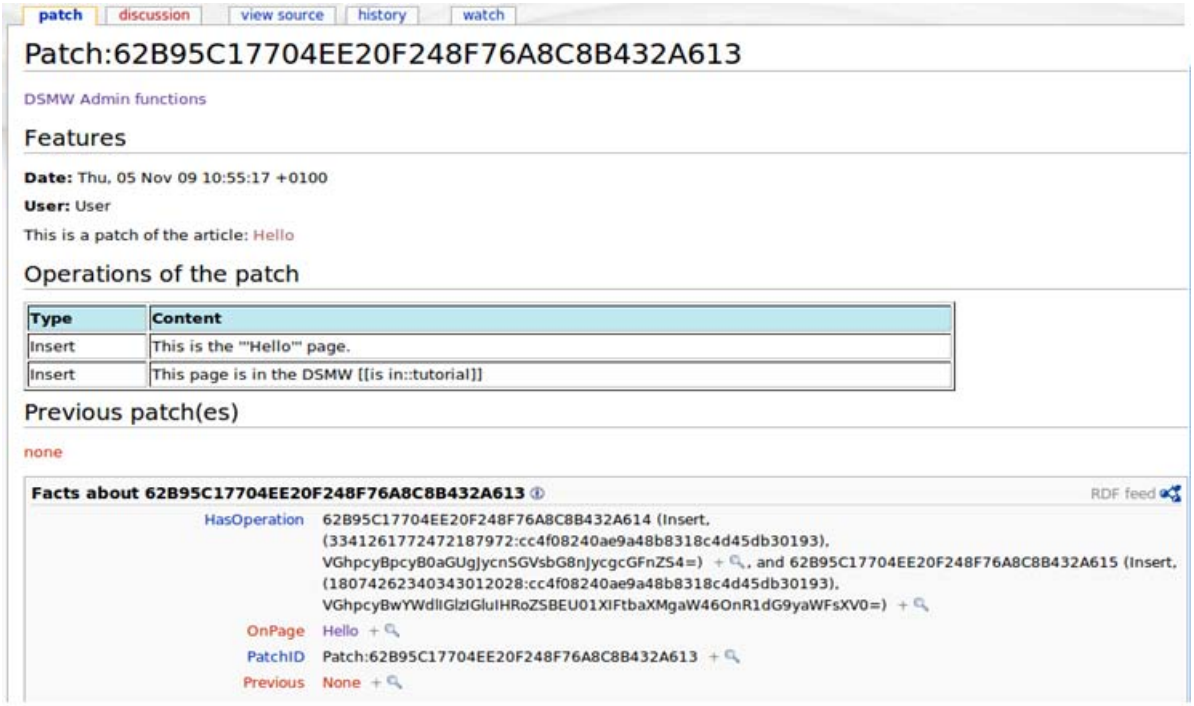

FIGURE 3.3 – La page wiki sémantique spéciale d'un patch

opérations convergent vers les mêmes copies (pages wikis sémantiques et entrepôt sémantique) tout en préservant l'intention et la causalité des opérations.

3.2 Scénarios de Collaboration

Cette section présente deux scénarios de collaboration dans un wiki sémantique multi-synchrone distribué. Dans ces scénarios, deux professeurs collaborent en utilisant chacun un wiki sémantique multi-synchrone pour préparer des cours, des exercices et des examens. Ensuite, ils envisagent de rendre publique ces cours pour les étudiants et finalement intégrer les commentaires et les corrections des étudiants.

Scenario 1 : Collaboration entre professeurs Considérons deux professeurs $prof_1$ et $prof_2$ qui collaborent dans la préparation de leur cours (voir figure 3.7). Chaque professeur possède son propre wiki sémantique multi-synchrone, $site_1$ pour $prof_1$ et $site_2$ pour $prof_2$. Le scénario de la collaboration se déroule en plusieurs étapes comme suit :

– Etape 1 (Editer) : $prof_1$ édite trois pages wikis sémantiques $lesson_1$, $exercises_1$ et $examen_1$ sur son site $site_1$. Par exemple, la page $lesson_1$ éditée par $prof_1$ contient trois lignes :

```
 Introduction:
Dans le travail "multi-synchrone", les parties travaillent en parallèle.
```

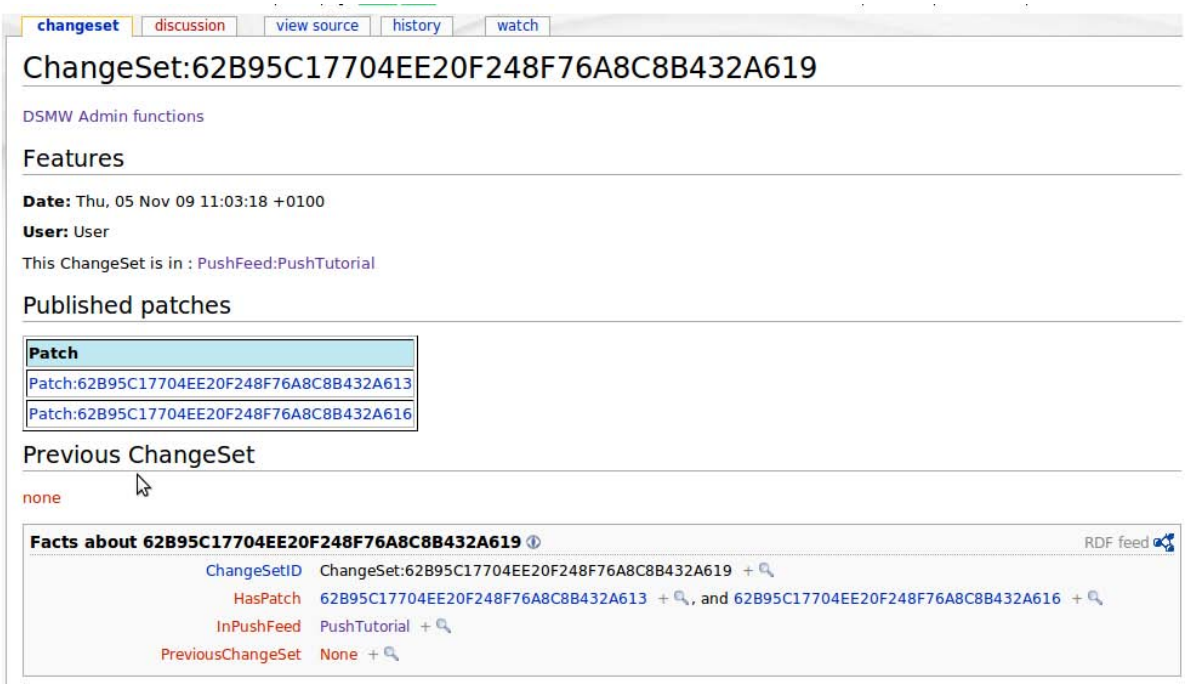

FIGURE 3.4 – La page wiki sémantique spéciale d'un *ChangeSet*

```
[Catégorie:: Leçon] [année:2010]
```

- Etape 2 (Publier) : $prof_1$ veut répliquer la page $lesson_1$, publier ses modifications apportées sur cette page et la rendre accessible pour le $prof_2$. Pour cela il crée un flux url_1 qui contient l'ensemble de ces changements. Puis, il communique l'adresse du flux au $prof_2$.
- Etape 3 (Souscrire) : Le $prof_2$ s'inscrit à ce flux en créant son propre flux url_2 qui sert à récupérer le contenu du flux url_1. Le résultat de cette étape est : (1) la création d'un lien entre les pairs des deux professeurs et (2) la réplication de la page wiki sémantique $lesson_1$ et de ses annotations sémantiques i.e. une création d'une copie locale sur le $site_2$ ayant le même contenu que celle de la $lesson_1$ du $prof_1$. Une réplication des annotations sémantiques a lieu dans l'espace du stockage des annotations du $site_2$.
- Etape 4 (Editer) : Le $prof_1$ insère sur son site une nouvelle ligne "Ce qui permet un travail en isolation." à la dernière position de la page $lesson_1$. Le résultat obtenu est comme suit :

```
 Introduction:
Dans le travail "multi-synchrone", les parties travaillent en parallèle.
[Catégorie:: Leçon] [année:2010]
Ce qui permet un travail en isolation.
```

En parallèle, le $prof_2$ sur le $site_2$ édite sa copie locale de la page $lesson_1$. Il insère

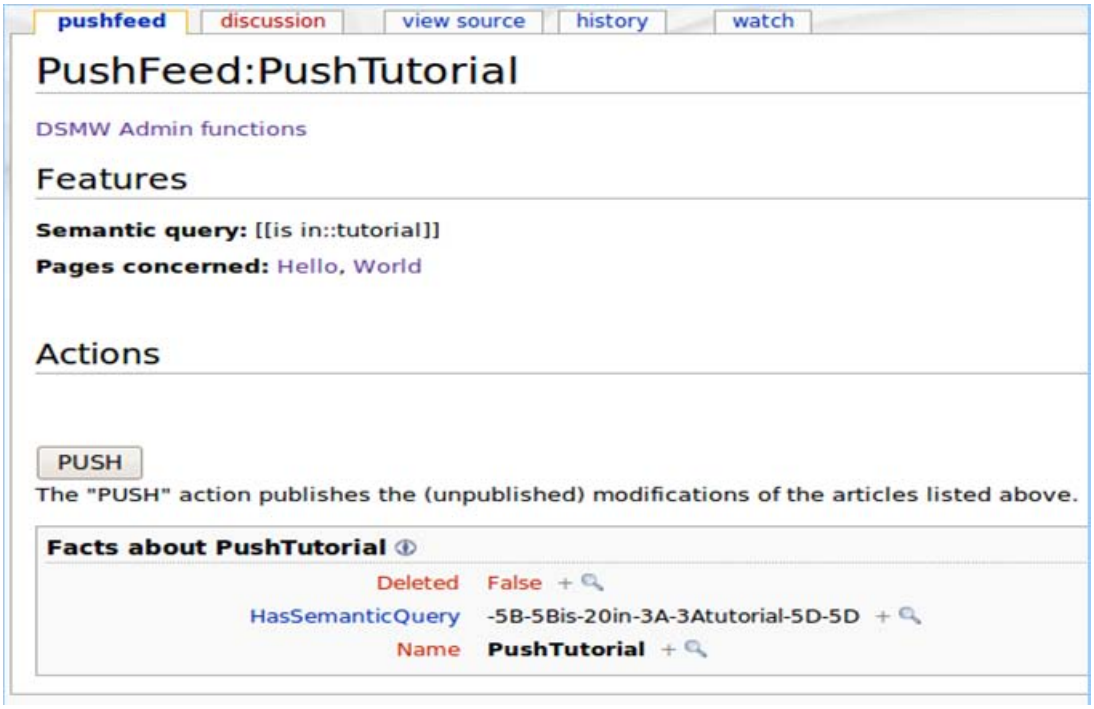

FIGURE 3.5 – La page wiki sémantique spéciale d'un flux $PushFeed$

deux nouvelles dernières lignes dans la page, celle-ci contient maintenant cinq lignes :

```
 Introduction:
Dans le travail "multi-synchrone", les parties travaillent en parallèle.
[Catégorie:: Leçon] [année:2010]
Ce mode est basé sur des phases de divergence et de synchronisation.
Des copies multiples avec différentes vues de l'état des données peuvent exister
pour les données répliquées.
```

Dans cette étape, les copies de la page $lesson_1$ des deux sites divergent. Des changements concurrents sur la page ont eu lieu, ils ne sont pas directement publiés. Ces changements seront publiés, propagés et intégrés puis visibles entre les sites ultérieurement.

– Etape 5 (Publier) : A son tour, le $prof_2$ veut partager ses modifications sur $lesson_1$ avec $prof_1$. Il crée le flux url_3 et publie ses modifications. $prof_1$ s'inscrit à ce flux en créant le flux url_4, récupère son contenu et finalement les modifications publiées par $prof_2$ seront intégrées dans la copie locale $lesson_1$ du $prof_1$. Le processus d'intégration fusionne les modifications distantes avec celles qui sont simultanées générées localement par le $prof_1$. Le processus d'intégration doit assurer la convergence de toutes les copies du $lesson_1$ et de ses annotations après l'intégration de toutes les opérations générées par les deux sites $site_1$ et $site_2$. De même, le $prof_2$ intègre les changements récents produits par le $prof_1$. Le résultat final obtenu sur les deux sites

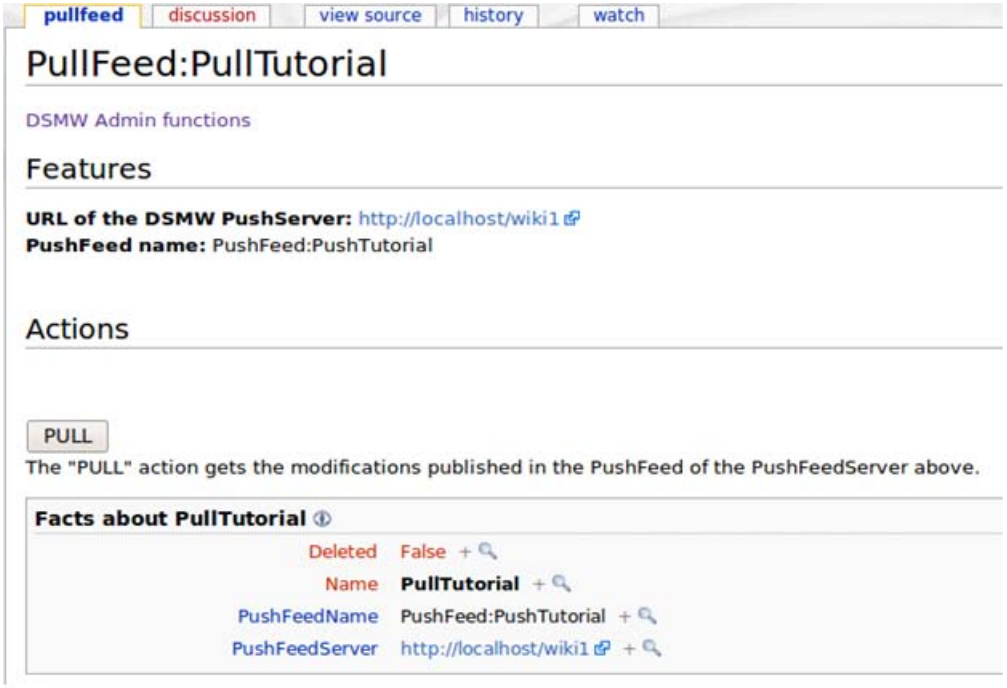

FIGURE 3.6 – La page wiki sémantique spéciale d'un flux $PullFeed$

est le suivant :

```
 Introduction:
Dans le travail "multi-synchrone", les parties travaillent en parallèle.
[Catégorie:: Leçon] [année:2010]
Ce qui permet un travail en isolation.
Ce mode est basé sur des phases de divergence et de synchronisation.
Des copies multiples avec différentes vues de l'état des données peuvent exister
pour les données répliquées.
```

Le tableau 3.1 représente une partie de ce scénario. Le processus de collaboration entre $prof_1$ et $prof_2$ continue via une séquence d'éditions suivie par des publications et échange des changements entre les deux sites. Les avantages de ce modèle de collaboration sont : (1) il ne requiert pas l'utilisation d'un serveur central pour collaborer et (2) la propagation des changements est sous le contrôle des professeurs, i.e chaque professeur modifie sa copie locale en isolation, publie et récupère les changements distants quand il veut. Ce processus de collaboration n'est supporté ni dans les wikis sémantiques classiques, ni dans Swooki.

Scenario 2 : Collaboration entre professeurs et étudiants Dans ce scénario, le $prof_1$ souhaite rendre disponible la page $lesson_1$ pour ses étudiants tout en continuant à la modifier et à la corriger. Afin de fournir les cours à ses étudiants, $prof_1$ les publie sur un site wiki sémantique publique $pubSite$ qui peut être soit son propre site ou un site de l'université par exemple. Les données du $pubSite$ ne peuvent pas être modifiées par

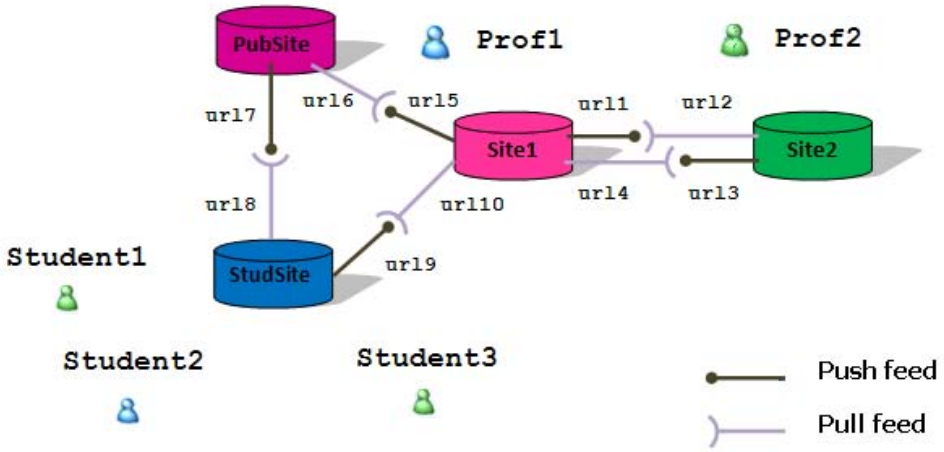

FIGURE 3.7 – Scénario de collaboration dans un wiki sémantique multi-synchrone distribué

les étudiants, elles sont accessibles en lecture seule. En revanche, $prof_1$ veut intégrer les "feedbacks" des étudiants. Toutefois, $prof_1$ crée des flux pour propager ces données. Les étudiants les récupèrent et les corrigent lorsque c'est nécessaire. Finalement, $prof_1$ intègre leurs corrections sur le $site_1$. Le déroulement du scénario de la collaboration est comme suit :

- Etape 1 (Publier) : $Prof_1$ crée le flux url_5 sur son site $site_1$.
- Etape 2 (Souscrire) : Le $prof_1$ se connecte à $pubSite$, crée le flux url_6 et récupère le contenu du flux url_5. Le résultat de cette étape est la création de la page $lesson_1$ sur $pubSite_1$ ayant le même contenu que celles du $site_1$. A partir de cette étape, le $prof_1$ peut publier ses cours progressivement d'une façon simple, facile et satisfaisante en créant le flux url_7
- Etape 3 (Souscrire) : Les étudiants peuvent disposer de leur propre serveur DSMW $StudSite$, ils peuvent par exemple récupérer la page $lesson_1$ du site $pubsite$ et la personnaliser. Ils réalisent cela en s'inscrivant sur le flux du $prof_1$ après avoir créer le flux url_8. Ils peuvent collaborer directement entre eux sans le professeur ou bien décider de communiquer leur modifications au $prof_1$ en créant le flux url_9 et les publier.
- Etape 4 (Souscrire) : Le $prof_1$ peut récupérer les modifications des étudiants vers son serveur privé en cas où il trouve des modifications pertinentes. A son tour, il peut rendre ces modifications disponibles sur son site public.

La collaboration entre le professeur et les étudiants est constituée d'un cycle d'édition, de publication et d'échange de certaines pages wikis sémantiques. Le professeur peut apporter des améliorations continues à ses cours, faire une intégration continue des modifications et rendre visible seulement les modifications cohérentes. Sans le support du mode multi-synchrone toutes les modifications incrémentales seront visibles directement

$Site_1$	$Site_2$	Commentaires
Edite($lesson_1$)		
Edite($exercises_1$)		
Edite($exam_1$)		
Créer(f_1,q_1)		créer flux pour publier les modifications
		sur les pages dans la requête q_1 ($lesson_1$)
Push(f_1)		publier les modifications disponibles dans f_1
	Inscrire(f_1, f_2)	souscription au flux f_1
Edite($lesson_1$)	Edite($lesson_1$)	édition multi-synchrone
Push(f_1)		nouvelles modifications sont disponibles
	Créer(f_3,q_2)	créer un flux pour publier les modifications

TABLE 3.1 – Scénario de collaboration multi-synchrone entre deux professeurs

par les utilisateurs et par les moteurs des requêtes sémantiques. Chaque participant que ce soit un professeur ou un étudiant souhaite contrôler la visibilité de leurs modifications et contrôler l'intégration des modifications des autres.

Ce processus de collaboration n'est pas supporté dans les wikis sémantiques classiques. Dans Swooki, ce processus de collaboration ne peut pas être supporté, les utilisateurs ne peuvent pas décider des pages à répliquer, elles les sont toutes par défaut.

En résumé, ces scénarios montrent qu'un wiki sémantique multi-synchrone distribué (DSMW) offre facilement le mode d'édition multi-synchrone et un support de procédés.

3.3 Architecture et implémentation du système

Dans cette section, nous présentons l'architecture du wiki sémantique multi-synchrone distribué. Nous développons également son implémentation.

3.3.1 Architecture

La figure 3.8 illustre les différents composants d'un pair du wiki sémantique multi-synchrone distribué. Ces composants sont décrits comme suit :

L'Interface Utilisateur Nous avons associé à chaque page wiki sémantique un onglet "dsmw (nombre de patches)". En cliquant sur cet onglet, une page s'ouvre (voir figure 3.9), elle contient des informations calculées sur l'état de la page wiki associée. Ces informations sont :

- La liste des patchs (ensemble d'opérations) appliqués à la page. Les patchs provenant des pairs distants sont affichés avec une couleur différente que ceux produits localement.

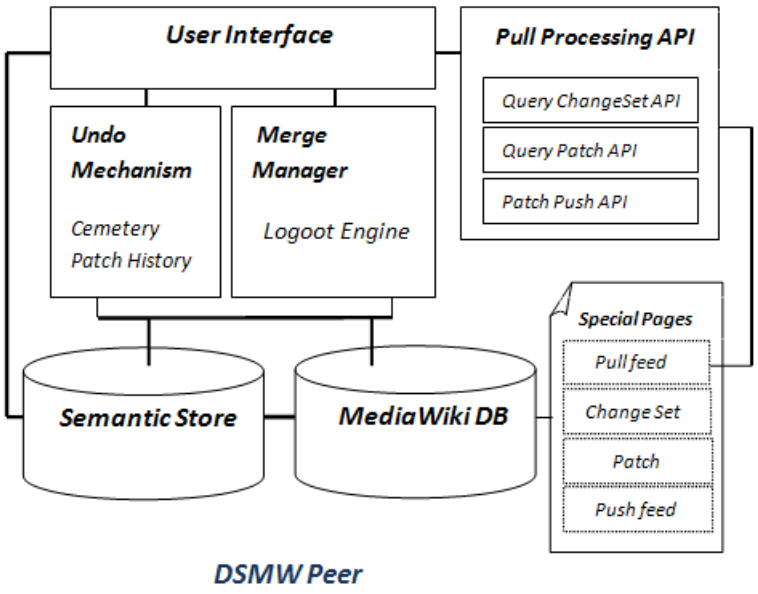

FIGURE 3.8 – Architecture du wiki multi-synchrone distribué

- La liste des flux $PushFeed$ dans lesquels cette page a été publiée (et le quotient des patchs publiés/non publiés).
- La liste des flux $PullFeed$ d'où les patchs ont été récupérés.

Nous avons défini une page spéciale appelée "DSMW Administration" qui permet la création, la supperssion et le maintient des flux $PushFeed$ et $PullFeed$ sur un pair. Elle permet aussi la publication et la récupération des changements des flux entre les pairs connectés (voir figure 3.10).

Elle contient donc la liste de ces flux et quelques informations. Pour la liste des $PullFeed$, elle affiche :

- Pages : le nombre de pages concernées par le flux correspondant.
- Patchs distants : le nombre de patchs publiés.
- Patchs locaux : le nombre des patchs locaux des pages concernées.

Pour la liste des $PushFeed$, la page d'administration indique :

- Pages : le nombre de pages concernées par le flux correspondant.
- Tous les patchs : la somme de tous les patchs (publiés + non publiés).
- Patchs publiés : le nombre de patchs publiés (pushed).
- Patchs non publies : le nombre des non publiés.

PULL processing Ce composant sert à récupérer les changements des flux distants d'autres pairs en exécutant des requêtes distantes. Il est composé de trois sous-composants qui servent chacun à récupérer une sous partie des données.

- Le premier sert à trouver les $ChangeSet$ publiés après la dernière récupération du flux.

FIGURE 3.9 – Une page 'dsmw (nombre patches)' associée à la page 'Hello'

- Le deuxième sert à trouver les *Patchs* de chaque *ChangeSet*.
- Le troisième sert à copier le contenu de ces *Patchs* qui sont des *Operations* localement.

Ce composant est le seul moyen d'échanger les données entre les pairs du réseau.

Gestionnaire de fusion Ce composant est responsable de la génération et de l'intégration des opérations qui représentent les changements. Il implémente l'algorithme Logoot, il effectue donc une fusion automatique des changements dans les pages wikis sémantiques.

Stockage des données Ce composant est constitué d'une base de données qui stocke séparément les pages wikis sémantiques et leurs annotations. Il contient différents espaces de nommage (PullFeed, PushFeed, ChangeSet, Patch, Operation) pour séparer les pages wikis sémantiques ordinaires de celles qui sont spéciales.

Mécanisme d'annulation Ce composant est responsable de l'annulation de toute modification à tout moment, il est développé par[WUM09a]. Il utilise le même principe d'annulation adopté dans Swooki. Mais il se distingue par une structure de données appelée cimetière qui sauvegarde des informations concernant des lignes ayant un degré de visibilité négatif c'est à dire supprimées. Cette structure remplace la notion des pierres tombales de Swooki puisque les lignes dans Logoot sont vraiment supprimées. L'avantage de cette structure est qu'elle est beaucoup plus petite que les pierres tombales. Ce mécanisme est en cours d'implémentation dans notre système.

3.3.2 Implémentation

L'implémentation du wiki sémantique multi-synchrone distribué a été effectuée en étendant Semantic MediaWiki [Sem], le logiciel utilisé par Semantic Wikipedia. Nous

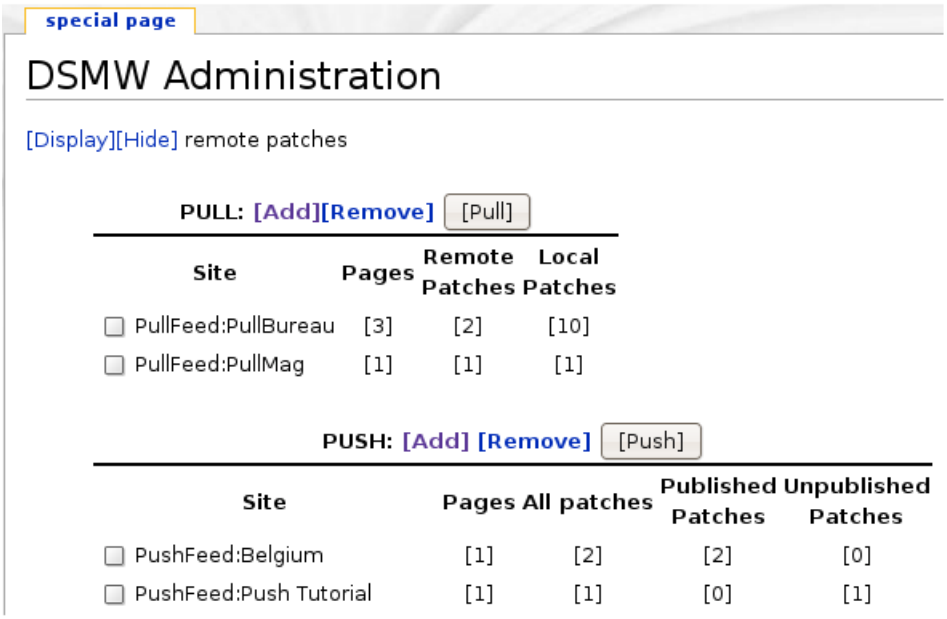

FIGURE 3.10 – La page d'administration spéciale pour la gestion des flux

avons adopté ce choix pour pouvoir construire rapidement notre système à partir d'un système existant qui est déjà utilisé. Nous appelons ce système, *Distributed Semantic MediaWiki* (DSMW). Notre système est donc implémenté en PHP et sous une licence libre GPL. Divers installations du DSMW sont disponibles. Une vidéo et le code source du système sont disponibles sur le site du DSMW [DSM09]. Pour implémenter l'algorithme Logoot dans DSMW, nous avons ajouté deux tables dans la base de données existante. Une table nommée "DSMW model" utilisée pour stocker le modèle des pages wikis sémantiques et une autre nommée "params" pour stocker les informations du pair telles que l'identifiant du site et son horloge logique. Les flux ont été implémentés sous formes de pages wikis non éditables.

3.4 Modèle de données et algorithmes du DSMW

Cette section présente le modèle de données et les algorithmes nécessaires pour le wiki sémantique multi-synchrone distribué (DSMW). Nous avons défini le modèle de données sous forme d'une ontologie dans le but d'exécuter des requêtes sémantiques et de raisonner sur le modèle lui même. Cela permet d'exécuter facilement les requêtes suivantes : "trouver tous les changements non publiés", "trouver les changements publiés dans un flux précis" et "trouver l'ensemble des changements non publiés concernant une page wiki sémantique particulière", etc.

3.4.1 Modèle de données du DSMW

Nous définissons le modèle de données du DSMW comme une ontologie. Une représentation graphique de cette ontologie est donnée à la figure 3.11. L'ontologie a été créée avec l'outil Protégé [GMF+02] et elle est visualisée en utilisant graphViz [Gra] un logiciel de visualisation d'ontologies. Elle décrit les changements dans les pages et la manière dont ils sont traités, stockés et propagés. La définition du modèle de données comme une ontologie permet d'interroger et de raisonner sur le modèle lui-même et permet aussi de faire des extensions futures. Par exemple, il est possible d'écrire des requêtes de type :

1. donner la liste des changements publiés et non publiés,
2. donner la liste des changements publiés dans un flux et
3. donner la liste des changements non publiés d'une page donnée.

Par la suite, nous présentons l'ontologie DSMW et nous détaillons ses concepts et ses propriétés.

- **WikiSite** : ce concept correspond à un serveur wiki sémantique. Puisque les pages wikis et leurs annotations sémantiques seront répliquées sur plusieurs serveurs wikis, chaque site du réseau doit être identifié d'une façon unique et doit posséder les propriétés suivantes :
  - $siteID$: cet attribut contient l'URL du site.
  - $logicalClock$: cet attribut prend une valeur numérique. Chaque serveur wiki sémantique maintient une horloge logique, cette horloge est utilisée pour identifier les changements (les patchs et les opérations) d'une façon unique dans tout le réseau.
  - $hasPush$: le co-domaine de cette propriété est le $pushfeed$. Un site wiki peut avoir plusieurs flux de type $pushfeed$.
  - $hasPull$: le co-domaine de cette propriété est le $pullfeed$. De même, un site wiki peut avoir plusieurs flux de $pullfeed$.
  - $hasPage$: le co-domaine de cette propriété est une page wiki sémantique. Un site wiki possède plusieurs pages wikis sémantiques.
- **Document** : un document peut être une image, un fichier audio ou vidéo ou n'importe quel type de fichier qui peut être inséré dans une page wiki sémantique.
- **SemanticWikiPage** : ce concept correspond à une page wiki sémantique normale ayant les propriétés suivantes :
  - $pageID$: cet attribut contient l'URL de la page.
  - $hasContent$: le co-domaine de cette propriété est une chaîne de caractères, elle contient du texte et les annotations sémantiques intégrées dans la page wiki sémantique.
  - $head$: cette propriété pointe sur le dernier patch appliqué sur la page. Elle laisse une trace du dernier changement fait sur la page.

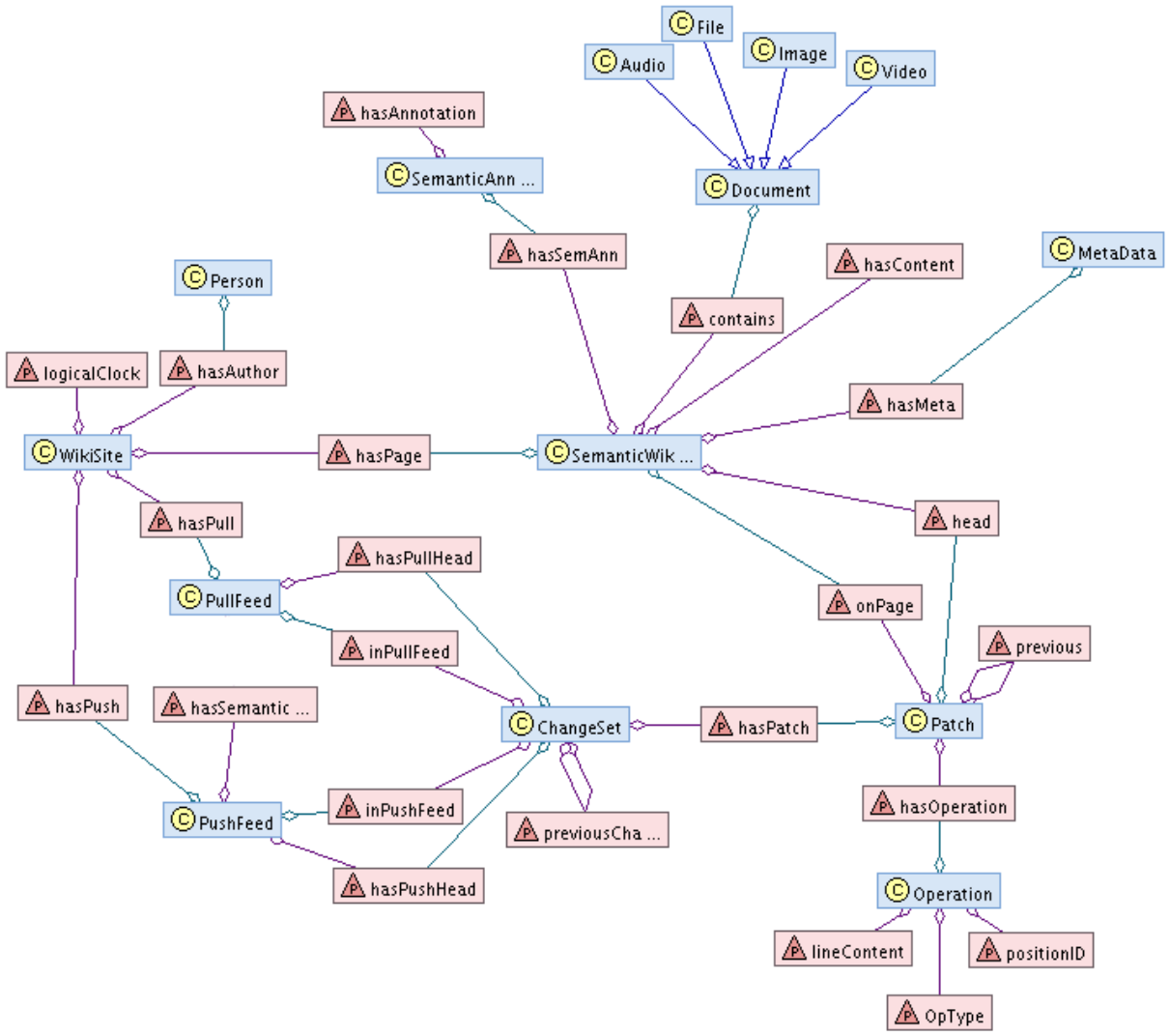

FIGURE 3.11 – Ontologie DSMW

- **Operation** : elle représente un changement dans une ligne d'une page wiki sémantique. Dans notre modèle, il y a deux types d'opérations d'édition : *insérer* et *supprimer*. Une mise à jour est considérée comme une suppression de l'ancienne valeur suivie par une insertion de la nouvelle valeur. Une opération a les propriétés suivantes :
  - $operationID$: cet attribut contient l'identifiant unique de l'opération, il est calculé par la concaténation de deux valeurs : l'identifiant du site et l'horloge logique, i.e. $operationID = \text{concat}(Site.siteID, Site.logicalClock++)$.
  - $opType$: cet attribut contient le type de l'opération qui peut être soit une insertion soit une suppression.
  - $positionID$: elle dénote la position de la ligne dans la page. Cet identifiant est calculé par l'algorithme Logoot [WUM09b] lors de la génération de l'opération.

  - *lineContent* : est une chaîne représentant du texte et les annotations sémantiques intégrées dans la ligne.
- **Patch** : un patch est un ensemble d'opérations. Il est calculé durant la sauvegarde des modifications faites sur une page wiki sémantique en utilisant l'algorithme Logoot. Un patch possède les propriétés suivantes :
  - *patchID* : est un identifiant unique du patch. Sa valeur est calculée par une concaténation de l'identifiant du site avec l'horloge logique.
  - *onPage* : le co-domaine de cette propriété est la page où le patch a été appliqué.
  - *hasOperation* : cette propriété pointe sur les opérations générées durant la sauvegarde de la page c'est-à-dire sur les opérations contenues dans le patch.
  - *previous* : pointe sur le patch précédent. Cette propriété est utilisée pour relier les patchs.
- **ChangeSet** : il contient un ensemble de patchs. Ce concept permet de regrouper les patchs générés sur plusieurs pages wikis sémantiques. Par conséquent, un utilisateur peut publier simultanément des modifications concernant plusieurs pages dans le but de préserver un état cohérent du wiki. Le *ChangeSet* a les propriétés suivantes :
  - *changeSetID* : est un identifiant unique du *changeSet*. Sa valeur est calculée par une concaténation de l'identifiant du site avec l'horloge logique.
  - *hasPatch* : cette propriété pointe sur les patchs ajoutés dans les flux.
  - *previousChangeSet* : pointe sur le *changeSet* précédent.
  - *inPushFeed* : le co-domaine de cette propriété est un *pushfeed*. Elle indique le *pushfeed* qui publie le *changeSet*.
  - *inPullFeed* : le co-domaine de cette propriété est un *pullfeed*. Elle indique le *pullfeed* qui contient le *changeSet*.
- **Push Feed** : ce concept est un flux utilisé pour publier les changements d'un *WikiSite*. Il hérite des propriétés du concept *SemanticWikiPage* et définit ses propres propriétés :
  - *hasPushHead* : cette propriété pointe sur le dernier *changeSet* publié.
  - *hasSemanticQuery* : cette propriété contient une requête sémantique. La requête détermine le contenu du *pushfeed*.
- **Pull Feed** : ce concept est utilisé pour récupérer les changements des sites distants. Un *pullfeed* est relié à un seul *pushfeed*, il récupère les données publiées dans ce flux. Un *pullfeed* est aussi une page wiki sémantique spéciale. Il hérite les propriétés du concept *SemanticWikiPage* et définit ses propres propriétés :
  - *hasPullHead* : cette propriété pointe sur le dernier *changeSet* récupéré.
  - *relatedPushFeed* : cette propriété relie un *pullfeed* à l'URL de son *pushfeed* associé.

L'ontologie DSMW est définie en OWL DL [HPSMW07], elle permet d'interroger et de raisonner sur les *Patchs*, *ChangeSet*, ... Par exemple, nous pouvons écrire des requêtes sémantiques en utilisant l'ontologie DSMW comme suit : (1) trouver les patchs publiés

dans un push feed :

$$Published \equiv \exists(hasPatch^{-1}).\exists(inPushFeed^{-1}).PushFeed$$

et (2) trouver les patchs non publiés dans un push feed :

$$unPublished \equiv Patch \sqcap \neg(\exists(hasPatch^{-1}).\exists(inPushFeed^{-1}).PushFeed)$$

3.4.2 Algorithmes

Notre système DSMW repose sur les fonctions suivantes *Save*, *Create Push*, *Push*, *Create Pull* et *Pull*. Par la suite, nous détaillerons ces opérations pour un serveur wiki sémantique multi-synchrone que nous appelons *site*.

L'opération Save Durant la sauvegarde d'une page wiki sémantique, un algorithme de $Diff$ calcule la différence entre la version sauvegardée et la version précédente de la page. Ensuite, Logoot génère les opérations correspondantes. L'idée de base de Logoot est de calculer des identifiants uniques pour les lignes. Un identifiant représente la position de la ligne dans la page wiki sémantique. Les opérations générées sont une insertion et une suppression. La dernière supprime physiquement la ligne de la page wiki. Après la sauvegarde d'une page, des instances d'*Operation* et une instance de *Patch* de l'ontologie DSMW sont créées. Les opérations sont intégrées localement, puis éventuellement publiées dans des flux de patchs.

```
1 On Save(page : String, page:String) :−
2   Patch(pid=concat(site.siteID, site . logicalClock + +))
3   foreach op ∈ Logoot(page, page) do
4     Operation(opid=concat(site.siteID,site . logicalClock+ +))
5     hasOperation(pid, opid)
6   endfor;
7   previous(pid, page.head)
8   head(page, pid)
9   onPage(pid, page)
```

A titre d'exemple, la figure 3.12 représente le patch associé à la page $lesson_1$ sur le site $site_1$ du scénario de collaboration présenté à la section 3.2. Le nombre affiché dans la boite en bas à droite du $wikiSite1$ de la figure indique la valeur de l'horloge logique du site.

L'opération CreatePushFeed La communication entre les pairs du DSMW se fait au moyen des flux. La publication et la propagation des changements sont sous le contrôle des utilisateurs. L'opération *CreatePushFeed* permet la création d'un flux $pushfeed$. Un $pushfeed$ est une page wiki sémantique spéciale contenant une requête sémantique qui spécifie les données à publier.

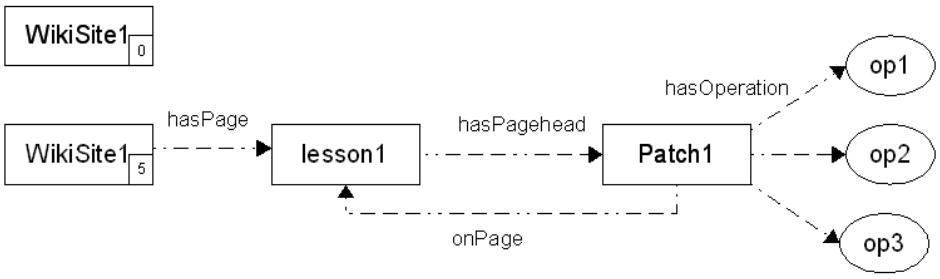

FIGURE 3.12 – L'état du $WikiSite1$ après la sauvegarde de la page $lesson_1$

```
1 on CreatePushFeed(name : String, request:String):−
2   PushFeed(name)
3   hasSemanticQuery(name, request)
4   hasPush(site, name)
5   call Push(name)
```

Un flux $pushfeed$ est utilisé pour publier les changements d'un serveur wiki sémantique multi-synchrone. Par conséquent, les sites autorisés peuvent accéder aux données publiées dans ce $pushfeed$. L'opération *CreatePushFeed* appelle l'opération *Push*.

L'opération Push L'opération *Push* crée un *changeSet* qui contient les changements effectués sur les pages concernées par la requête sémantique du flux $pushfeed$. Ensuite, le *changeSet* créé est ajouté dans le $pushfeed$. Premièrement, la requête sémantique est exécutée, puis les patchs des pages concernées résultants de la requête sont extraits. Ces patchs sont ajoutés au *changeSet* s'ils n'ont pas été déjà publiés dans le $pushfeed$.

```
1  On Push(name:String):−
2     ChangeSet(csid=concat(site.siteID,site . logicalClock++))
3     inPushFeed(csid, name)
4     let published ← {x/ ∃x ∃y ∧ inPushFeed(y,name) ∧ hasPatch(y,x) }
5     let patches ← {x/ ∃x ∀p ∈ execQuery(name.hasSemanticQuery) ∧ onPage(x,p)}
6     foreach patch ∈ {patches − published}
7       hasPatch(csid, patch)
8     endfor
9     previousChangeSet(csid, name.hasPushHead)
10    hasPushHead(name,csid)
```

L'opération CreatePullFeed Puisque la réplication des données et la communication entre les serveurs DSMW sont faites au moyen des flux, les flux $pullfeed$ sont créés afin de récupérer les changements des $pushfeed$ des pairs distants par un pair local autorisé.

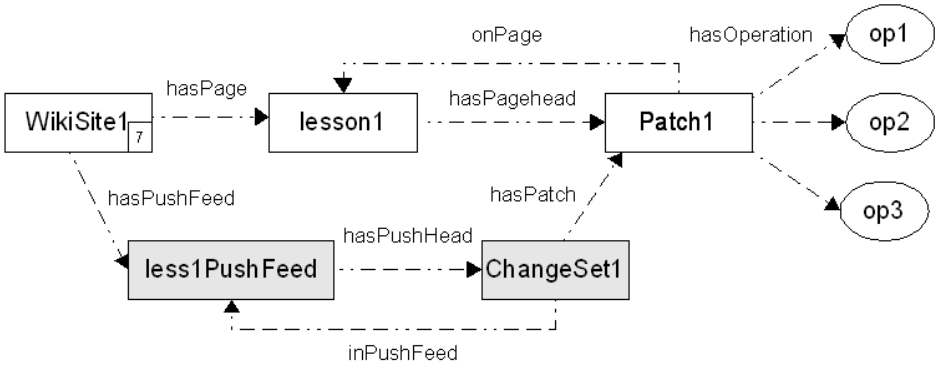

FIGURE 3.13 – Un push feed est créé et la page $lesson_1$ est publiée

Un $pullfeed$ est relié à un push feed, il récupère les données publiées dans le dernier.

```
1 On CreatePullFeed(name:String, url:URL):−
2     PullFeed(name);
3     relatedPushFeed(name,url);
4     call  Pull(name);
```

L'opération Pull Cette opération cherche les *changeSets* publiés mais qui ne sont pas encore récupérés. Elle ajoute ces *changeSets* au flux $pullfeed$ et les intègre dans les pages wikis sémantiques concernées du site en appelant la méthode Pull. Ensuite, le $pullfeed$ référence le dernier *changeSet* récupéré.

```
1 On Pull(name:String):−
2     while ((cs ← get(name.headPullFeed, name.relatedPushFeed)≠ null)
3         inPullFeed(cs,name)
4         call  Integrate(cs)
5         hasPullHead(name,cs)
6   endwhile
```

L'opération get Cette fonction permet d'extraire un changeSet d'un $PushFeed$.

```
1 On ChangeSet get(cs : ChangeSetId , url: pageID):−
2     if ∃x previousChangeSet(x,cs) return x
3     else return null;
```

La création du $pushfeed$ et la publication (le *pull*) de la page $lesson_1$ sur le site $WikiSite1$ dans le scénario présenté à la section 3.2 sont données à la figure 3.13. Le résultat est une instanciation des concepts $PullFeed$ et *changeSet* et de leurs propriétés.

L'opération d'Intégration L'intégration d'un *changeSet* est effectuée comme suit. Premièrement, tous les patchs de ce *changeSet* sont extraits. Puis chaque opération dans un patch est intégrée dans la page wiki sémantique correspondante grâce à l'algorithme Logoot [WUM09b]. Ensuite, les annotations sémantiques dans ces pages wikis sont extraites et stockées dans un entrepôt séparé. Les annotations sémantiques étant intégrées dans le contenu de ces pages, elles seront synchronisées par effet de bord. L'intégration des annotations sémantiques est faite par SMW comme suit : après l'intégration des changements dans les pages wikis sémantiques c'est-à-dire le patch dans une page, les annotations sémantiques de la page sont supprimées de l'entrepôt sémantique. Ensuite, le nouveau contenu de la page wiki est parsé et les annotations sémantiques trouvées sont insérées dans l'entrepôt. Cela garantit la préservation de l'intention et la convergence des annotations sémantiques dans l'entrepôt des pairs répliquants les mêmes pages wikis sémantiques.

```
1 Integrate(cs:ChangeSet):−
2     foreach patch ∈ cs
3     previous(patch, patch.onPage.head)
4    head(patch.onPage,patch)
5        foreach op ∈ hasOperation.patch
6           call logootIntegrate(op);
7        endforeach
8     endforeach
```

La création d'un $pull feed$ sur le site $WikiSite2$ et le pull de la page $lesson_1$ telle qu'elle a été publiée dans le push feed du $WikiSite1$ sont données dans la figure3.14. Différentes instances sont créées, au début le $pull feed$ a eu lieu, suivi par le *changeSet* et la copie du *Patch* et des ses *Operations*. Puisque le patch concerne une page wiki sémantique non existante, une instance d'une $SemanticWikiPage$ est créée où le patch est appliqué.

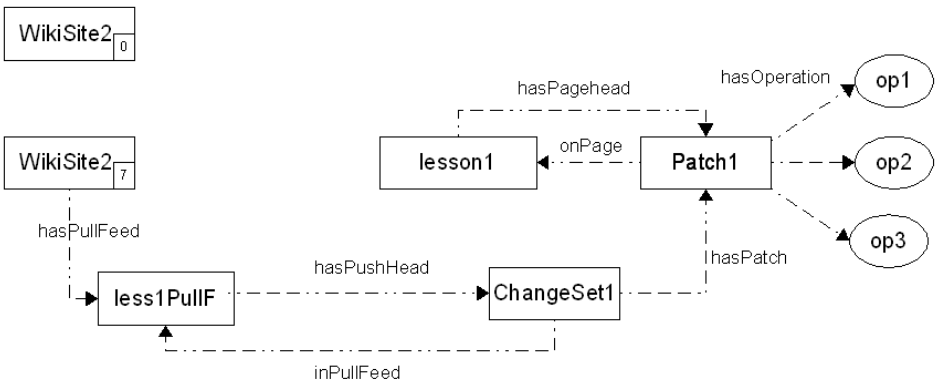

FIGURE 3.14 – Un pull feed est créé et la page $lesson_1$ est copiée du site WikiSite1

3.4.3 Modèle de Correction

Théorème 9. *Nos algorithmes assurent la causalité.*

Démonstration. Un site respecte toujours la causalité sur ses propres opérations puisqu'il génère ces opérations tout seul, donc ces opérations placées dans le $pushfeed$ sont dans l'ordre causal.

Par la suite, nous prenons comme hypothèse qu'un site S récupère les données (fait un *pull*) depuis N sites qui assurent la causalité et nous voudrons montrer que le site lui même assure la causalité.

Nous montrons que le site S ne peut pas violer la causalité en utilisant le raisonnement par absurde. Considérons que le site S viole la causalité i.e. dans le contenu du $pushfeed$ de ce site il y a une opération op_2 qui dépend causalement (dépendance de Lamport) d'une autre opération op_1 et que op_2 est placée avant op_1 dans ce feed.

Les opérations dans un $pushfeed$ ont deux sources différentes, soient elles sont locales générées par le site lui même ou bien elles sont reçues des sites distants. Etudions ces deux cas :

Opérations locales : les opérations locales respectent toujours la causalité, donc un site ne va pas violer la causalité.

Opérations distantes : Pour avoir l'opération op_2 placée à la position N dans le $pushfeed$ du site S, S doit récupérer (faire le *pull*) toutes les opérations qui précèdent op_2. Obtenir op_2 sans avoir op_1 signifie que op_1 ne figurait pas parmi les $N-1$ opérations qui précèdent op_2 dans le $pushfeed$, ce qui viole l'hypothèse que le site S fait le *pull* seulement à partir des N sites qui respectent la causalité.

Par conséquent, il n'est pas possible de trouver op_2 qui précède op_1. En conclusion, un site respecte toujours la causalité tout seul. Si ce site fait le pull de N sites qui assurent la causalité, il assure la causalité lui même. Par récurrence, tous les sites assurent la causalité pour n'importe quel N où N est un entier. Par conséquent, nos algorithmes assurent la causalité pour tous les sites du réseau. □

Théorème 10. *Nos algorithmes assurent le modèle CCI (la causalité, la Convergence, l'Intention et sa préservation).*

Démonstration. Nos algorithmes utilisent l'algoritme Logoot [WUM09b] pour la génération et l'intégration des opérations. Il est prouvé dans [WUM09b], que l'algorithme *Logoot* assure la convergence et préserve l'intention, si la causalité est préservée. Puisque nos algorithmes assurent la causalité entre les opérations (le théorème 9), par conséquent ils assurent aussi la convergence et préservent l'intention.

□

3.5 Synthèse

Dans ce chapitre, nous avons proposé DSMW une instanciation du modèle de réplication optimiste pour les wikis sémantiques. Nous avons défini le modèle de données de DSMW sous forme d'une ontologie qui permet d'exploiter l'historique au moyen des requêtes sémantiques. Nous avons développé un modèle Publier/Souscrire basé sur des flux pour la propagation des changements entre les pairs. La propagation est faite sous le contrôle des utilisateurs, ce qui permet de créer un réseau ami-à-ami sémantique. En effet, les utilisateurs choisissent avec qui ils souhaitent collaborer, ce qui permet de modéliser des procédés d'édition collaborative et de supporter l'édition multi-synchrone.

DSMW assure le modèle de cohérence CCI pour les pages wikis sémantiques et les entrepôts sémantiques.

Tout comme Swooki, DSMW supporte le modèle de cohérence CCI et préserve les mêmes intentions. Par contre, le modèle Publier/Souscrire de DSMW permet d'apporter l'édition multi-synchrone et l'implémentation des procédés d'édition collaborative qui manquaient à Swooki.

Chapitre 4

Conclusions et perspectives

Dans cette thèse nous avons instancié le modèle de réplication optimiste dans le cadre des wikis sémantiques. Nous avons produit deux systèmes Swooki et DSMW : Swooki adresse plus particulièrement les problèmes du passage à l'échelle et de la tolérance aux pannes, DSMW s'intéresse plus aux problèmes de représentation des procédés.

1. Swooki est une instanciation du modèle de réplication optimiste pour des données composées de pages wikis et d'annotations. Il combine les avantages des wikis sémantiques et ceux des wikis pair-à-pair.
   - Nous avons défini les opérations d'édition du nouveau type de données répliqué et leurs intentions.
   - Pour assurer la réception des opérations par tous les pairs, nous avons proposé d'utiliser l'algorithme Lpbcast [EGH+03] pour propager rapidement les modifications à tous les pairs connectés et un mécanisme d'anti-entropie [DGH+88] pour supporter le mode déconnecté.
   - Nous avons modifié Woot [WUM07] un algorithme de réplication optimiste pour synchroniser le nouveau type de données tout en préservant l'intention des opérations et la convergence des pages et des annotations des pairs.
   - De plus, nous avons développé un mécanisme d'annulation de groupe dans Swooki qui permet d'annuler toute modification.
   - Nous avons montré que Swooki assure le modèle de cohérence CCI sur les pages wikis contenant des annotations sémantiques et des entrepôts sémantiques des pairs.

   Ainsi Swooki adresse bien les motivations relatives au passage à l'échelle et à la tolérance aux pannes. Par contre, Swooki ne permet pas de représenter des procédés.
2. Pour supporter l'édition multi-synchrone et les procédés, nous avons proposé Distributed Semantic MediaWiki (DSMW) une instanciation du modèle de réplication optimiste pour les wikis sémantiques.
   - Nous avons défini le modèle de données de DSMW sous forme d'une ontologie qui permet d'exploiter l'historique au moyen des requêtes sémantiques.

- Nous avons développé un modèle Publier/Souscrire basé sur des flux pour la propagation des changements entre les pairs. La propagation est faite sous le contrôle des utilisateurs, ce qui permet de créer un réseau ami-à-ami sémantique. En effet, les utilisateurs choisissent avec qui ils souhaitent collaborer, ce qui permet de modéliser des procédés d'édition collaborative et de supporter l'édition multi-synchrone.
- Nous avons montré que DSMW assure le modèle de cohérence CCI pour les pages wikis sémantiques et les entrepôts sémantiques.

Tout comme Swooki, DSMW supporte le modèle de cohérence CCI et préserve les mêmes intentions. Par contre, le modèle Publier/Souscrire de DSMW permet d'apporter l'édition multi-synchrone et l'implémentation des procédés d'édition collaborative.

4.1 Perspectives

Nous voyons plusieurs perspectives de cette thèse, à court et à moyen terme :

1. à court terme, nous voulons valider l'expression des procédés dans DSMW en l'utilisant dans le cadre de la collaboration homme-machine [BSMMN10].
2. A moyen terme, nous voulons améliorer la disponibilité des données dans DSMW. En effet, si un nœud est indisponible alors les flux qu'il héberge sont indisponibles. En utilisant un réseau P2P structuré pour stocker les flux, alors la synchonisation est possible même en cas d'indisponibilité d'un serveur.
3. Dans cette thèse, nous avons choisi de distribuer les wikis sémantiques sur un réseau P2P non structuré afin de pouvoir représenter les procédés. Il est tout de même possible et intéressant de construire un wiki sémantique distribué sur un réseau P2P structuré comme dans UniWiki [OMMD10]. Les algorithmes proposés dans cette thèse peuvent être réutilisés mais ils doivent être instanciés sur chaque nœud de la DHT. Dans ce cas, le problème à résoudre est l'exécution des requêtes sémantiques avec des données stockées sur un réseau P2P structuré.
4. Dans les éditeurs collaboratifs distribués basés sur la réplication optimiste tel que DSMW, les utilisateurs répliquent les objets partagés et alternent des phases de divergence et de convergence. La divergence peut alors être observée, mesurée et visualisée. Nous parlons de conscience de la divergence (divergence awareness). De nombreux travaux ont été menés pour mesurer et visualiser la divergence [MSMB01, MSMO02]. Cependant, les métriques obtenues sont intimement liées aux applications. Il n'est donc pas possible de réutiliser facilement les métriques existantes en dehors de leur contexte d'origine. Si nous observons attentivement les métriques de divergence, nous pouvons nous rendre compte qu'elles peuvent être construites sur la seule connaissance des histoires causales. Ces histoires causales sont capturées

par l'ontologie utilisée dans DSMW. Cette ontologie n'est que faiblement couplée à DSMW. Une de nos perspectives est donc de proposer un cadre général qui permet de calculer les métriques de divergence indépendamment des applications [Alh].

Bibliographie

[AD91] Gregory D. Abowd and Alan J. Dix. Giving undo attention. *Interacting with Computers (IWC) Journal*, 4(3) :317–342, December 1991.

[AD+06] Sören Auer, Sebestian Dietzold, , Thomas Riechert, and Thomas Riechert. Ontowiki - a tool for social, semantic collaboration. In *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, Athens, GA, USA*, pages 736–749. Springer, November 2006.

[AFK+95] L. Allen, G. Fernandez, K. Kane, D. Leblang, D. Minard, and J. Posner. ClearCase MultiSite : Supporting Geographically-Distributed Software Development. *Software Configuration Management : Scm-4 and Scm-5 Workshops : Selected Papers*, 1995.

[Alh] Nagham Alhadad. Divergence awareness in distributed collaborative systems. Master SSR 2009-2010, SCORE Team, LORIA.

[ATS04] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4) :335–371, December 2004.

[Baz] Bazaar a distributed version control system. http ://bazaar.canonical.com/en/.

[BCC+09] F. Badra, J. Cojan, A. Cordier, J. Lieber, T. Meilender, A. Mille, P. Molli, E. Nauer, A. Napoli, H. Skaf-Molli, and Y. Toussaint. Knowledge acquisition and discovery for the textual case-based cooking system wikitaaable. In *8th International Conference on Case-Based Reasoning - ICCBR 2009, Workshop Proceedings*, 2009.

[BGE+08] Michel Buffa, Fabien L. Gandon, Guillaume Ereteo, Peter Sander, and Catherine Faron. Sweetwiki : A semantic wiki. *Journal of Web Semantics*, 6(1) :84–97, 2008.

[BHI93] Sara A. Bly, Steve R. Harrison, and Susan Irwin. Media spaces : bringing people together in a video, audio, and computing environment. *Communications of the ACM*, 36(1) :28–46, 1993.

[BKvH02] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame : A generic architecture for storing and querying rdf and rdf schema. In *ISWC 2002 : First International Semantic Web Conference*, pages 54–68. Springer Verlag, May 2002.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. 2001.

[BSMMN10] Alexandre Blansché, Hala Skaf-Molli, Pascal Molli, and Amedeo Napoli. Human-machine collaboration for enriching semantic wikis using formal concept analysis. In *Submitted to SemWiki2010 - 5rd Semantic Wiki Workshop at the 7th Extended Semantic Web Conference - ESWC 2010*, Heraklion, Greece, May 2010.

[CD95] Rajiv Choudhary and Prasun Dewan. A general multi-user undo/redo model. In *Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work ECSCW'95*, pages 231–246. Kluwer Acadamic, 1995.

[CF04] Min Cai and Martin Frank. Rdfpeers : a scalable distributed rdf repository based on a structured peer-to-peer network. In *Proceedings of the 13th international conference on World Wide Web*, pages 650–657, 2004.

[CF07] M. Cart and Jean Ferrie. Asynchronous reconciliation based on operational transformation for p2p collaborative environments. In *International Conference on Collaborative Computing : Networking, Applications and Worksharing*, 2007.

[CIKN04] Paul-Alexandru Chirita, Stratos Idreos, Manolis Koubarakis, and Wolfgang Nejdl. Publish/subscribe for rdf-based p2p networks. In *The Semantic Web : Research and Applications, First European Semantic Web Symposium, ESWS 2004*, pages 182–197. LNCS 3053, 2004.

[CS01] David Chen and Chengzheng Sun. Undoing any operation in collaborative graphics editing systems. In *International Conference on Supporting Group Work GROUP'01*, pages 197–206. ACM, September 2001.

[Cun95] Ward Cunningham. Wikiwikiweb history, September 1995.

[Dar] Darcs a free, open source source code management system. http ://-darcs.net/.

[DB92] Paul Dourish and Victoria Bellotti. Awareness and coordination in shared workspaces. In *CSCW'92 : Proceedings of the 1992 ACM conference*

on Computer Supported Cooperative Work, pages 107–114, New York, NY, USA, 1992. ACM.

[DB08] Bowei Du and Eric A. Brewer. Dtwiki : a disconnection and intermittency tolerant wiki. In *WWW '08 : Proceeding of the 17th international conference on World Wide Web*, pages 945–952, New York, NY, USA, April 2008. ACM.

[DFAB97] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human-computer interaction*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[DGH+87] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *PODC'87 : Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, New York, NY, USA, 1987. ACM.

[DGH+88] Alan Demers, Dan Greene, Carl Houser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. *SIGOPS Operating Systems Review*, 22 :8–32, 1988.

[Dou95] Paul Dourish. The parting of the ways : Divergence, data management and collaborative work. In *4th European Conference on Computer Supported Cooperative Work*, pages 215–230, 1995.

[DPV05] Alain Désilets, Sébastien Paquet, and Norman G. Vinson. Are wikis usable ? In *WikiSym '05 : Proceedings of the 2005 international symposium on Wikis*, pages 3–15, New York, NY, USA, 2005. ACM.

[DSM09] Distributed semantic mediawiki (dsmw) an extension of semantic mediawiki. http ://dsmw.loria.fr, 2009.

[EGH+03] P. Th. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems*, 21(4) :341–374, November 2003.

[EGR91] Clarence A. Ellis, Simon J. Gibbs, and Gail Rein. Groupware : some issues and experiences. *Communications of the ACM*, 34(1) :39–58, January 1991.

[Euz01] Jérôme Euzenat. Construction collaborative de bases de connaissance et de documents pour la capitalisation. In Manuel Zacklad and Michel Grundstein, editors, *Ingénierie et capitalisation des connaissances*, chapter 2, pages 25–48. Hermès Science publisher, Paris (FR), 2001.

[FlR07] Extension :flaggedrevs – mediawiki. http ://www.mediawiki.org/wiki/Extension :FlaggedRevs, 2007.

[git] Git - fast version control system. http ://git.or.cz.

[GMF+02] John H. Gennari, Mark A. Musen, Ray W. Fergerson, William E. Grosso, Monica Crubzy, Henrik Eriksson, Natalya F. Noy, and Samson W. Tu. The evolution of protégé : An environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58 :89–123, 2002.

[Gra] Graphviz - graph visualization software. http ://www.graphviz.org.

[gui] Guide pratique du travail collaboratif : théories, méthodes et outils au service de la collaboration. http ://www.a-brest.net/IMG/pdf/Guide_pratique_du_travail_collaboratif.pdf.

[HPSMW07] Ian Horrocks, Peter F. Patel-Schneider, Deborah L. McGuinness, and Christopher A. Welty. OWL : a Description Logic Based Ontology Language for the Semantic Web. In *The Description Logic Handbook : Theory, Implementation, and Applications (2nd Edition)*. Cambridge University Press, 2007.

[IROM06] Abdessamad Imine, Michaël Rusinowitch, Gérald Oster, and Pascal Molli. Formal Design and Verification of Operational Transformation Algorithms for Copies Convergence. *Theoretical Computer Science : Algebraic Methodology of Software Technology*, 351(2) :167–183, feb 2006.

[Joh88] Robert Johansen. *GroupWare : Computer Support for Business Teams*. The Free Press, New York, NY, USA, 1988.

[Kan] Brent Kang. Summary hash history for optimistic replication of wikipedia. http ://isr.uncc.edu/shh.

[KK08] Aniket Kittur and Robert E. Kraut. Harnessing the wisdom of crowds in wikipedia : quality through coordination. In *CSCW '08 : Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pages 37–46, New York, NY, USA, 2008. ACM.

[KP90] Michael J. Knister and Atul Prakash. Distedit : a distributed toolkit for supporting multiple group editors. In *CSCW '90 : Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, pages 343–355, New York, NY, USA, 1990. ACM.

[KRH07] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Conjunctive queries for a tractable fragment of owl 1.1. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC'2007 + ASWC'2007, Busan, Korea*, pages 310–323, November 2007.

[KSV07] Markus Krötzsch, Sebastian Schaffert, and Denny Vrandecic. Reasoning in semantic wikis. In *Proceedings of the 3rd Reasoning Web Summer School, Dresden, Germany*, volume 4636 of *LNCS*, pages 310–329. Springer, September 2007.

[KVV06] Markus Krötzsch, Denny Vrandecic, and Max Völkel. Semantic mediawiki. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 935–942, 2006.

[KVV+07] Markus Krötzsch, Denny Vrandecic, Max Völkel, Heiko Haller, and Rudi Studer. Semantic wikipedia. *Journal of Web Semantics*, 5(4) :251–261, 2007.

[KWK03] Brent Byunghoon Kang, R. Wilensky, and J. Kubiatowicz. The hash history approach for reconciling mutual inconsistency. *23rd International Conference on Distributed Computing Systems*, pages 670–677, May 2003.

[Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7) :558–565, July 1978.

[LCL04] Paul B. Lowry, Aaron Curtis, and Michelle R. Lowry. Building a taxonomy and nomenclature of collaborative writing to improve interdisciplinary research and practice. *Journal of Business Communication*, 41(1) :66–99, January 2004.

[LLSG92] Rivka Ladin, Barbara Liskov, Liuba Shrira, and Sanjay Ghemawat. Providing high availability using lazy replication. *ACM Transactions on Computer Systems*, 10(4) :360–391, November 1992.

[LN02] Yann Laurillau and Laurence Nigay. Clover architecture for groupware. In *CSCW '02 : Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 236–245, New York, NY, USA, 2002. ACM.

[LPS09] Mihai Letia, Nuno M. Preguiça, and Marc Shapiro. Crdts : Consistency without concurrency control. *CoRR*, abs/0907.0929, 2009.

[Mer] Mercurial a fast lightweight source control management system. http ://-mercurial.selenic.com/wiki/.

[MLS08] Patrick Mukherjee, Christof Leng, and Andy Schurr. Piki - a peer-to-peer based wiki engine. In *P2P08 : Eighth International Conference on Peer-to-Peer Computing*, pages 185–186. IEEE, 2008.

[MM02] Petar Maymounkov and David Mazières. Kademlia : A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, chapter 5, pages 53–65. Berlin, Heidelberg, October 2002.

[Mor07] J.C. Morris. DistriWiki : : a distributed peer-to-peer wiki network. *Proceedings of the 2007 international symposium on Wikis*, pages 69–74, 2007.

[MSMB01] Pascal Molli, Hala Skaf-Molli, and Christophe Bouthier. State treemap : an awareness widget for multi-synchronous groupware. In *7th International Workshop on Groupware - CRIWG'2001*, Darmstadt, Germany, September 2001.

[MSMO02] Pascal Molli, Hala Skaf-Molli, and Gérald Oster. Divergence awareness for virtual team through the web. In *Integrated Design and Process Technology, IDPT 2002*, Pasadena, CA, USA, June 2002. Society for Desing and Process Science.

[MSMOJ02] Pascal Molli, Hala Skaf-Molli, Gérald Oster, and Sébastien Jourdain. Sams : Synchronous, asynchronous, multi-synchronous environments. In *The Seventh International Conference on CSCW in Design*, 2002.

[MTEBG07] Christian Morbidoni, Giovanni Tummarello, Orri Erling, and Reto Bachmann-Gmür. Rdfsync : efficient remote synchronization of rdf models. In *Proceedings of the ISWC/ASWC2007*, volume 4825, pages 533–546, November 2007.

[Mye86] Eugene W. Myers. An o(ND) difference algorithm and its variations. *Algorithmica*, 1(2) :251–266, March 1986.

[MYS05] James R. Miller, Serhan Yengulalp, and Patrick L. Sterner. A framework for collaborative control of applications. In *SAC '05 : Proceedings of the 2005 ACM symposium on Applied computing*, pages 1244–1249, New York, NY, USA, 2005. ACM.

[NMN09] Antonio De Nicola, Michele Missikoff, and Roberto Navigli. A software engineering approach to ontology building. *Information Systems*, 34(2) :258–275, 2009.

[NR04] Sylvie Noël and Jean-Marc Robert. Empirical study on collaborative writing : What do co-authors do, use, and like ? *Computer Supported Cooperative Work*, 13(1) :63–89, 2004.

[NWQ+02] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. Edutella : a p2p networking infrastructure based on rdf. In *WWW '02 : Proceedings of the 11th international conference on World Wide Web*, pages 604–615, New York, NY, USA, 2002. ACM.

[OMMD10] Gérald Oster, Rubén Mondéjar, Pascal Molli, and Sergiu Dumitriu. Building a collaborative peer-to-peer wiki system on a structured overlay. 2010.

[O'R05] Tim O'Reilly. What is web 2.0. design patterns and business models for the next generation of software. http ://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, September 2005. Stand 12.5.2009.

[OUMI06] Gérald Oster, Pascal Urso, Pascal Molli, and Abdessamad Imine. Data Consistency for P2P Collaborative Editing. In *Conference on Computer-Supported Cooperative Work CSCW'06*. ACM Press, 2006.

[OWL] Owl web ontology language. http ://www.w3.org/TR/owl-features.

[pin] Downtime in 2007 for the 20 most popular websites. http ://uptime.pingdom.com/site/month_summary/site_name/en.wikipedia.org.

[PMSL09] Nuno Preguiça, Joan Manuel Marquès, Marc Shapiro, and Mihai Letia. A commutative replicated data type for cooperative editing. In *International Conference on Distributed Computing Systems ICDCS'09*, pages 395–403, Montréal, Canada, June 2009.

[PST04] T. Prante, N. A. Streitz, and P. Tandler. Roomware : computers disappear and interaction evolves. *Computer*, 37(12) :47–54, 2004.

[RDF] Rdf vocabulary description language 1.0 : Rdf schema. http ://www.w3.org/TR/rdf-schema/.

[RDF04] Resource description framework (rdf). http ://www.w3.org/RDF, 2004.

[RE91] Gail L. Rein and Clarence A. Ellis. ribis : A real-time group hypertext system. *International Journal of Man-Machine Studies*, 34(3) :349–367, 1991.

[RG96] Mark Roseman and Saul Greenberg. Building real-time groupware with groupkit, a groupware toolkit. *ACM Transactions on Computer-Human Interaction*, 3(1) :66–106, 1996.

[RSS] Rss 2.0 at harvard law. http ://cyber.law.harvard.edu/rss/rss.html.

[Sal95] Daniel Salber. *De l'interaction individuelle aux systèmes multi-utilisateurs. L'exemple de la Communication Homme-Homme-Médiatisée*. PhD thesis, 1995. Thèse de doctorat Informatique préparée au Laboratoire de Génie Informatique (IMAG), Université Joseph Fourier 303 pages.

[SBBK08] Sebastian Schaffert, François Bry, Joachim Baumeister, and Malte Kiesel. Semantic wikis. *IEEE Software*, 25(4) :8–11, 2008.

[sca] Scalaris distributed transactional key-value store. http ://-code.google.com/p/scalaris.

[Sch06] Sebastian Schaffert. Ikewiki : A semantic wiki for collaborative knowledge management. In *WETICE'06 : Proceedings of the 15th IEEE International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises*, pages 388–396. IEEE Computer Society, 2006.

[SE98] Chengzheng Sun and Clarence A. Ellis. Operational transformation in real-time group editors : Issues, algorithms, and achievements. In *Proceedings of Computer Supported Cooperative Work CSCW'98*, pages 59–68, New York, New York, États-Unis, 1998. ACM Press.

[Sem] http ://semantic-mediawiki.org/wiki.

[sem06] *Reusing Ontological Background Knowledge in Semantic Wikis*, volume 206 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[SJZ+98] Chengzheng Sun, Xiaohua Jia, Yanchun Zhang, Yun Yang, and David Chen. Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems. *ACM Transactions on Computer-Human Interaction*, 5(1), 1998.

[SJZY97] Chengzheng Sun, Xiahua Jia, Yanchun Zhang, and Yun Yang. A generic operation transformation scheme for consistency maintenance in real-time cooperative editing systems. In *In Proceedings of International ACM SIGGROUP Conference on Supporting Group Work*, pages 425–434. Press, 1997.

[SL98] H. Spencer and D. Lawrence. *Managing Usenet*. O'Reilly Sebastopol, 1998.

[SMIRM07] Hala Skaf-Molli, Claudia-Lavinia Ignat, Charbel Rahhal, and Pascal Molli. New work modes for collaborative writing. In *Enterprise Information Systems and Web Technologies*, pages 176–182, 2007.

[SMRM09] Hala Skaf-Molli, Charbel Rahhal, and Pascal Molli. Peer-to-peer semantic wiki. In *DEXA'09 : 20th International Conference on Database and Expert Systems Applications*, Lecture Notes in Computer Science. Springer, 2009.

[spa07] Sparql query language for rdf. http ://www.w3.org/TR/rdf-sparql-query, 2007.

[SS05a] Yasushi Saito and Marc Shapiro. Optimistic Replication. *ACM Computing Surveys*, 37(1) :42–81, 2005.

[SS05b] Yasushi Saito and Marc Shapiro. Optimistic replication. *ACM Computing Surveys*, 37(1) :42–81, 2005.

[SS05c] Steffen Staab and Heiner Stuckenschmidt, editors. *Semantic Web And Peer-to-peer*. Springer, 2005.

[Sun00] Chengzheng Sun. Undo any operation at any time in group editors. In *CSCW '00 : Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 191–200, New York, NY, USA, 2000. ACM.

[Sun02] Chengzheng Sun. Undo as concurrent inverse in group editors. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 9(4) :309–361, December 2002.

[TMGS97] S. G. Tammaro, J. N. Mosier, N. C. Goodwin, and G. Spitz. Collaborative writing is hard to support : A field study of collaborative writing. *Comput. Supported Coop. Work*, 6(1) :19–51, 1997.

[TMP+04] Giovanni Tummarello, Christian Morbidoni, Joackin Petersson, Paolo Puliti, and Francesco Piazza. Rdfgrowth, a p2p annotation exchange algorithm for scalable semantic web applications. In *The First International Workshop on Peer-to-Peer Knowledge Management P2PKM'04*, CEUR Workshop Proceedings. CEUR-WS.org, August 2004.

[uri] Naming and adressing : Uris, urls, ... http ://www.w3.org/Addressing.

[VCFS00] Nicolas Vidot, Michelle Cart, Jean Ferrié, and Maher Suleiman. Copies convergence in a distributed real-time collaborative environment. In *In Proceedings of the ACM Conference on Computer Supported Cooperative Work CSCW'00*, pages 171–180, 2000.

[Vid02] Nicolas Vidot. *Convergence des Copies dans les Environnements Collaboratifs Répartis*. PhD thesis, Université de Montpellier, September 2002. Jury : Jean Ferrié, Rachid Guerraoui, Michelle Cart, Alain Jean Marie, Pascal Molli, Marc Shapiro, Philippe Pucheral.

[VKV+06] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic Wikipedia. *Proceedings of the 15th international conference on World Wide Web*, pages 585–594, May 2006.

[VPST05] Denny Vrandecic, H. Sofia Pinto, York Sure, and Christoph Tempich. The diligent knowledge processes. *Journal of Knowledge Management*, 9(5) :85–96, October 2005.

[WUM07] Stéphane Weiss, Pascal Urso, and Pascal Molli. Wooki : a p2p wiki-based collaborative writing tool. In *Web Information Systems Engineering WISE'07*, Lecture Notes in Computer Science, Nancy, France, 2007.

[WUM08] Stéphane Weiss, Pascal Urso, and Pascal Molli. An undo framework for p2p collaborative editing. In *Collaborative Computing : Networking, Applications and Worksharing, 4th International Conference, CollaborateCom'2008, Orlando, Florida, USA*, pages 529–544, Orlando, USA, November 2008. Springer.

[WUM09a] Stéphane Weiss, Pascal Urso, and Pascal Molli. Logoot-Undo : Distributed Collaborative Editing System on P2P Networks. *IEEE Transactions on Parallel and Distributed Systems TPDS'09*, 21 :1162–1174, December 2009.

[WUM09b] Stéphane Weiss, Pascal Urso, and Pascal Molli. Logoot : a scalable optimistic replication algorithm for collaborative editing on p2p networks. In *International Conference on Distributed Computing Systems ICDCS'09*. IEEE Computer Society, 2009.

Résumé

Le Web 2.0 a montré l'importance des systèmes collaboratifs. Il a permis de transformer des internautes en écrivains du Web. De nombreux outils sociaux tels que les wikis et les blogs sont utilisés par des larges communautés produisant une grande quantité d'informations. Afin d'améliorer la recherche, la navigation et l'interopérabilité, les outils sociaux évoluent en intégrant les technologies du Web sémantique. Les wikis en tant qu'outils sociaux suivent cette tendance et évoluent vers des wikis sémantiques. L'introduction des technologies du Web sémantique dans les outils sociaux pose un certain nombre de problèmes tels que le passage à l'échelle et le support des procédés d'édition collaborative pour garantir la qualité des contenus.

Dans ce manuscrit, nous adressons ces problèmes par une instanciation du modèle de réplication optimiste dans le cadre des wikis sémantiques, une instanciation répliquant un nouveau type de données composé de pages wikis et d'annotations sémantiques. En effet, le modèle de réplication optimiste repose sur la réplication des objets partagés sur un ensemble de sites et assure leur cohérence. Nous nous plaçons dans un contexte où le nombre de sites n'est pas connu et varie à l'exécution à l'instar des réseaux pair-à-pair. En tant que système collaboratif, un système de réplication optimiste est correct s'il respecte le modèle de cohérence CCI (Causalité, Convergence, Intention). Nous proposons deux approches Swooki et DSMW, deux instanciations du modèle de réplication optimiste pour les wikis sémantiques assurant la cohérence CCI sur le nouveau type de données répliqué (les pages wikis sémantiques et les entrepôts des annotations). Swooki est le premier wiki sémantique sur réseau pair-à-pair, il combine les avantages des wikis sémantiques et ceux des wikis P2P. Swooki adresse particulièrement les problèmes de passage à l'échelle et de la tolérance aux pannes. Tandis que DSMW (Distributed Semantic MediaWiki) permet de construire un réseau ami-à-ami sémantique et s'intéresse plus aux problèmes de représentation des procédés. Dans DSMW, nous proposons un modèle de collaboration Publier/souscrire qui permet aux utilisateurs de contrôler la propagation des changements et la synchronisation des données répliquées en se basant sur la notion de flux. Par conséquent, DSMW permet de supporter l'édition multi-synchrone et l'implémentation des procédés, ce qui n'était pas possible dans Swooki. Nous proposons également une ontologie DSMW afin de modéliser sémantiquement les changements et leurs partages et de les exploiter au moyen des requêtes sémantiques.

Abstract

The Web 2.0 has shown the importance of collaborative systems. It has allowed to transform a community of strangers into a community of collaborators. Nowadays, many social tools such as wikis and blogs are used by large communities, they produce a large amount of information. The Semantic Web aims structuring this information in order to make it processable by machines, thus enabling more precise searches and more efficient navigation. Many social tools evolve by integrating the Semantic Web technologies. Wikis as social tools follow this trend and evolve towards semantic wikis. The introduction of Semantic Web technologies in social tools raises a number of problems such as scalability and support for collaborative editing processes which is necessary to maintain the quality of the content.

In this manuscript, we address these problems through an instantiation of the optimistic replication model in the context of semantic wikis, an instantiation replicating a new data type composed of wiki pages and semantic annotations. Indeed, the optimistic replication model is based on the replication of shared objects on multiple sites and ensures their consistency. In our context, the number of sites is unknown and varies like in peer-to-peer (P2P) networks. A collaborative editing system using the optimistic replication is correct if it ensures the consistency model CCI (Causality, Convergence, Intention). We propose two approaches Swooki and DSMW, two instantiations of the optimistic replication model for semantic wikis. Both approaches ensure the CCI consistency of the new replicated data type (semantic wiki pages and semantic stores). Swooki is the first peer-to-peer semantic wiki ; it combines the advantages of semantic wikis and those of P2P wikis. Swooki addresses specifically the problems of scalability and fault tolerance. While DSMW (Distributed Semantic MediaWiki) allows to build a semantic friend-to-friend network and to support collaborative editing processes. In DSMW, we propose a model of collaboration Publish/subscribe based on feeds. It allows users to control the propagation of the changes and the synchronization of the replicated data. Consequently, DSMW can support multi-synchronous editing and the implementation of processes which was not possible in Swooki. We propose also an ontology to model semantically the changes, their share and to exploit them through semantic queries.