MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique

Département de Mathématiques et informatique

Filière : Informatique

MÉMOIRE DE FIN D'ÉTUDES

Pour l'Obtention du Diplôme de Master en Informatique

Option : **Ingénierie des Systèmes d'Information**

Présenté par :

OUALI Yousef

ZAIDI Houcine

THÈME :

Edition collaborative des données liées (Linked Data)

Devant le jury composé de :

ABDALLAH BENSALLOUA	MCA	Université de Mostaganem	Président
BENHAMED S.	MAA	Université de Mostaganem	Examinateur
MECHAOUI M.D	MCA	Université de Mostaganem	Encadreur

Année Universitaire 2021-2022

Résumé

Le Web sémantique est devenu de plus en plus attrayante on permettant aux données sémantiques d'être produites en mode en ligne et d'être disponibles pour un large public. Linked Data (LD) ou bien les données liées sont utilisées pour remplacer des collections de données RDF hors ligne, le but de LD est de permettre aux personnes de partager des données structurées sur le Web aussi facilement qu'elles peuvent partager des documents ordinaires.

L'objectif de ce projet est d'étudié les approches existantes d'édition collaborative des données sémantique, de faire une collaborative comparaison entre ces approches, et de proposer un éditeur collaboratif distribué pour les données sémantiques capable de supporter les groupes dynamiques où les utilisateurs peuvent joindre et quitter à tout moment.

Mots-clés: **RDF, web sémantique, édition collaborative.**

Abstract

The Semantic Web has become increasingly attractive by allowing semantic data to be produced in online mode and to be available to a wide audience. Linked Data (LD) or Linked Data is used to replace offline RDF data collections; the purpose of LD is to enable people to share structured data on the Web as easily as they can share ordinary documents.

The objective of this project is to study existing approaches to collaborative editing of semantic data, to make a collaborative comparison between these approaches, and to propose a distributed collaborative editor for semantic data capable of supporting dynamic groups where users can join and leave at any time.

Keywords: RDF, Semantic web, collaborative editing

الملخص

أصبح الويب الدلالي جذابًا بشكل متزايد من خلال السماح للبيانات الدلالية بأن يتم إنتاجها في وضع الاتصال بالإنترنت وأن تكون متاحة لجمهور عريض. يتم استخدام البيانات المرتبطة (LD) أو البيانات المرتبطة لاستبدال مجموعات بيانات RDF غير المتصلة؛ الغرض من البيانات المرتبطة هو تمكين الأشخاص من مشاركة البيانات المنظمة على الويب بنفس سهولة مشاركة المستندات العادية.

الهدف من هذا المشروع هو دراسة الأساليب الحالية للتحرير التعاوني للبيانات الدلالية، لإجراء مقارنة تعاونية بين هذه الأساليب، واقتراح محرر تعاوني موزع للبيانات الدلالية قادر على دعم المجموعات الديناميكية حيث يمكن للمستخدمين الانضمام والمغادرة في أي وقت.

الكلمات الرئيسية. التحرير التعاوني، الويب الدلالي.

Table des matières

Remerciement

En tout premier lieu, je remercie mon Dieu, tout puissant, de m'avoir donné la force pour survivre, ainsi que l'audace pour dépasser toutes les difficultés.

Je tiens à exprimer mes vifs remerciements envers mon encadreur **Dr. Mechaoui Moulay Driss** pour sa disponibilité, son encadrement, sa confiance et les conseils qu'elle m'a généreusement prodigués.

J'exprime toute ma reconnaissance à Monsieur ABDALLAH BENSALLOUA pour avoir bien voulu accepter de présider le jury de ce mémoire.

Que Madame BENHAMED trouve ici l'expression de mes vifs remerciements pour avoir bien voulu juger ce travail.

Dedicas

À mes chers parents..

Ma famille

Mes amies

Liste des figures

Liste des tableaux

Liste des abréviations

Abréviation	Expression Complète	Page
RDF	Resource Description Framework	14
CCI	Causalité, Convergence et Intention	20
OR-Set	Observed Remove Set	22
SU-Set	Sparql Update-Set	26
B-Set	Boolean-Set	28
P2P	Peer-to-Peer	10

1. Introduction Générale

Dans le Web sémantique et dans le Web en général, un problème fondamental est la comparaison et l'appariement des données et la capacité de résoudre la multiplicité des références de données aux mêmes objets du monde réel, en définissant les correspondances entre les données sous forme de liens de données. La tâche de couplage de données devient de plus en plus importante à mesure que le nombre de données structurées et semi structurées disponibles sur le Web augmente. La transformation du Web d'un « Web de documents » en un « Web de données », ainsi que la disponibilité d'importantes collections de données générées par des capteurs (Internet des objets), mènent à une nouvelle génération d'applications Web fondées sur l'intégration des données et des services. Parallèlement, de nouvelles données sont publiées chaque jour à partir de contenus générés par les utilisateurs et de sites Web publics mènent à une nouvelle génération d'applications Web.

L'objectif de ce rapport est d'étudier les approches existantes en matière d'édition collaborative des données sémantiques, de faire une comparaison collaborative entre ces approches, et de proposer un éditeur collaboratif distribué pour les données sémantiques capables de supporter des groupes dynamiques où les utilisateurs peuvent se joindre et partir à tout moment [1].

Problématique

Les systèmes d'édition collaborative répartis se caractérisent par des échanges d'opérations entre sites, une forte concurrence et une coordination décentralisée. Dans ce contexte, ils doivent garantir la réactivité locale, le passage à l'échelle et la cohérence [2].

Les échanges d'opérations entre sites constituent une caractéristique fondamentale pour une collaboration en temps réel [3]. Cependant l'exécution des opérations dans un ordre différent, sur des copies différentes a pour conséquence une divergence de ces copies.

Donc développer un système efficace pour étendre la transition L'édition collaborative de triplets RDF (Resource Description Framework) sur les réseaux P2P devient un problème

majeur dans de nombreux environnements Web sémantique, l'idée principale étant de répliquer les données RDF sur le stockage local pour que tous les pairs puissent collaborer. La propriété clé est de savoir comment assurer la cohérence des répliques effectuant des opérations simultanées, c'est-à-dire à la fin de la boucle Dans la vie publiée, toutes les reproductions de volets distribués sont identiques. C'est un processus très difficile à mettre en œuvre car de nombreuses requêtes peuvent entrer en conflit lors de l'édition, de la synchronisation et de la modification simultanée. Pour illustrer la nature et l'étendue de certains des défis posés par le problème d'édition collaborative du stockage sémantique avec des structures intégrées.

Exemple de problème de convergence

Considérerons trois utilisateurs sur trois pairs différents qui travaillent simultanément pour éditer un store RDF. Le store RDF est décrit comme un ensemble de triplets, où chaque triplet contient trois composants <sujet, prédicat, objet>.

Les utilisateurs travaillent sur leurs pairs étant appelés : Peer-1, Peer-2 et Peer-3 avec leurs activités séquentielles. Les trois pairs sont interconnectés par un réseau informatique. Initialement, chaque pair possède une copie du store sémantique partagé qui est considéré comme vide.

Le premier utilisateur effectue deux opérations sur sa réplique locale, Op1=insT(T1) pour insérer le triplet T1 et Op2= delT(T1) pour supprimer le triplet ainsi inséré, ces opérations sont ensuite propagées aux Peer-2 et Peer-3 pour y être exécutées. Une fois reçues par le pair Peer-2, Op1 et Op2 sont tout de suite exécutées séquentiellement. Après cela, le deuxième utilisateur met à jour sa copie locale du store RDF en exécutant l’opération Op3 =insT (T1), qui insère un nouveau triplet T1, cette opération est publiée immédiatement. Dans peer-3, lorsque Op1 est exécutée, le troisième utilisateur supprime le triplet T1 en effectuant Op4=delT(T1) avant l’exécution de Op2. Durant cette étape, Op4 est envoyée au Peer-2 pour s’intégrer à sa réplique. Ensuite, Op3 et Op4 sont réexécutées respectivement sur Peer-2 et Peer-3.

À la fin de l’exécution des modifications concurrentes, les répliques du store de données sémantiques divergent. C'est parce que les opérations d’insertion et de suppression du même triplet ne commutent pas, ainsi, les intentions des opérations d'insertion et de suppression sont violées, la convergence ne serait donc pas assurée.

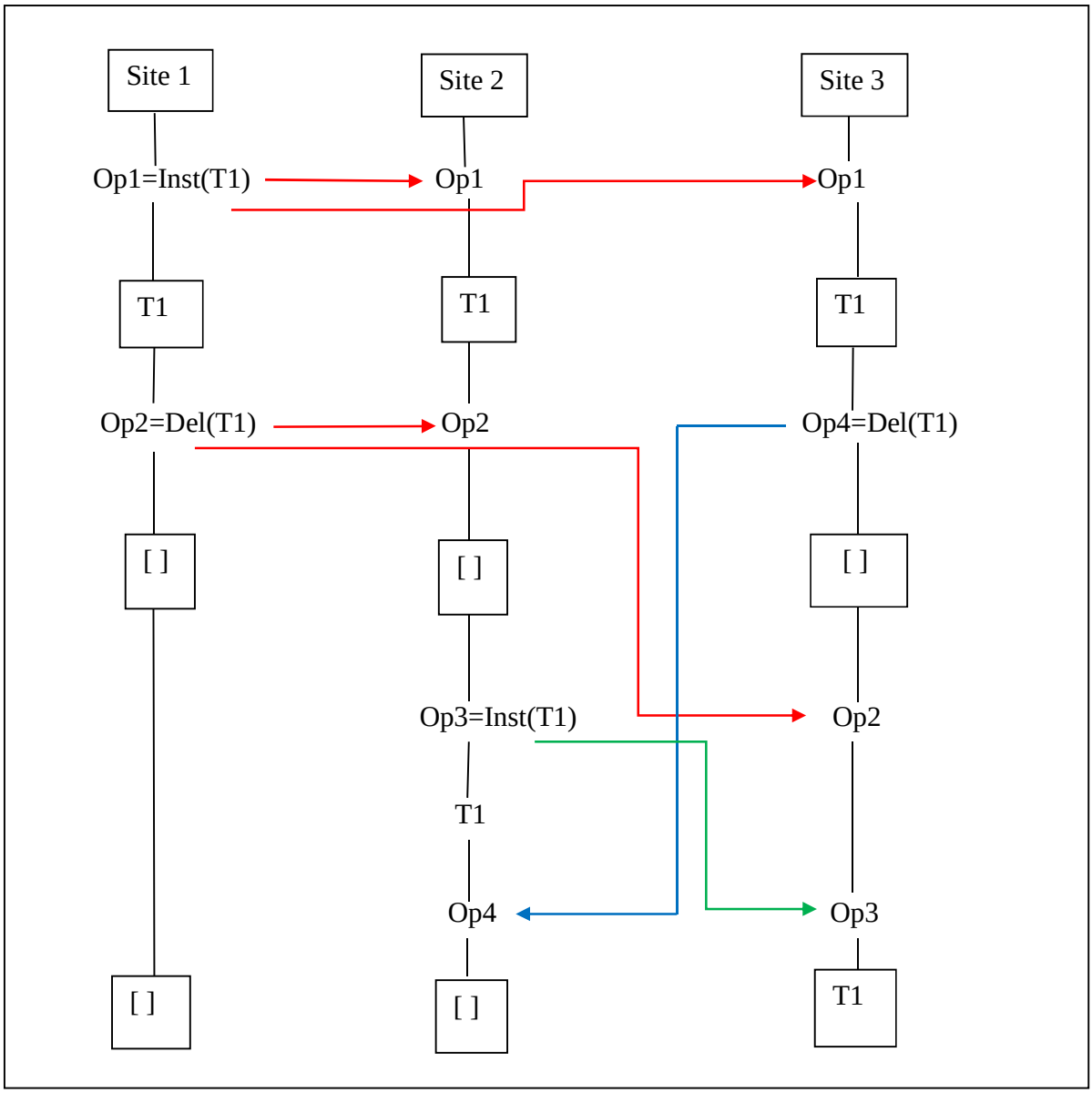


Figure 1 : Violation de la convergence après exécutions des opérations concurrentes.

2. Chapitre 1

Web et données sémantique

2.1 Introduction

Avec le Web 2.0, l'édition collaborative est énorme Nous vivons une nouvelle démocratie sur Internet et de transition vers la société de l'information, qui a conduit à un processus de mondialisation, qui a incité les gens à se connecter les uns aux autres et à créer des équipes virtuelles pour atteindre un objectif commun. L'émergence de la dénomination A, décrite comme « une attitude plutôt qu'une technologie » est appelée web 2.0. Ce dernier se concentre sur une nouvelle forme d'interaction qui place l'utilisateur au centre d'internet et qui est plus sociale et collaborative. L'objectif principal de Web Sémantiques est d'ajouter une logique au Web actuel, c'est-à-dire d'exprimer la signification des données, les propriétés des objets et les relations complexes qui existent entre eux à travers une série de règles formelles. Machine. L'accessibilité machine doit être comprise comme représentant des informations de telle sorte que, quelle que soit la forme sous laquelle les informations sont présentées, les requêtes peuvent être effectuées en fonction de la signification (c'est-à-dire de la sémantique) des données [22]. Dans ce chapitre, nous nous intéressons plus précisément à l'édition collaborative des Documents RDF.

2.2 Web et données sémantique

Le Web sémantique Pour bien comprendre, nous commençons par une donation. Le grand dictionnaire terminologique définit le terme sémantique comme << l'ensemble des relations entre les caractères, ou groupes de caractères, et leurs significations, indépendamment de la façon de les employer ou de les interpréter>>. Il précise par la suite que << si, en linguistique, la sémantique porte sur l'étude du sens à partir de la combinaison des mots, en intelligence artificielle, elle porte sur la capacité d'un réseau [le Web] à représenter de la manière la plus humaine possible des relations entre des objets, des idées ou des situations>>. Le terme

sémantique implique donc que la machine ne se contentera plus de présenter visuellement les données du Web, mais, en les reliant, elle pourra conserver les significations qui leur sont attribuables. Or, en transformant le contenu du Web pour qu'il soit <<compréhensible>> par la machine et non seulement présentable, nous permettons à cette même machine d'être plus efficace dans le traitement de l'information. Ainsi, le dialogue avec les moteurs de recherche devient possible. Nous sommes alors en mesure de nous exprimer dans des termes que nos ordinateurs peuvent aussi interpréter et échanger. Il est également possible d'automatiser, d'intégrer et de réutiliser l'information entre diverses applications Le Web sémantique est décrit généralement comme un Web destiné aux machines. Disposer d'un Web dont le contenu est abordable par les machines peut apporter de grands bénéfices : L'automatisation de nombreuses tâches fondées sur le contenu comme la recherche de ressources ayant un contenu particulier, la comparaison du contenu de ressources (pages, bases de données, ontologies, etc.). Le Web sémantique permettrait de résoudre la relative difficulté de trouver de l'information sur le web [4]. Le Web sémantique a pour objectif de transformer le World Wide Web actuel, entièrement tourné vers la présentation des documents, vers un Web dont le contenu serait compréhensible par les machines. La vision s'appuie sur l'utilisation d'ontologies. Le Web sémantique n'est pas un Web séparé mais une extension du Web actuel, dans lequel l'information est bien définie, permettant ainsi aux ordinateurs et aux personnes de travailler en coopération. Le Web sémantique permettra aux machines de comprendre les documents et les données sémantiques, et non la parole humaine et les écrits [2].

2.2.1 RDF

RDF est un acronyme de Resource Description Framework. RDF est une norme W3C des technologies Web sémantiques et la base de l'architecture standardisée des technologies Web Sémantique. Le RDF est un langage simple pour exprimer des modèles de données sous forme d’objets ressources et de leurs relations. Il sera utilisé pour annoter des documents écrits dans des langages non structurés, ou comme une interface pour des documents écrits dans des langages ayant une sémantique équivalente (des bases de données, par exemple). Un document RDF est un ensemble de triplets de la forme. Les éléments de ces triplets peuvent être des URIs (Universal Resource Identifier), des littéraux ou des variables. Cet ensemble de triplets peut être représenté de façon naturelle par un graphe, où les éléments apparaissant comme sujet ou objet

sont les sommets, et chaque triplet est représenté par un arc dont l'origine est son sujet et la destination son objet [2].

RDF est un standard décrivant : - des ressources, une ressource pouvant être n'importe quoi (personnes, lieux, animaux, documents, concepts, etc.) ; - la description de ces ressources par des attributs et des relations ; - le Framework contenant un modèle de données, des langages et des syntaxes.

2.2.2 RDFS

Le schéma RDFS donne véritablement sa sémantique à la description RDF [3]. Il permet aussi de définir un vocabulaire pouvant être utilisé pour décrire des ressources. Adaptés chacun à un domaine ou à une application spécifique. Notons que les vocabulaires, appelés aussi schémas de description, sont eux-mêmes écrits en RDF, en utilisant des balises de l'espace de nom RDFS (ex., rdfs : Class, rdfs : subClassOf, rdfs : Domain, rdfs : range,...) écrits en RDF. Deux de vocabulaire sont définis pour ce schéma. #Personne, #Chercheur, #Doctorant sont des classes de ressources d'un annuaire universitaire, #Chercheur et #Doctorant sont les sous classes de #Personne. #nom et #email sont des propriétés applicables aux ressources de la classe #Personne pour donner le nom et l'adresse email de chacune. #sous-direction est une propriété d'association entre #Doctorant et #Chercheur.

```
<?xml version="1.0"?>

 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-nsit"
          xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdfs :Class rdf:ID="Personne"/>

<rdf:Property rdf:ID="nom"›>

<rdfs:domain rdf:resource="#Personne"/>

<rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>

 </rdf:Property>

 <rdf:Property rdf:ID="email">

 <rdfs:domain rdf:resource="#Personne"/>

<rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>

</rdf:Property>

<rdfs:Class rdf:ID="Doctorant"› <rdfs:subelassOf rdf:resource="#Personne"/>

 </rdfs:Class>

<rdfs:Class rdf:ID="Chercheue>

 <rdfs:subelassOf rdf:resource="#Personne"/>

 </rdfs:Class>

<rdf:Property rdf:ID="sousDirection"›>

<rdfs:domain rdf:resource="#Doctorant"/>

<rdfs:range rdf:resource="#Chercheur"/>

</rdf:Property>

</rdf:RDF>
```

Figure 2 : Un exemple d'un schéma RDFS

2.2.3 LINKED DATA (DONNÉES LIÉES)

Le Web est de plus en plus considéré comme un espace d'information global constitué non seulement de documents liés, mais aussi de données liées. Plus qu'une simple vision, le Web of Data qui en résulte est né de la maturation de la pile de technologies du Web sémantique et de la publication d'un nombre croissant d'ensembles de données selon les principes des Données Liées [4].

2.2.4 LINKED OPEN DATA

Linked Open Data définit une vision des données liées et accessibles globalement sur l'internet, basée sur les normes RDF du web sémantique. LOD est souvent considéré comme un nuage de données virtuel où chacun peut accéder à toutes les données qu'il est autorisé à voir et peut également ajouter à toutes les données sans perturber la source de données originale. Cela fournit un environnement ouvert où les données peuvent être créées, connectées et consommées à l'échelle de l'internet. Une théorie de base de LOD est que les données ont plus de valeur si elles peuvent être connectées à d'autres données. Dans ce contexte, les données sont toute information structurée basée sur le web. Le LOD a été proposé comme base pour un gouvernement ouvert et pour résoudre de nombreux problèmes d'intégration de données.

- **Avantages LOD :**

La mise en relation des ensembles de données ouvertes favorise la créativité et l'innovation, car tous les développeurs, les citoyens et les entreprises peuvent utiliser tous ces ensembles de données pour mettre les choses en contexte et créer des connaissances et des applications. Certains des avantages des données ouvertes liées sont :

- Utilisation efficace des ressources : Les données ouvertes liées réduisent la redondance en s'appuyant sur le travail des autres plutôt que de reproduire les systèmes existants.
- Amélioration de la qualité des informations : Les données ouvertes liées encouragent la normalisation des métadonnées et des formats de données, ce qui rend les données plus ables et crédibles.
- Création de valeur ajoutée : En se connectant directement à d'autres données, les données ouvertes liées permettent aux utilisateurs de découvrir, d'utiliser et de réutiliser les informations de manière nouvelle et inattendue.

- Identification des lacunes dans les informations : Les données ouvertes liées permettent de mettre en évidence et de corriger les erreurs de données.
- Amélioration de la transparence : Les données ouvertes liées créent les moyens pour les citoyens et les groupes de défense de demander des comptes au secteur privé et aux gouvernements.

2.3 Conclusion

Nous avons donné une vue globale sur le web et données sémantique, et Nous avons présenté le RDF (Resource Description Framework) et sa structure, Nous avons aussi déterminé quelque concept important de données sémantique tel que Linked Data et Linked Open Data.

3. Chapitre 2

Edition des données sémantiques

3.1. Introduction

La réconciliation des données divergentes est un problème qui se pose toujours dans les systèmes distribués, De nombreuses solutions basées sur la technique de réplication ont été proposées afin de garantir la cohérence des données partagées. Cependant, les solutions existantes ne sont pas adéquates aux environnements distribués puisqu'elles sont soit centralisées, soit ne tiennent pas compte des comportements dynamiques des sites [23]. Les systèmes d'édition collaborative permettent à plusieurs utilisateurs de visualiser et d'éditer la même copie en même temps à partir de plusieurs sites reliés par réseaux de communication. Le maintien de la cohérence est l'un des défis les plus importants dans la conception et la mise en œuvre de systèmes d'édition collaboratives en temps réel [24].
Dans ce chapitre, nous allons présenter quelques approches d'édition collaborative de Type de Donnée Répliquée Commutatif CRDT.

3.2. Edition des données sémantiques

Le développement fulgurant du Web 2.0 est à l'origine d'une nouvelle génération des éditeurs appelés 'éditeurs collaboratifs' passant de la centralisation à la décentralisation et de l'individu à la communauté. Depuis son introduction, le type de données répliqué commutatif (CRDT) a été largement étudié et continue d'être l'objet de nombreux travaux de recherche.

3.2.1. CRDT (Type de donnée répliqué commutatif)

Un type de donnes pour lequel l'exécution des opérations concurrentes est commutative. On peut facilement montrer qu'un système basé sur un CRDT de données qui respecte la causalité est assuré de converger. Efficace, Selon les conditions de précédence, seules les opérations causales ne commutent pas. Cependant, ces opérations doivent être effectuées dans le même

ordre Toutes les répliques. Donc, si le système fournit une causalité, le type de données CRDT la convergence [5].

CRDT est une structure de données qui peut être répliquée sur plusieurs ordinateurs d'un réseau, où les répliques peuvent être mises à jour indépendamment et simultanément sans coordination entre les répliques, et où il est toujours mathématiquement possible de résoudre les incohérences qui pourraient survenir. Les CRDT ont également été utilisés dans les plateformes de distributions et les systèmes de chat en ligne [6].

Les algorithmes basés sur le modèle CRDT ont été initialement conçus pour la collaboration P2P en mode asynchrone, mais évoluent actuellement vers un mode de collaboration en temps réel. Un système basé sur la structure de données CRDT dont les opérations concurrentes est commutative donc CRDT garantie la convergence des systèmes.

L'idée consiste à associer un identifiant unique à chaque emplacement partagez des symboles, des lignes ou des atomes des documents. Si l'opération est générée, l'identifiant unique est également associé au paramètre positionnel. Cependant, la gestion des identifiants est un problème très sérieux car, la précision de cette méthode repose sur l'unicité et la préservation de l'identifiant au changement dans l'ordre général des opérations. Par conséquent, la plage de valeurs de l'identifiant doit être choisie pour être compacte. Ainsi, entre les deux identifiants, il doit toujours être possible de créer un nouvel identifiant. Parmi les algorithmes de CRDT existants, nous distinguons : OR-Set , C-SET ,SU-Set , B-SET, sr-CE et LD-Set Nous présentons ci-après chacun de ces algorithmes.

3.2.2. Le modèle de cohérence CCI

Le modèle le plus utilise pour l’édition collaborative est le modèle appelé CCI [7], un éditeur collaboratif doit respecter le modèle CCI, c’est-à-dire assurer la Convergence, la Causalité et l’Intention.

- **Respect de la causalité**

Le terme « causalité » désigne une relation de précédence entre deux évènements ou plus. Dans une édition collaborative, on parle de causalité lorsqu’il existe une relation de précédence entre les différentes opérations. Le respect de la causalité signifie que l’ordre des opérations reçues peut être maintenu, on dit que les opérations sont reçues dans un ordre causal.

Afin d'assurer une relation de précédence causale entre n'importe quel pair d'opérations (OP1, OP2), on doit assurer que OP1 doit toujours s'exécuter avant OP2 sur tous les sites. Supposons les deux opérations OP1 et OP2 générées respectivement sur les deux répliques A et B.
L'opération OP1 précède causalement l'opération OP2 (noté OP1 -> OP2) dans seulement l'une des situations suivantes : soit A et B deux répliques,

• Si A et B sont identiques (le même site) et l'opération OP1 a été générée avant l'opération OP2.

• Si A et B ne sont pas identiques (deux sites différents) et l'opération OP1 a été exécutée sur B avant que l'opération OP2 soit généré sur ce site.

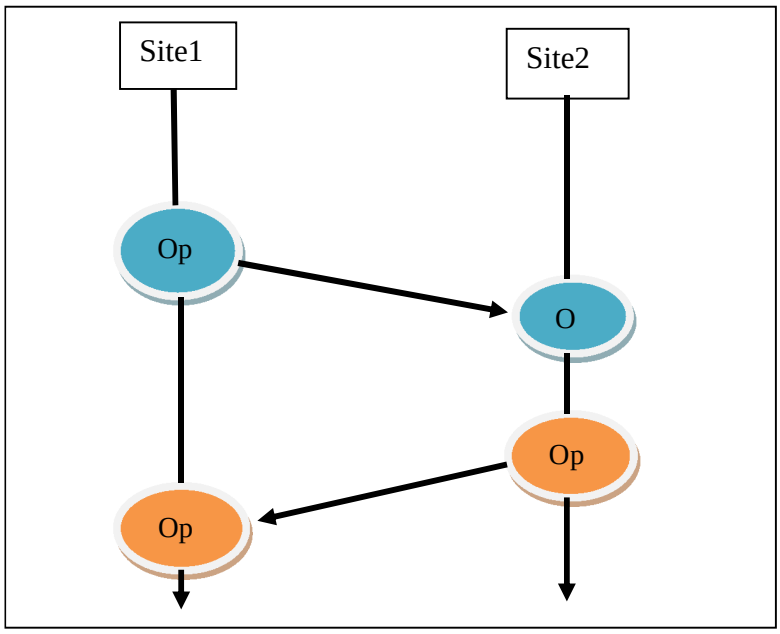


Figure 3 : Principe de Causalité

Pour expliquer cet ordre d'apparition des opérations, on suppose que l'opération OP2 dépend des effets de l'opération OP1. Dans ce cas, si le respect de la causalité est préservé, les mises à-jour seront exécutées dans le même ordre sur tous les sites du réseau et par conséquent les répliques vont finir par avoir la même valeur.

- **Convergence**

Dans un système distribué avec réplication des données sur chaque site, un algorithme distribué est convergent si toutes les répliques sont identiques sur chaque site après exécution de toutes les opérations considérées. Dans le cas contraire, on parlera de divergence.
Cette propriété est difficile à assurer et fait toujours l'objet de recherches actuelles dans différents domaines [8].

- **La préservation de l'intention**

Afin de respecter les intentions, l'exécution d'une opération doit être adaptée à l'état d'exécution afin de tenir compte des opérations concurrentes. Dans le cas d'un document texte, il s'agit généralement de recalculer la position d'insertion ou de suppression. En plus de ce calcul, il peut y avoir l'insertion d'un bloc de conflit [9] ou encore, l'annulation de l'effet d'autres opérations [10]. L'intention n'a jamais été formalisée et il n'existe donc pas de méthode générique pour la préserver.

3.3. Les algorithmes d'édition de données sémantique

3.3.1. OR-Set

C'est un ensemble qui peut être modifié simultanément et finira par atteindre le même état partout il est "finalement cohérent" [21]. OR-Set ne prend en considération que les opérations d'insertion et suppression d'un seul élément, il serait impossible d'appliquer OR-Set directement sur SPARQL Update .Cela signifie qu'avant de supprimer un élément, l'opération d'ajout correspondant à cet élément doit être arrivé et doit être exécuté.
L'objectif de l'attribution d'étiquettes uniques est de garantir que les éléments stockés dans la charge utile sont toujours uniques, et donc, seulement ajouté et supprimé une fois. La convergence d'OR-Set est assurée par le réseau sous-jacent qui livre les messages dans l'ordre causal avec les identifiants uniques.
Dans OR-Set, chaque élément est associé à un ensemble de tag unique. Une opération d'insertion locale crée un tag pour l'élément et une opération locale de suppression entraîne la suppression de tous les tags de l'élément en question.

Dans l'ensemble OR-Set chaque élément est inséré et étiqueté avec un identifiant unique. Ces identifiants uniques peuvent être générés par un mécanisme pour ordonner des événements dans un système distribué.

Bien que ses créateurs n'aient pas considéré l'intention de préservation dans leur modèle de cohérence, OR-Set le préserve également. L'effet que chaque opération d'ajout ou de suppression d'une paire est unique, ce qui signifie qu'il n'y a pas de conflit entre les opérations indépendantes. En cas d'ajout et de suppression parallèles du même élément, l'insertion prévaudra. **La figure 4** montre une exécution d'OR-Set

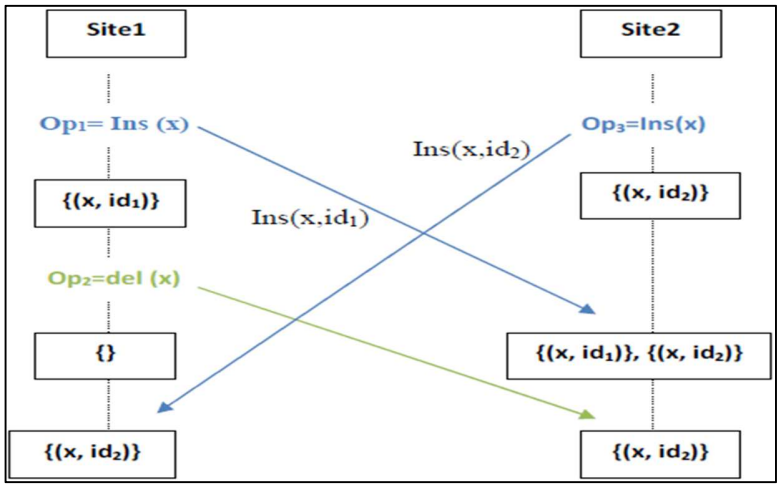


Figure 4 : Exécution de l'addition concurrente et suppression du même élément [11].

3.3.2. C-SET [12]

C-Set est un CRDT conçu pour le type de données pour les l'ensemble. Il utilise un compteur associé à chaque élément pour suivre le nombre de fois qu'il a été ajouté et supprimé. Un élément est considéré comme membre de l'ensemble C si son compteur est supérieur à zéro [18].C-Set définit un type de données CRDT pour les ensembles, ou chaque élément dans cet ensemble correspond à un triple RDF.

Il représente S d'éléments comme un ensemble de couple de (e : élément, x : Z). Il définit sur cette série quatre opérations : ins(e : élément),del(e : élément), rins(e : élément, k : Z) et rdel (e : élément, k : (Z)) [13].

L'opération ins (e : élément) peut être exécuté localement immédiatement. Il génère et envoi des rins(e : élément, k : z) d'opération d'insertion à distance qui est exécutée à distance. De la

même manière, l'opération del(e) peut s'effectuer localement, il génère et envoie l'opération de suppression distante (e : élément, k : Z*) [13].

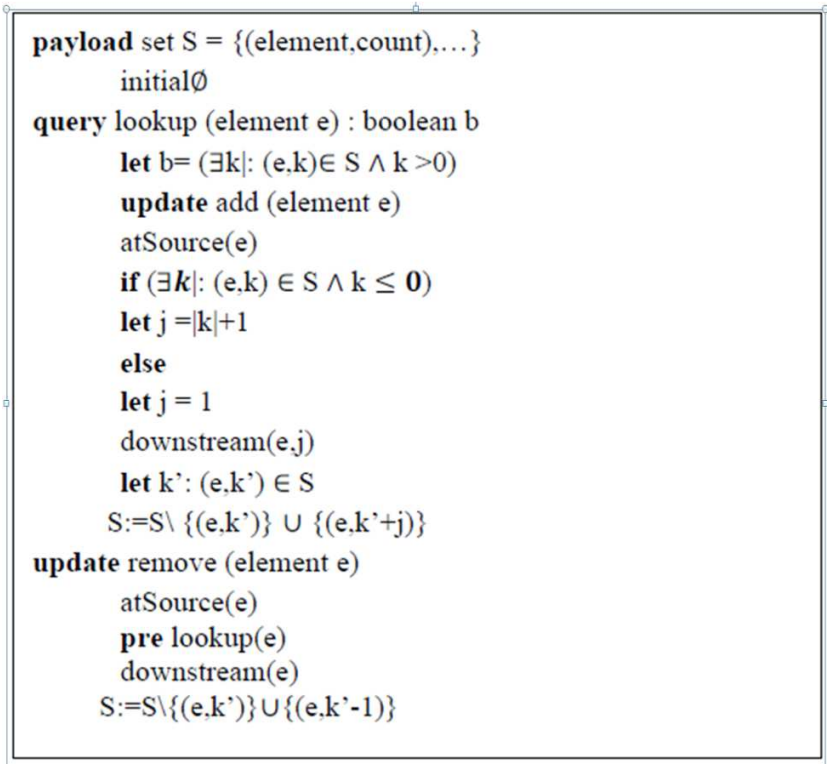


Figure 5 : Algorithme C-Set

- **Exemple :**

Cet exemple présente une exécution avec C-Set et une exécution simple sans C-Set.
La figure 6 montre un exemple d'exécution de C-Set. Et la figure 5 montre un contre-exemple pour traditionnel ensemble. Le site 1 exécute la séquence [op 1 ; op 2 ; op 3] qui en fait exécute [(+1) + (+1) + (-1)] sur le compteur associé à la valeur x dans le C-Set. Lorsque les mêmes opérations sont exécutées dans un ordre différent sur le site3, la séquence [op 1 ; op 3 ; op 2] force le calcul de [(+1) + (-1) + (+1)] sur le compteur associé à la valeur x dans le C-Set.

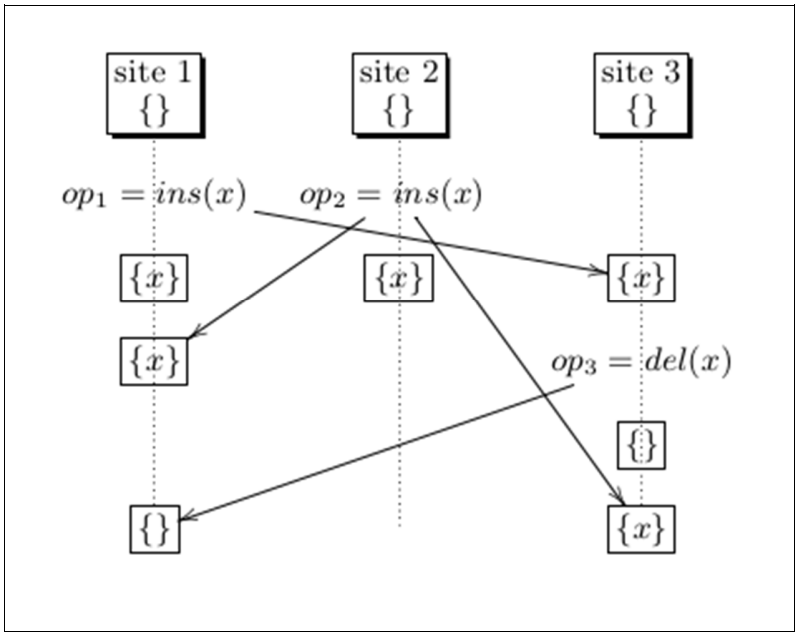


Figure 6 : Exécution avec ensemble traditionnel.

Évidemment, les deux sites convergent. Une exécution alternative est également présentée dans la figure 6. Cette exécution illustre comment l'opération op3 peut déclencher l'envoi de l'opération rdel (x, -2).

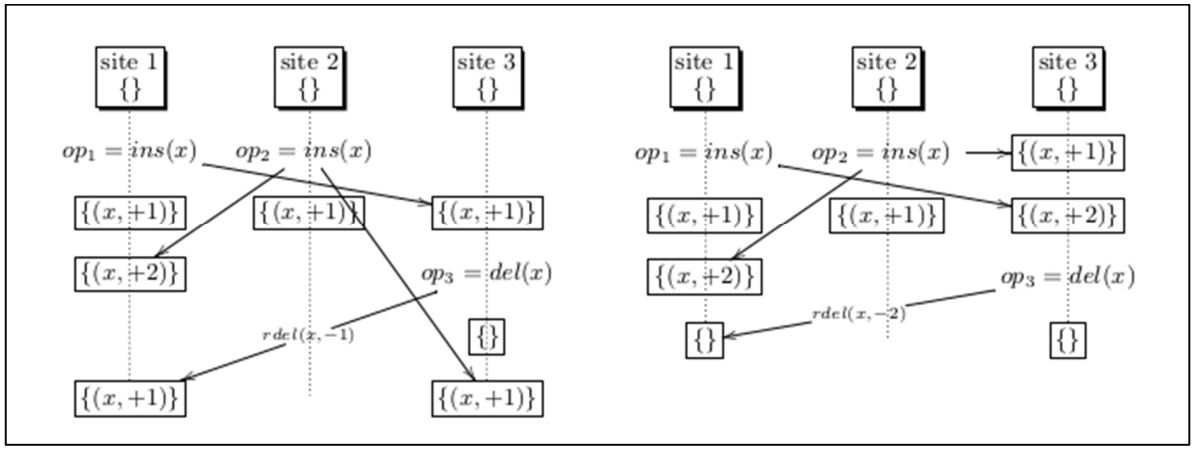


Figure 7 : Deux exécutions C-Set convergentes [14]

C-Set génère une séquence d'additions d'éléments qui appartiennent à (Z, +).
Comme l'addition dans (Z, +) est commutative, C-Set converge donc il préserve la cohérence.

3.3.3. SU-Set

SU-Set est un type de données répliquées commutative (CRDT) pour RDF graph basé sur OR-Set, il suit un modèle de réplication optimiste et supporte la norme de SPARQL Update, il permet la réplication à grande échelle des graphes RDF pour assurer une forte cohérence à terme, c'est-à-dire lorsque le système est inactif, toutes les répliques sont convergentes.

SU-set permet d'assurer la causalité, la convergence et les intentions dans une vision de données liées.

SU-Set modifie le comportement des opérations à diffuser de telles sortes que l'ensemble des triplets seront affectés un par un. SU-Set repose sur la livraison causale du réseau sous-jacent, ce qui est présente un défi et peut poser des problèmes dans les plateformes fortement dynamiques. En outre, dans SU-Set, il n'a pas été défini le mécanisme de l'intention des opérations d'édition.

```
payload set S
    initial ∅
query lookup (triple t) : boolean b
    let b = (∃u : (t,u) ∈ S)
update insert (set<triple> T)
    atSource(T)
        let α = unique()
    downstream(T, α)
        let R = {(t,α) : t ∈ T}
        S := S ∪ R
update delete (set<triple> T)
    atSource(T)
        let R = ∅
        foreach t in T:
            let Q = {(t,u) | (∃u : | (t,u) ∈ S)}
            R := R ∪ Q
    downstream(R)
        // Causal Reception
        pre All add(t,u) delivered
        S := S \ R
update delete − insert(whrPat, delPat, insPat)
    // match(m, pattern): triples matching
    //     pattern within mapping m.
    atSource(whrPat, delPat, insPat)
        let S' = {t | (∃u | : (t,u) ∈ S)}
        // M is a Multiset of mappings
        let M = eval(Select *
                        from S' where whrPat)
        let D' = ∅
        foreach m in M:
            D' = D' ∪ match(m, delPat)
        let D = {(t,u) : t ∈ D' ∧ (t,u) ∈ S}
        let I' = ∅
        foreach m in M:
            let I' = I' ∪ match(m, insPat)
        let α = unique()
    downstream(D, I', α)
        // Causal Reception
        pre All add(f,u) ∈ D delivered
        let I = {(i,α) : i ∈ I'}
        S := (S \ D) ∪ I
```

Figure 8 : SU-Set optimisé

Exemple : Une opération de suppression d'un élément implique qu'il n'est plus présent dans le document. Dans notre exemple si dessus (Figure8) lorsque le site1 exécute op=DELETE DATA {picanto est voiture.}, l'opération supprime simplement les triplets avec l'identifiant 2 sur les

deux sites, mais à la fin d'exécution, nous voyons que le triplet supprimé est toujours présent mais avec un autre identifiant ce qui montre qu'il y a un manque de préservation de l'intention.

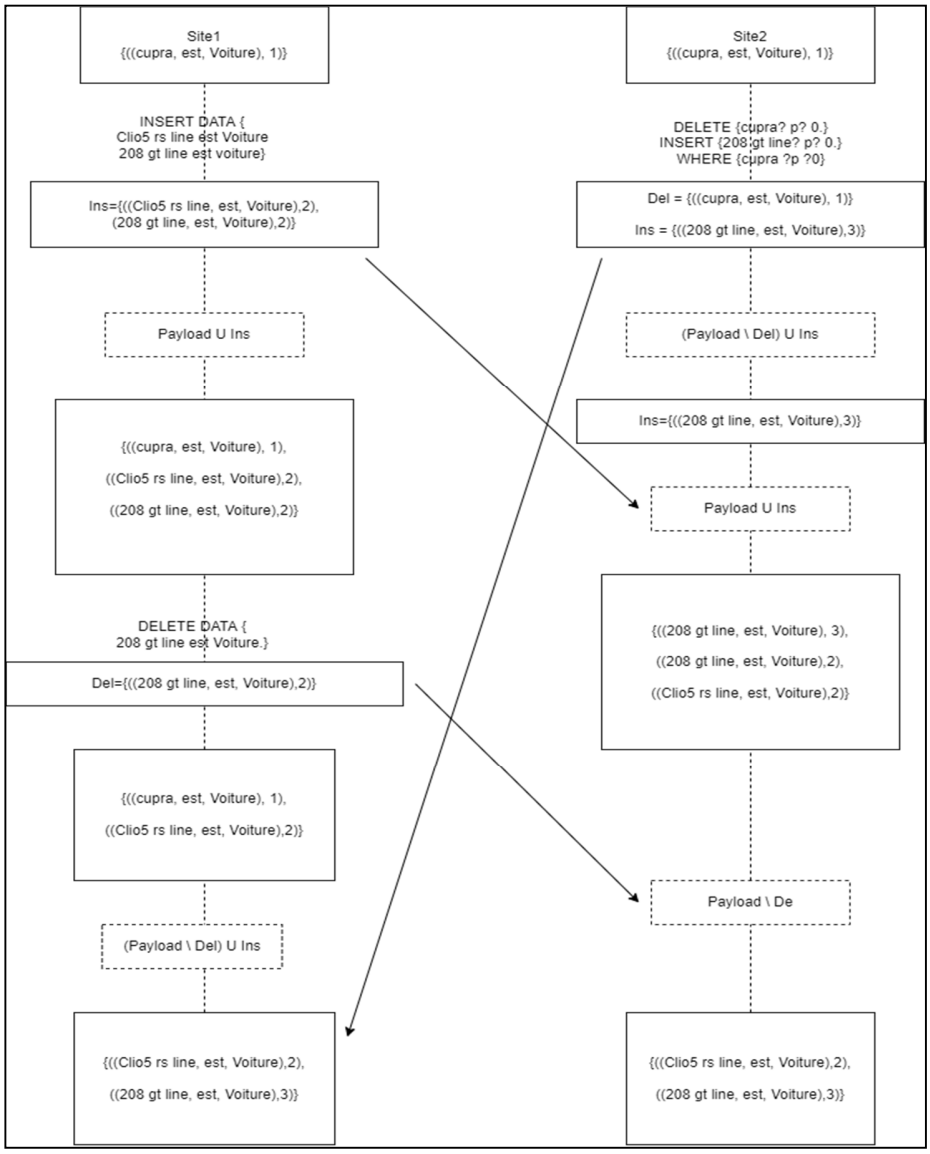


Figure 9 : SU-Set violation d'intention [15]

3.3.4. B-SET [16]

B-Set est un CRDT qui a été conçu pour soutenir l'édition collaborative évolutive de magasins sémantiques distribuées. Contrairement aux autres approches CRDT basées sur le type d'ensemble, B-Set prend en compte le modèle CCI par l'une l'introduction d'une nouvelle structure couplée à de nouvelles opérations.

Les opérations définies sur la structure de données B-Set sont les suivantes :

- L-Ins(t) : opération d'insertion locale générée et exécutée sur le même site. Elle insère un triplet t.
- R-Ins(t) : est une opération d'insertion distante générée exécutée sur un autre site. Elle insère un triplet t.
- L-Del(t) : est une opération de suppression locale générée et exécutée sur le même site. Elle supprime un triplet t.
- R-Del(t) : est une opération de suppression à distance générée et exécutée sur un autre site. Elle supprime un triplet t.

Initialement, les opérations L-Ins(t) et L-Del(t) sont exécutées localement, puis génèrent respectivement R-Ins(t) et R-Del(t) pour être exécutées à distance.

Afin de maintenir la cohérence des magasins sémantiques, chaque opération générée et exécutée sur un site doit être exécutée sur tous les autres magasins sémantiques. Cela ne nécessite que chaque opération générée localement (L-Ins (t) et L-Del (t)) doive être diffusée aux autres sites. Pour garantir que toutes les opérations simultanées des utilisateurs se succèdent, B-Set définit un ensemble de règles dans tous les cas de combinaison possibles. Ces règles décrivent les combinaisons d'algorithmes des opérations locales exécutées sur un site donné.

Considérons les axiomes suivants :

- Un triple t est retiré localement si opType=L etB=false, et il est retiré à distance si opType=Ret B=faux.
- Un triple t est ajouté localement si opType=L et B=true, et il est ajouté à distance si opType=R et B=vrai.

Dans un ensemble classique, il existe deux opérations de base add(obj:objet) et remove(obj:objet), qui sont utilisées pour construire de nouveaux ensembles à partir d'ensembles donnés. Pour tout ensemble S, la première correspond à l'union de l'ensemble S avec un objet obj, et la seconde correspond à la différence de l'ensemble S et de l'objet obj. Cependant, le type de données set traditionnel avec les deux opérations ne peut jamais fournir un CRDT correct, parce que pour un même objet, l'addition et la suppression ne commuent pas commuter comme illustré.

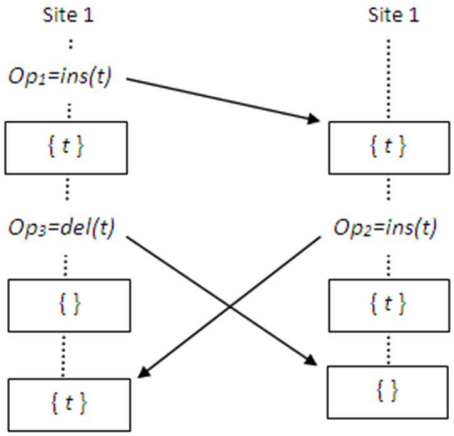


Figure 10 : Divergence d'un ensemble classique

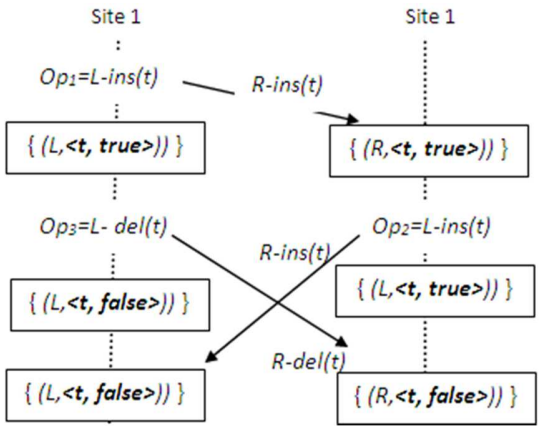


Figure 11 : Convergence avec B-Set

Cependant, B-Set a une caractéristique très intéressante qui assure une cohérence éventuelle sans aucune exigence de causalité ou de réception causale.

Une deuxième caractéristique intéressante de B-Set est que les triplets ayant une fausse étiquette comme visibilité peuvent être supprimés physiquement lorsque le système est en état de repos. Ce mécanisme facilitera d'effectuer un garbage collection.

B-Set assure une cohérence éventuelle, puisqu'une booléenne est associée à chaque triplet et la nouvelle valeur de cette variable dépend de l'ensemble des règles de l'opération de base de conjonction et de disjonction selon l'opération, donc B-Set converge [17].

3.3.5. srCE

C'est environnement collaboratif de publication collaborative P2P pour les magasins de distribution Sémantique pour assurer la cohérence des données répliquées entre tous les utilisateurs collaborateurs. Il fournit une nouvelle approche de collaboration de stockage RDF extensible. srCE n'est pas seulement adaptée au stockage distribué de triplets RDF, mais également à la prise en charge d'opérations de traitement simultanées à grande échelle. La méthode est un nouveau type de données répliquées échangeables (CRDT) qui garantit la structure définie du modèle de cohérence CCI. Le but est de garantir que les sites puissent accéder aux données RDF sans synchronisation préalable. Dans cette méthode, deux multi-Sets sont généralement utilisés : le premier contient des triplets ajoutés et le second contient des triplets supprimés. L'ensemble de résultats contient des données cohérentes calculées en calculant la multiplicité [18].

- **Multiplicité :**

Soit T un triplet RDF, S un ensemble de triplets, pour chaque t ∈ T alors f(t) est la multiplicité de t tel que f : T → N, où N est l'ensemble des nombres entiers. La valeur de f(t) est dite la multiplicité du triplet t dont la valeur correspond au nombre de fois le triplet t figure dans l'ensemble S. La multiplicité d'un élément ne peut être jamais négative.

- **RDF store :**

Un RDF store est un entrepôt utilisé pour le stockage des triplets RDF. Il est considéré comme un multi-ensemble qui se définit par un couple (T, F) où T est un ensemble de triplets et F est une fonction de multiplicité dont on associe à chaque triplet une valeur correspondant à ses occurrences.

- **RDF store d'insertion :**

Un RDF store d'insertion, noté A, est un RDF store qui contient tous les triplets insérés par un utilisateur couplés à des valeurs de multiplicité.

- **RDF store de suppression :**

RDF store de suppression, noté par D, est un RDF store qui contient tous les triplets avec leurs valeurs de multiplicités f(t), supprimés par l'utilisateur.

- **RDF store résultant :**

Un RDF store résultant, noté R, est l'ensemble résultant qui comporte tous les triplets contenus dans RDF store d'insertion tel que les valeurs de la multiplicité de ces triplets sont supérieures à celles des mêmes triplets existants dans RDF store de suppression D [30].

La **Figure 9** montre comment obtenir le RDF store résultant à partir des RDF stores ajouté et supprimé. Tous les cas possibles sont présentés dans cet exemple. Seuls les premiers et deuxièmes triplets de A apparaissent dans R car ils ont une multiplicité plus grande que les mêmes triplets dans D. Ainsi, le RDF résultant contient <houcine,nom, zaidi> et <houcine,mbox,houcine@gmail.com>.

Le premier triple n'est pas supprimé car sa multiplicité est dans le RDF ajouté alors qu'il n'existe pas dans le RDF supprimé. Pendant ce temps, le deuxième triple a une valeur de multiplicité dans le RDF ajouté qui est supérieure à celle du RDF supprimé. Ce mécanisme de construction de RDF résultant garantit la convergence et la cohérence dans tous les cas. Donc, différents utilisateurs devraient avoir le même RDF store lorsque chaque triple est ajouté ou supprimé [6].

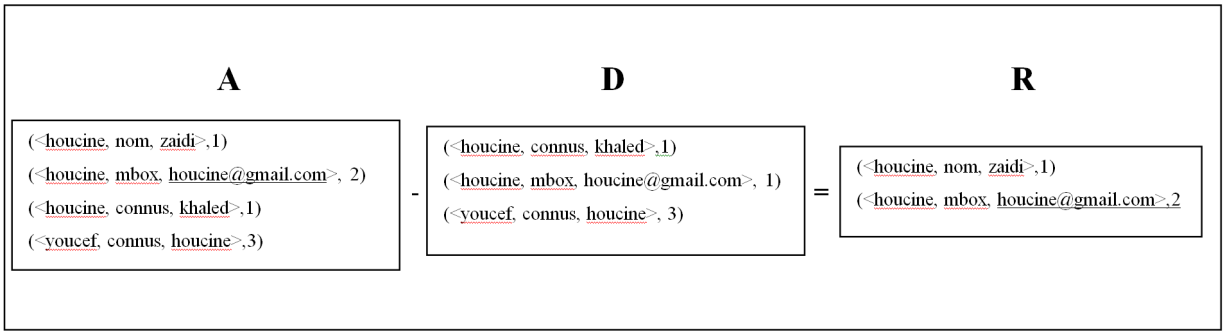


Figure 12 : Exemple de construction du RDF store résultat [15]

- **Opérations de modification :**

L'utilisateur travaille sur un RDF store par l'ajout, la suppression et la modification des Triplets du RDF store. Chaque action de mise à jour lancée par un utilisateur, est également réalisée par une opération. Dans les systèmes d'édition collaborative, deux primitives d'opérations génériques peuvent affecter un RDF store : insertion et suppression. Une opération de mise à jour peut être considérée comme une suppression de la valeur existante à mettre à jour suivie par une insertion de la nouvelle valeur. Les deux primitives d'opérations génériques utilisées lors de l'édition collaborative d'un RDF store sont :

– insT(t) : est une opération à laquelle le triplet t est inséré dans le RDF store d'insertion.

– delT(t) : est une opération à laquelle le triplet t est inséré dans le RDF store d'une suppression.

La mise à jour d'un triplet est donc équivalente à l'exécution de la séquence [delT(t1),insT(t2)]. Une requête de mise à jour consiste en deux opérations, comprenant un triplet à supprimer et un autre à ajouter. En d'autres mots, l'exécution d'une opération de modification, changeant la valeur d'un triplet t1 par une nouvelle valeur de t2, correspond à l'exécution séquentielle de delT(t1) suivi par insT(t2).

3.3.6. LD-Set

LD-set est une solution qui utilise un objectif de réplication optimiste [19], où le système se compose d'un ensemble de nœuds distribués et interconnectés, où tous les nœuds partagent une seule réplique donnée tout au long du processus pour une meilleure convivialité et de meilleures performances. L'idée clé derrière cela est qu'il existe un moyen de faire en sorte que plusieurs nœuds mettent à jour une réplique partagée. Plusieurs nœuds mettent à jour le contenu du réplica

partagé d'une manière ou d'une autre En même temps, identifiez les conflits et résolvez-les afin d'obtenir le même résultat à la fin de la session de collaboration.

3.4. Étude comparative entre ces algorithmes

srCE est une approche destinée à la réplication optimiste pour l'édition collaborative des stores sémantiques sur un réseau P2P. C-Set assure aussi la convergence mais ne mentionne pas la garantir des critères de causalité, de cohérence et de préservation de l'intention du modèle CCI. B-Set est conçu non seulement pour assurer la convergence des répliques triplets mais aussi pour préserver les intentions de l'utilisateur intégrées dans une architecture distribuée. Les ensembles d'opérations sont également définis an de permettre l'édition simultanée des mêmes mémoires triplets partagées. SU-Set permet d'assurer la causalité, la convergence et les intentions dans une vision d'un "Live" Linked Data. LD-Set se concentre sur un mécanisme de réplication optimiste pour l'édition collaborative sociale de données liées distribuées au sein d'une communauté virtuelle d'utilisateurs. Il permet de maintenir une cohérence éventuelle tout en supportant une mise à jour simultanée.

Le tableau suivant résume une comparaison entre les algorithmes présentés dans ce rapport, cette comparaison est basée sur le respect de ces algorithmes au modèle CCI, leur capacité a passé à l'échelle, et en fin sur leur compatibilité avec SAPRQL.

Critères	B-set	C-Set	SU-Set	LD-Set	Or-Set	srCE
Passage à l'échelle						X
Intention						
Causalité						
Convergence	X	X	X	X	X	X
Compatibilité SPARQL	X	X	X	X		X

Tableau 1 : Résumé de comparaison entre les algorithmes

3.5. Conclusion

Dans ce chapitre nous avons évalué les algorithmes basés sur les CRDT. Nous avons étudiés dans ce rapport l'OR-Set, C-Set, SU-Set, B-Set, srCE, LD-Set. Malgré leur robustesse elles ont leurs faiblesses. Dans le chapitre suivant, nous présentant l'algorithme LD-Set basé sur l'approche des CRDT capable de supporter l'édition collaborative dans les environnements P2P et d'assurer la cohérence des documents partagées.

4. Chapitre 03

Conception et modélisation de l'algorithme LD-Set

4.1. Introduction

LD-Set est une solution qui utilise un objectif de réplication optimiste, LD-Set permet l'édition collaborative en mode peer2peer. Comme illustrer par la figure (12) la structure d'un groupe collaboratif composé de 3 sites, chaque site possède sa propre copie des données et y applique des opérations. Le résultat attendu est que toutes les copies des sites soient les mêmes après la fin de l'exécution des opérations.

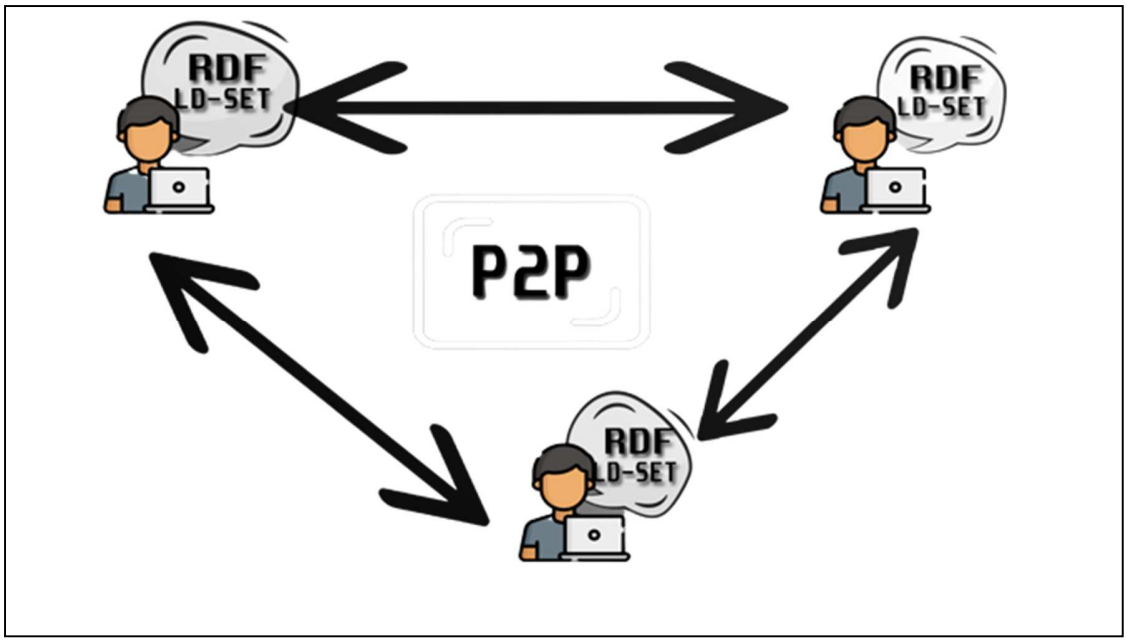


Figure 13 : Structure générale d'une édition collaborative

LD-set utilise deux opérations principales pour mettre à jour les magasins de données liées en effectuant des actions d'ajout et de suppression.

- L'opération **d'ajout (X)** ajoute un triple, donné en ligne dans la requête, dans le magasin de données liées.
- L'opération **Remove (X, n)** supprime un triple, donné en ligne dans la requête demande, du magasin de données liées n fois s'il existe déjà.

Ces opérations sont invoquées par les utilisateurs sociaux et exécutées en tant qu'opération SPARQL/UPDATE sur les magasins RDF.

La spécification du mécanisme de cohérence qui permet de résoudre le problème de la cohérence entre les répliques partageant les mêmes données liées et exécutant des modifications concurrentes dans un ordre différent. L'idée principale est d'associer à chaque triple de Linked Data à un compteur unique comme triple supplémentaire qui est initialisé à 0 et incrémenté lorsque le même triple est réinséré dans le magasin de Linked Data.

- Lors de la suppression d'un triple, un paramètre intéressant doit être déclaré pour déterminer combien de fois on doit supprimer le triple inséré.
- Payload est la structure de données définie ci-dessus. Lookup et Update sont deux méthodes qui utilisent la charge utile pour donner un résultat pour un argument et effectuer des opérations d'ajout et de suppression, respectivement.
- Add(t) insère un triple à la source et le diffuse ensuite à tous les autres nœuds, qui ajoutent l'élément reçu dans leurs charges utiles privées. De cette manière, si le triple existe déjà, son compteur est seulement incrémenté ; sinon, il est inséré avec 0 comme valeur initiale.
- Remove(t,n) supprime n fois le triple t de la réplique locale et envoie l'opération correspondante à tous les autres nœuds qui réalisent les mêmes effets dans leur charge utile.

La manière dont le LD-Set est définie, permet la commutativité entre toutes les opérations générées. Ainsi, le LD-Set est conforme à CRDT [20]. De même LD-Set garantit la cohérence éventuelle dans tous les cas.

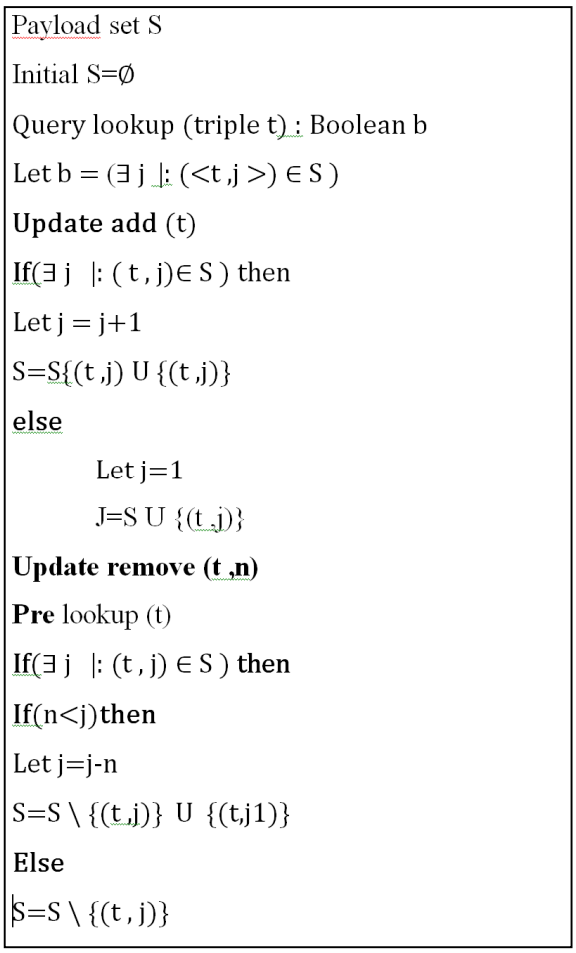


Figure 14 : Specification de LD-Set

La figure 15 montre les problèmes de divergence des données liées collaboratives après avoir exécuté des mises à jour simultanées. Lorsque deux utilisateurs dans deux pairs exécutent la même séquence d'opérations dans des ordres différents, ils obtiennent des résultats divergents. Cela est dû à la non-commutativité des opérations concurrentes d'insertion et de suppression exécutées sur le même magasin de données liées. Par contre la **figure 16** montre la cohérence éventuelle entre les deux nœuds lorsque LD-Set est utilisé.

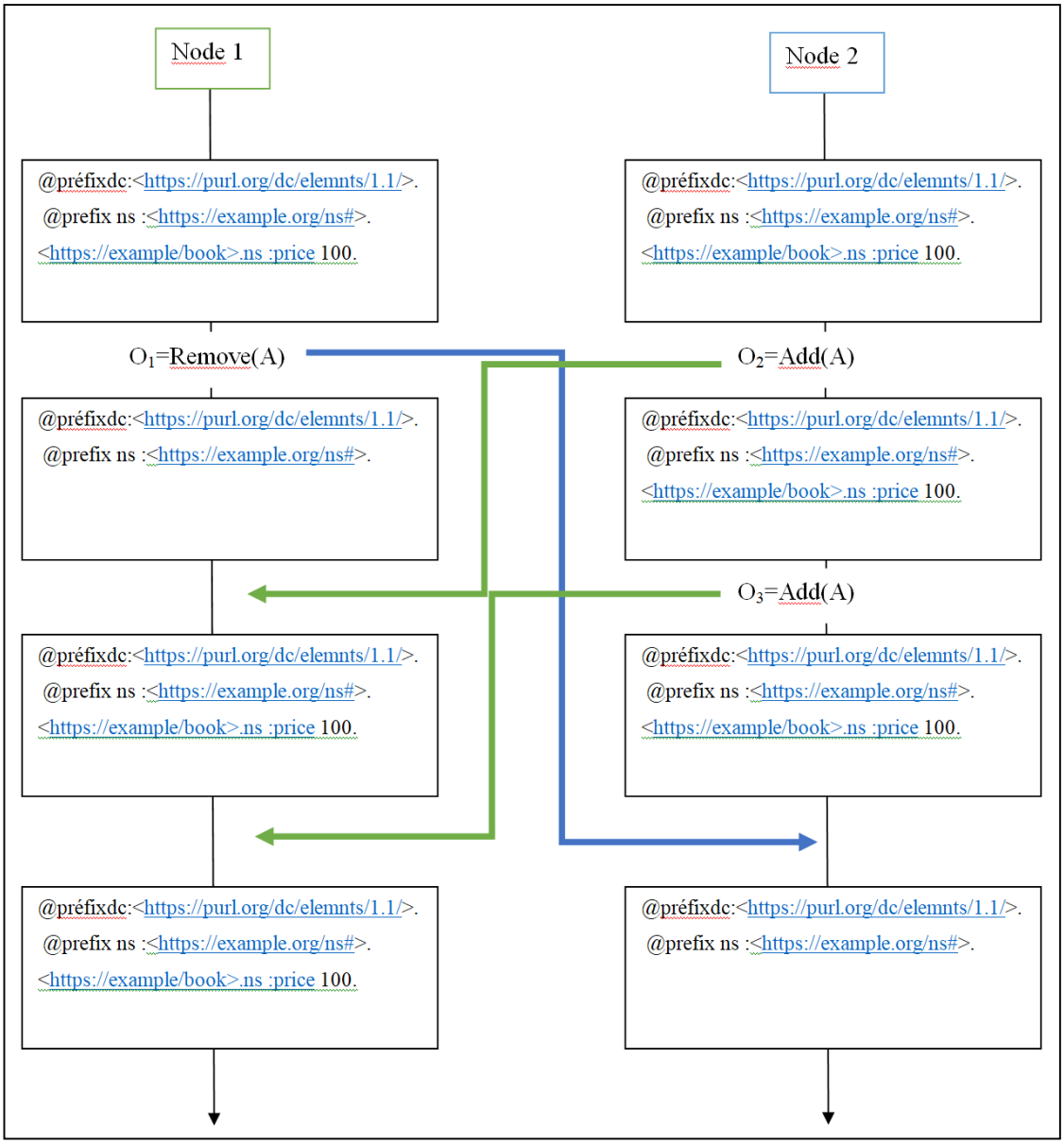


Figure 15 : Divergence des données liées collaboratives

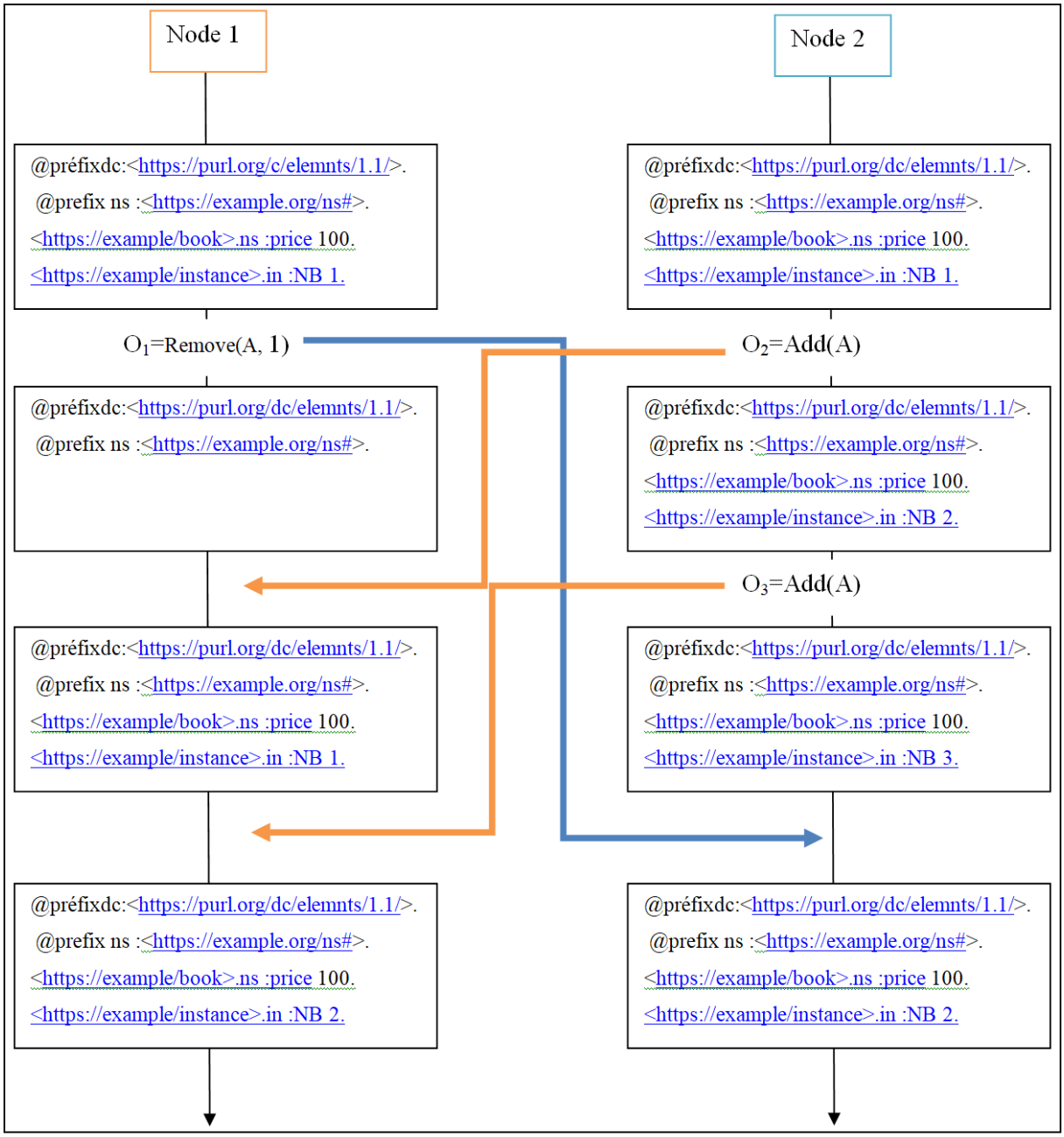


Figure 16 : Convergence après intégration de LD-Set

Les auteurs de LD-Set affirment que l'algorithme permet la commutativité entre toutes les opérations générées et garantit la cohérence éventuelle dans tous les cas. Par ailleurs, nous allons montrer dans ce qui suit des scenarios entrainant une divergence des opérations.

Puis nous présenterons notre solution proposée à ce problème.

4.2. Problème de LD-Set

Nous avons constaté que LD-Set que LD-Set à de mal à maintenir la convergence dans certaines situations de concurrence à cause de l'absence de causalité entre leurs opérations. Dans ce qui suit, nous allons présenter deux scénarios de divergences de LD-Set.

4.2.1. Premier Scenario de divergence

Dans la figure (17), nous montrons le scénario où LD-Set ne parvient pas à assurer la convergence pour les opérations d'insertion. Dans ce qui suit, Nous discuterons en détail du scénario de divergence d'insertion illustré par la figure (17).

Considérons trois utilisateurs sur trois sites différents qui travaillent simultanément pour éditer un store RDF. Les utilisateurs travaillent sur leurs sites étant appelés : Site1, Site2 et Site3 respectivement avec leurs activités séquentielles. D'abord chaque site a une copie du store sémantique partagé vide, c1=c2=c3= {}. Dans un premier temps, Site1 et Site3 appliquent les opérations d'insertion O1 =Add(T) et O3=Add(T) sur leurs copies locales. Ce qui revient à ajouter le triplet c1={(T,1)}, c3 ={(T,1)}. O1 et O2 sont diffusés vers Site2, Site3 et Site1, Site2, respectivement. O1 et O2 sont récupérés et exécutés par Site2 et sa copie locale est mise à jour, c2 = {(T,2)}. Ensuite, Site1 récupère et exécute O3, ce qui entraîne la mise à jour de la copie locale vers c1 = {(T,2)}. Ensuite, Site2 supprime deux T en exécutant l'opération O3 = Remove(T,2) résultant en un magasin sémantique vide c2= {}. O3 est diffusé et récupéré immédiatement par Site1 et Site3 après l'exécution de l'opération c1 et c3 deviennent vides. Enfin, Site3 récupère et exécute O1 de manière différée. Cela entraîne la violation de la condition de causalité. Ainsi, la convergence n'est pas assurée et les copies finales ne sont pas toutes identiques.

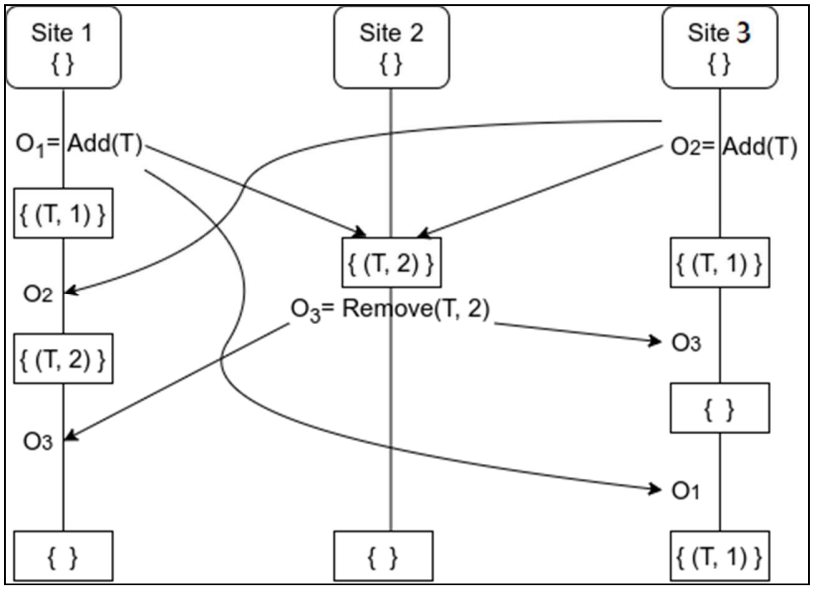


Figure 17 : Divergence des opérations d'insertion

4.2.2. Deuxième Scenario de divergence

Dans la figure (18), nous montrons les scénarios où LD-Set ne parvient pas à assurer la convergence pour les opérations de suppressions. Dans ce qui suit, Nous discuterons en détail du scénario de divergence de suppressions. Illustré par la figure (18).

Considérons trois utilisateurs sur trois sites différents qui travaillent simultanément pour éditer un store RDF. Les utilisateurs travaillent sur leurs sites étant appelés : Site1, Site2 et Site3 respectivement avec leurs activités séquentielles. D'abord chaque site a une copie du store sémantique partagé, c1=c2=c3={(T,2)}. Dans un premier temps, Site1 et Site3 appliquent les opérations de suppressions O1 = Remove(T, 2) et O3= Remove (T, 2) sur leurs copies locales. Ce qui revient à supprimer le triplet c1={(T,1)}, c3 ={(T,1)}. O1 et O2 sont diffusés vers Site2, Site3 et Site1, Site2, respectivement. O1 et O2 sont récupérés et exécutés par Site2 et sa copie locale est mise à jour, c2 = { }. Ensuite, Site1 récupère et exécute O3, ce qui entraîne la mise à jour de la copie locale vers c1 = {(T, 1)}. Ensuite, Site2 ajoute un T en exécutant l'opération O3 = Add(T) résultant en un magasin sémantique rempli c2= {(T, 1)}. O3 est diffusé et récupéré immédiatement par Site1 et Site3 après l'exécution de l'opération c1 et c2 deviennent rempli. Enfin, Site3 récupère et exécute O1 de manière différée. Cela entraîne la violation de la

condition de causalité. Ainsi, la convergence n'est pas assurée et les copies finales ne sont pas toutes identiques.

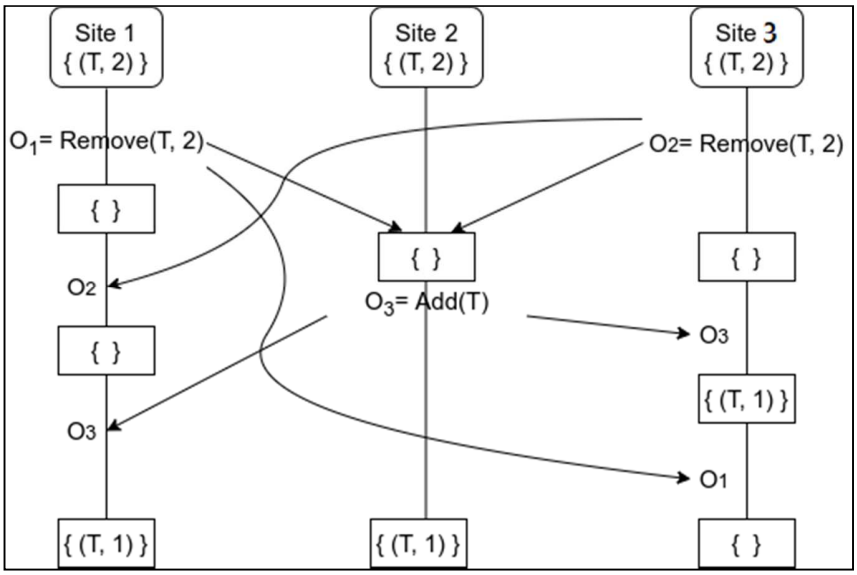


Figure 18: Divergence des opérations de suppression

Dans ce cas on a le problème de divergence on peut aussi trouver d'autres problèmes (des conflits entre les opérations, violation de l'intention…etc.) ce qui implique qu'on ne respecte pas le modèle de cohérence CCI (causalité, convergence et intention). Car, en tant qu'environnement d'édition collaborative, un environnement de réplication est correct s'il respecte le modèle de cohérence CCI. Notre objectif est de trouver une solution qui améliore l'algorithme LD-set d'une manière qui adhère au modèle de cohérence CCI.

4.3. Principe de notre solution

Pour régler le problème de causalité, nous proposons d'ajouter un Log pour l'état de document RDF. C'est-à-dire le Log va contenir toutes les opérations qui ont changé l'état du document RDF.

- S : L'état de document RDF
- Log : ensemble d'opérations qui ont modifié S.

Le principe de cette solution consiste à Hasher les triplets dans une table d'hashage comme HashSet (Log), ou la clé est le triplet (unique), et la valeur est l'ensemble des opérations qui sont appliquées sur ce triplet.

Key	Value
T1	ADD(T1), DEL(T1), ADD(T1)
T2	…..
…	…..

Tableau 2 : Exemple d'un HashSet

La table d'hachage permet de trouver un triplet dans un temps constant. Cela permet d'augmenter la performance de l'algorithme.

4.3.1. Algorithme

Dans le modèle utilisé, nous définissons la requête **q** comme un quadruplet (SiteID, k, dep, op) où **Site ID** est l'identifiant du site collaborateur (ou l'utilisateur) émetteur de la requête et **K** est le numéro d'opération générée localement, **K** est incrémenté à chaque nouvelle génération d'opération. Il est à noter que la concaténation de **Site ID** et de **K** est définie comme l'identifiant de **q**, c'est-à-dire l'identifiant de la requête **ID** = **SiteID** + **K**. La composante **dep** est l'identifiant de la requête de dépendance, et enfin **op** est l'opération à être exécutée sur l'état partagé.

Key	Value
T1	q1(ID,dep,op1), q2(ID,dep,op2),….
T2	…..
…	…..

Tableau 3 : Exemple d'un HashSet de triplet et requête.

1.1.1 Calcul de dépendance causale

Une requête peut dépendre d'autres requêtes précédentes selon l'ordre d'exécution. Le suivi de cette dépendance dans un journal permet d'identifier les requêtes qui doivent être exécutées sur tous les sites selon le même ordre.

La dépendance est calculée dans la phase de génération locale, il faut d'abord chercher le triplet qu'on veut le modifier, si le triplet n'existe pas dans le Log alors la dépendance de la requête est **null**, sinon la requête dépendra de la dernière requête qui se trouve dans la liste des requêtes de **T1** dans la partie value du Log. Les cas de dépendance d'un triplet est résumé dans les points suivants :

- Une requête **q** qui ajoute un nouveau triplet **T1**(qui n'existe pas dans Log) à l'état **S** du document RDF, ne dépend d'aucune autre requête c'est-à-dire **q.dep = null**.
- Une requête **q'** qui ajoute un triplet**T1** déjà existant dans le **Log**, dépend de la dernière requête qui à changer l'état de **S** à travers **T1**. C'est-à-dire **q'.dep =q.ID** .
- Une requête **q'** qui supprime un triplet**T1** déjà existant dans le **Log**, dépend de la dernière requête qui à insérer **T1**. C'est-à-dire **q'.dep =q.ID (avec q.op = INS(T1))**.
- Une requête **q** qui supprime un triplet**T1** inexistant dans le Log, ne dépend d'aucune autre requête c'est-à-dire **q.dep = null**.

1.1.2 Algorithme de contrôle de concurrence

Notre système d'édition collaborative des documents **RDF** est constitué d'un groupe de **N** sites (où **N** est variable dans le temps) commençant une session de collaboration avec le même état initial $\mathbf{S_0}$. Chaque site stocke toutes les opérations effectuées dans un journal **Log**. Notre procédure de contrôle de concurrence est donnée dans **l'Algorithme 1**.

A. Génération d'une requête locale

Lorsqu'une opération **op** est générée localement, elle est exécutée immédiatement sur son état de génération, à savoir **S = Apply (op, S)** qui calcule l'état résultant lors de l'exécution opération **op** sur l'état **S**. Par la suite, on incrémente le compteur des requêtes locales **(K++)**. Après avoir formé la requête **q = (SiteID, k, null, op)**, on calcul la dépendance de cette requête par grâce à la fonction **COMPUTE_DEP (q, Log)** qui retourne une nouvelle requête **q'**, cette dernière sera diffusée aux autre utilisateurs (sites) pour être intégrée.

B. Intégration d'une requête distante

Chaque site contient une file d'attente **Q** pour stocker les requêtes distantes envoyées par les autres sites. Une fois reçue, la requête est ajoutée à **Q**. Afin de préserver la causalité, une requête distante **q** est extraite de **Q** si et seulement si elle est causalement prête. C'est-à-dire que la requête dont elle dépend est déjà exécutée dans ce site. Pour les requêtes avec une dépendance **null** elles sont automatiquement causalement prêtes.

Quand une requête distante **q** est causalement prête, elle est retirée de la file d'attente **(Q ← Q – q)**, et appliquée sur l'état **S** du site par la fonction **Apply(op, S)**.

```
1: Main:
2: INITIALIZATION
3: while not aborted do
4: if there is an input o then
5: GENERATE REQUEST (o)
6: else
7: RECEIVE REQUEST
8: INTEGRATE REMOTE REQUESTS
9: end if
10: end while

11: INITIALIZATION:
12: S ← [ ]
13: Log ← [ ]
14: K ← 0
15: Q ← [ ]
16: dep ← null

17: RECEIVE REQUEST:
18: if there is a request q from a network then
19: Q ← Q + q
20: end if

21:INTEGRATE REMOTE REQUEST:
22: if there is q in Q that is causally-ready then
23: Q ← Q– q
24: S ← Apply (op, S)
25: end if
```

Algorithme 1. Algorithme de contrôle de la concurrence

```
1: GENERATE REQUEST (op):
2: S ← Apply (op, S)
3: K++
4: q ← (SiteID, K, dep, op)
5: q0 ← COMPUTE_DEP (q, Log)
6: Broadcast q0 to otherusers
```

Algorithme 2. Génération des opérations locales

```
1 : COMPUTE_DEP (q, Log): q'
2: q ← q'
3: if (q.op.T) in Log then
4: q'.dep← the identifier of the last request
applied on T.
4: end if
5: return q'
```

Algorithme 3. Calcul de dépendance causale

4.4. Conclusion

Lors de cette phase, nous avons présenté d'une manière détaillée le fonctionnement de l'algorithme LD-Set et montré le problème de sa fonctionnalité à travers des scenarios de divergence des opérations d'insertion et suppression. Dans le prochain chapitre nous présenterons notre tentative d'amélioration du LD-set pour qu'il adhère au modèle CCI.

5. Chapitre 04

Implémentation et validation

5.1. Introduction

Ce chapitre va être consacré à la partie réalisation. Pour cela nous allons présenter dans un premier lieu l'environnement matériels et logiciel de développement, par la suite, nous décrivons la phase d'implémentation en nous basant sur quelques captures.

5.2. Environnement de travail

5.2.1. Environnement matériel

Pour la réalisation de notre projet, nous avons utilisé Surface Microsoft caractérisé par :

- Système d'exploitation : Windows 10 Professionnel
- Processeur : Intel(R) Core(TM) m3-7Y30 CPU @ 1.00GHz 1.61 GHz
- Mémoire vive :4,00 Go
- Disque Dur : 115 Go.
- Type du Système : Système d'exploitation 64 bits, processeur x64.

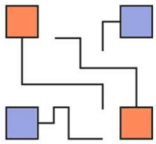
5.2.2. Environnement logiciel

Python 3 : Python est un langage de programmation généraliste interprété de haut niveau. Sa philosophie de conception met l'accent sur la lisibilité du code grâce à l'utilisation d'une indentation significative. Python est typé dynamiquement et ramassé. Il prend en charge plusieurs paradigmes de programmation, y compris la programmation structurée (en particulier

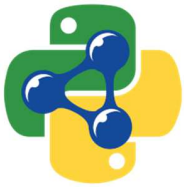
procédurale), orientée objet et fonctionnelle. Il est souvent décrit comme un langage "piles incluses" en raison de sa bibliothèque standard complète.

Pycharm IDE : PyCharm est un environnement de développement intégré utilisé dans la programmation informatique, en particulier pour le langage de programmation Python. Il est développé par la société tchèque JetBrains

Diagrams : Diagrams est une bibliothèque python qui vous permet de générer automatiquement des diagrammes.



RDFLib : RDFLib est une bibliothèque Python pour travailler avec RDF, un langage simple mais puissant pour représenter l'information. Cette bibliothèque contient des analyseurs/sérialisés pour presque toutes les sérialisations RDF connues, telles que RDF/XML, Turtle, N-Triplets et JSON-LD, dont beaucoup sont désormais pris en charge dans leur forme mise à jour.



Tkinter: Tkinter est une liaison Python à la boîte à outils Tk GUI. Il s'agit de l'interface Python standard de la boîte à outils Tk GUI et de l'interface graphique standard de facto de Python. Tkinter est inclus avec les installations Linux, Microsoft Windows et macOS standard de Python. Le nom Tkinter vient de l'interface Tk.

5.3. Réalisation

Dans cette section, nous allons présenter notre implémentation du l'algorithme LD-Set, nous avons aussi implémenté un validateur automatique de scenarios afin de vérifier la validité des algorithmes implémentés. Enfin, nous allons présenter l'implémentation de notre solution au problème de la causalité qui fait défaut à LD-Set.

5.3.1. Implémentation de LD-Set

Dans la **Figure 19**, nous montrons les sections de code de notre implémentation LD-Set. C'est l'implémentation du l'algorithme proposé dans [7] en python. Cette implémentation nous a permis de tester la validité de cet algorithme.

```python
class LDSet:
    def __init__(self, S):
        self.S: set = S

    def add(self, e):
        if (len(e)) == 1:
            e = e[0]
        result = list(filter(lookup(e), self.S))
        if len(result):
            _, j = result[0]
            self.S = self.S.difference({result[0]}).union({(e, j + 1)})
        else:
            self.S = self.S.union({(e, 1)})

    def remove(self, e, n):
        if (len(e)) == 1:
            e = e[0]
        result = list(filter(lambda x: x[0] == e, self.S))
        if len(result):
            _, j = result[0]
            if n < j:
                self.S = self.S.difference({result[0]}).union({(e, j - n)})
            else:
                self.S = self.S.difference({result[0]})
```

Figure 19 : Implémentation du LD-Set

4.2.2 Développement du validateur

L'idée dernière l'implémentation d'un validateur c'est de fournir un outil capable de tester aléatoirement une multitude de scénarios d'exécution d'algorithmes, et de les générer automatiquement. Notre validateur est une classe destinée à étudier la validité des algorithmes. Ce validateur sera utilisé pour vérifier si le déroulement de tous les scénarios vérifie la condition de convergence de tous les magasins (stores) de données de groupe collaboratif. La **Figure 20** montre la section du code validateur.

```
class Validator:
    def __init__(self):
        self.success = 0
        self.fail = 0
        self.successful_scenarios = []
        self.failed_scenarios = []
        self.is_valid = False
        self.n_scenarios = 0

    def add_scenario(self, scenario: Scenario):
        self.n_scenarios += 1
        if scenario.success:
            self.success += 1
            self.successful_scenarios.append(scenario)
        else:
            self.fail += 1
            self.failed_scenarios.append(scenario)

    def is_valid(self):
        return self.fail == 0 and self.success > 0

    def report(self):
        print(20 * '-')
        print(f'[!]The run contained {self.n_scenarios} scenarios')
        print(f'\t({self.success}) scenarios succeeded')
        print(f'\t({self.fail}) scenarios failed')
        print(f"[!] The algorithm{' does not work ' if self.fail else ' works '}for all the scenarios")
        print(20 * '-')
```

Figure 20 : Implémentation du validateur

5.3.2. Développement de notre solution

Nous avons implémenté en python notre solution présentée dans le chapitre précédant. Cette solution permet de régler le problème de causalité qui est considérer comme le point faible de plusieurs algorithmes tels que LD-Set, B-Set, SU-Set, et srCE.

```python
class SocialUser:
    def __init__(self, ld_set: LDSet, group: CollaborativeGroup = None):
        self.id = None
        self.group = None
        self.original_ld_set = deepcopy(ld_set)
        self.ld_set = deepcopy(ld_set)
        self.created_operations = []
        self.executed_operations = []

        if group is not None:
            self.join(group)
        self.Q = []

    def is_causally_ready(self, operation, log):
        log_ops = log[operation.element] if operation.element in log else []
        log_ops_ids = to_ids(log_ops)
        executed_ops_ids = to_ids(self.executed_operations)
        try:
            idx = log_ops_ids.index(operation.id)
            prev_op = log_ops_ids[idx - 1] if (idx - 1 > 0) else log_ops_ids[0]
        except:
            prev_op = log_ops[-1].id if len(log_ops) else None
        if prev_op is None:
            return True
        else:
            return (prev_op in executed_ops_ids) or (operation.id == prev_op)

    def generate_request(self, op_type, e, n=1):
        op_id = self.generate_operation_id()
        dep_op_id = self.compute_dep()
        operation = Operation(op_type, e, id=op_id, dep_op_id=dep_op_id, n_times=n, from_user=self.id)
        self.created_operations.append(operation)
        return operation

    def compute_dep(self):
        dep_op_id = None
        if len(self.executed_operations):
            dep_op_id = self.executed_operations[-1].id
        return dep_op_id

    def execute_operation(self, operation: Operation, log=None):
        if self.is_causally_ready(operation, log):
            self.update_remote_op_deps(operation)
            if operation.type == 'ADD':
                self.add(operation.element)
            elif operation.type == 'REMOVE':
                self.remove(operation.element, operation.n_times)
            operation.is_local = self.is_op_local(operation)
            if operation.element in log:
                if operation not in log[operation.element]:
                    log[operation.element].append(operation)
            else:
                log[operation.element] = [operation]
            return operation
        return False
```

Figure 21 : Le code de la class SocialUser

Dans la **Figure 21**, nous présentons la classe **SocialUser** qui rassemble les fonctions importantes pour notre solution, dans ce qui suit nous donnerons plus de détails :

Is_causally_ready : cette fonction renvoie True/False en fonction de la causalité de l'opération avec laquelle elle est appelée. La valeur True signifie que nous pouvons exécuter cette opération.

Generate_request : cette fonction prépare une requête/opération en définissant son identifiant, en calculant l'identifiant de l'opération de dépendance et l'utilisateur qui la crée.

Compute_dep : cette fonction calcule l'opération de dépendance d'une opération donnée.

Execute_operation : cette fonction exécute les opérations causalement prêtes d'un site et ajoute l'opération au journal.

5.4. Tests et validation de notre solution

Afin de tester la validité de notre solution et la comparer avec LD-Set original, nous avons encodé les opérations dans une liste d'opérations. L'ordre de l'opération dans la liste signifie l'ordre d'exécution dans le scénario. En changeant l'ordre des opérations on simule l'arrivée retardée des opérations.

La **Figure 22** et **Figure 23** montrent le codage sous la forme d'une séquence d'opérations, les deux séquences simulent les deux scénarios de divergence de LD-Set présentés dans le chapitre précédant.

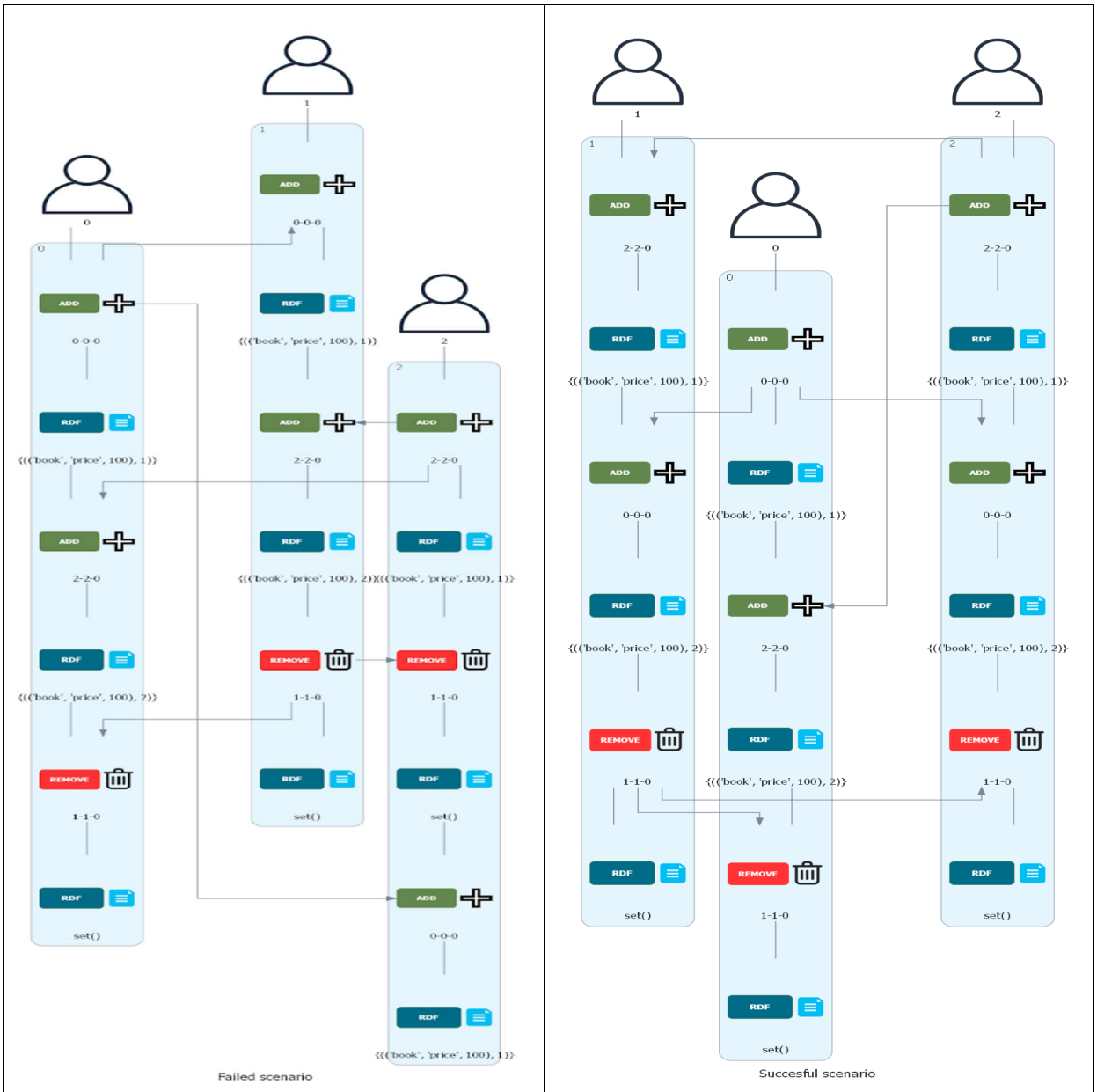
```
s1 = [
    (0, 'ADD', 1), (2, 'ADD', 1), (1, 'ADD', 1),
    (1, 'ADD', 1), (0, 'ADD', 1), (1, 'REMOVE', 2),
    (0, 'REMOVE', 2), (2, 'REMOVE', 2), (2, 'ADD', 1)
]
```

Figure 22 : Transformation du scénario 1 en séquence d'opérations

```
s2 = [
    (0, 'REMOVE', 2), (2, 'REMOVE', 2), (1, 'REMOVE', 2),
    (1, 'REMOVE', 2), (0, 'REMOVE', 2), (1, 'ADD', 1),
    (0, 'ADD', 1), (2, 'ADD', 1), (2, 'REMOVE', 2)
]
```

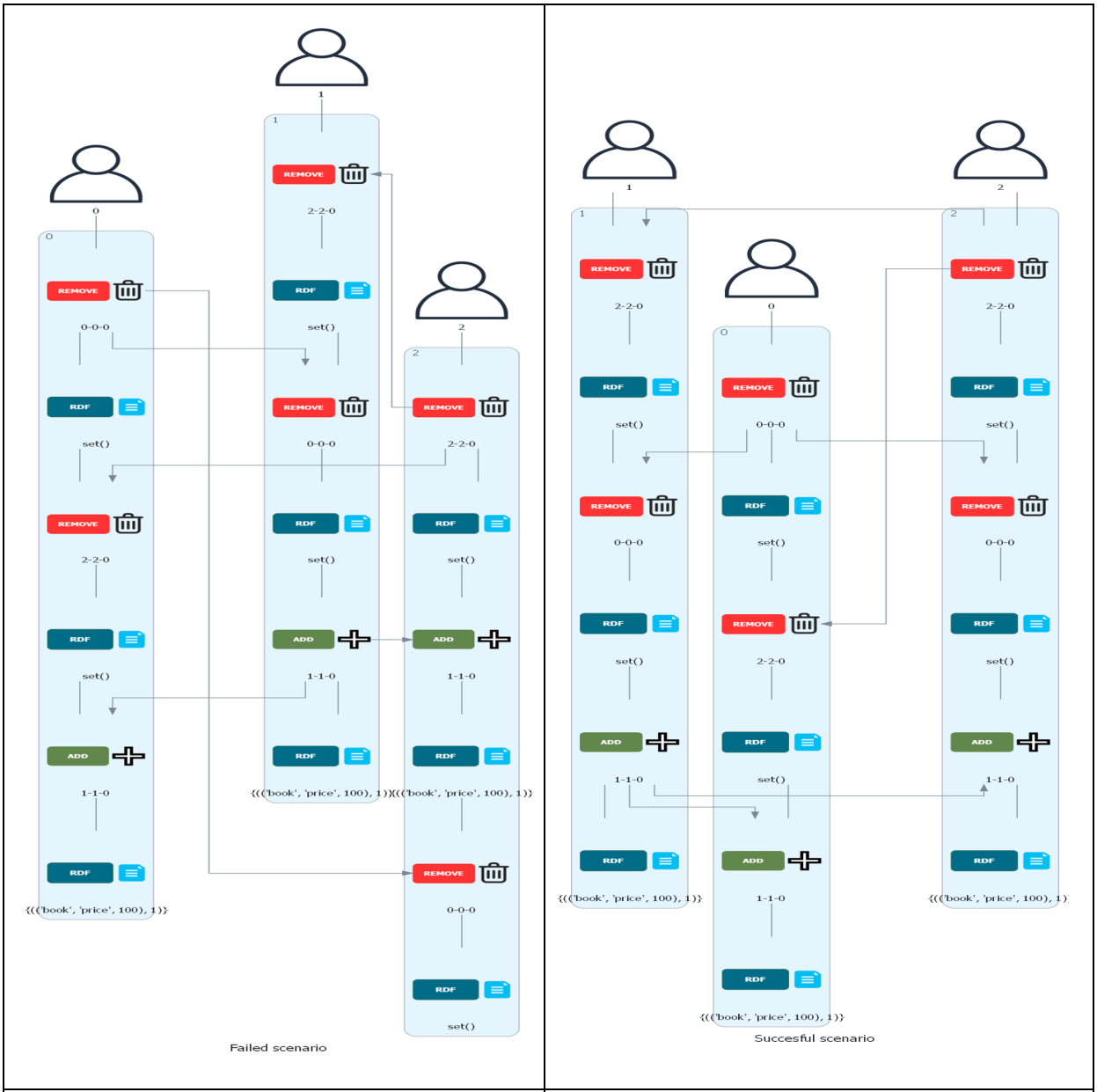
Figure 23 : Transformation du scénario 2 en séquence d'opérations

Pour mieux visualiser les résultats des scenarios de divergences, nous avons implémenté ***un générateur graphique de scenarios***.



(a) Résultat de LD-Set (*diverge*) **(b)** Résultat de notre algorithme (*converge*)

Figure 24 : Visualisation de Scenario 1



(c) Résultat de LD-Set (*diverge*) | **(d)** Résultat de notre algorithme (*converge*)

Figure 25 : Visualisation de Scenario 2

Les Figures 25 et 26 illustrent les visualisations des résultats d'exécution des deux scénarios (1 et 2) déjà présenté dans le chapitre 03. Les figures montrent clairement que notre solution ***converge***, par contre LD-Set ***diverge***.

Pour pousser les expériences de validation un peu loin, nous avons utilisé la classe Permutation pour créer plusieurs scénarios à partir d'un seul scénario. Cela nous permis ***de tester*** LD-Set et notre algorithme sur tous les scénarios de retards d'opérations où nous avons pu analyser les cas de succès et d'échec de LD-Set.

Nous avons exécuté ***10000 scenarios*** grâce à notre validateur, la Figure **27** montre le résultat d'exécution de 100 00 scenarios différents sur LD-Set. Il a réussi a convergé dans **647** situations, mais malheureusement, il a échoué dans **9353** autres.

```
[!]The run contained 10000 scenarios
    (647) scenarios succeeded
    (9353) scenarios failed
[!] The algorithm does not work for all the scenarios
```

Figure 26 : Le résultat du validateur avant l'intégration de notre solution

```
[!]The run contained 10000 scenarios
    (10000) scenarios succeeded
    (0) scenarios failed
[!] The algorithm works for all the scenarios
```

Figure 27 : Lerésultat du validateur après l'intégration de notre solution

La Figure**Erreur ! Source du renvoi introuvable.**28 montre le résultat du validateur pour notre solution sur le même nombre de scenarios (10000), le résultat est tranchant **10000** convergence contre **0** échec.

5.5. Conclusion

Dans ce chapitre nous avons présenté les implémentations de LD-Set, notre solution, et le validateur des scenarios. D'après les résultats des tests des scenarios 1 et 2, et aussi les résultats de générateur automatique des scenarios, nous pouvons dire que notre proposition de solution pour le problème de causalité de LD-Set est correcte et valide.

6. Conclusion Générale

Dans ce mémoire, nous avons présenté le web sémantique, Linked Data et quelques modèles d'édition collaboratifs basés sur l'approche des CRDT. Nous avons évalué les algorithmes CRDT étudiés dans ce mémoire selon le modèle CCI. Malgré leur robustesse elles ont leurs faiblesses. Par conséquent, dans notre projet de fin d’études, nous nous sommes concentrés sur la correction des problèmes de LD-Set. Plus précisément, nous avons proposé une solution qui respecte la condition de causalité dans LD-Set.

Comme perspectives, nous nous intéressons à développer une application qui utilise la version LD-Set amélioré. Nous pensons aussi à analyser sa performance dans une application P2P réel.

Bibliographie

[1] T. Berners-Lee, J. Hollenbach, K. Lu, and J. Presbrey, Tabulator redux : Browsing and writing linked data, CEUR Workshop Proceedings, vol. 369,(01), 2008.

[2] IMINE, A. Decentralized concurrency control for real-time collaborative editors, 8th international conference on New technologies in distributed systems - NOTERE'2008. Lyon, France, (313–321) 2008.

[3] AHMED, D. T., SHIRMOHAMMADI, S . A hybrid p2p protocol for real-time collaboration. Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE'06. 15th IEEE International Workshops on. IEEE, 73–78. 2006.

[4] T. Berners-Lee, J. Hendler, and O. Lassila, The semantic web, Scientic American, vol. 284, pp. 34-43, 2001.

[5] Tuan Anh Ta, Web sémantique et réseaux sociaux - Construction d'une mémoire collective par recommandations mutuelles et représentations. Thèse de doctorat, Télécom ParisTech, 2005.

[6] Hafed Zarzour. Environnement collaboratif sur une infrastructure GRID pour la gestion électronique de documents. Thèse de doctorat, université d'Annaba, 2013

[7] Hafed Zarzour, Mahmoud Al-Ayyoub, Yaser Jararweh, Towards Social Collaborative Editing of Distributed Linked Data, 9th International Conference on Information and Communication Systems (ICICS), 2018

[8] Hanaa Mazyad. Une Approche Multi-agents à Architecture P2P pour l'Apprentissage Collaboratif. Thèse de doctorat en informatique. Université du littoral Côté D'Opale France, 2013

[9] Brian Berliner, CVS II : Parallelizing software development. In Procee- dings of the USENIX Winter 1990 Technical Conference,Berkeley, Californie, États-Unis, 1990

[10] Anne-Marie Kermarrec, Antony I. T. Rowstron, Marc Shapiro,and Peter Druschel. The IceCube approach to the reconciliation of divergent replicas. In Proceedings of the twentieth annual ACM symposium on Principles of distributed computing PODC'01, 2001.

[11] Luis-Daniel Ibanez, Hala Skaf-Molli, Pascal Molli, Oliver Corby. Live Linked Data: Synchronizing Semantic Stores with Commutative Replicated Data Types, International Journal of Metadata, Semantics and Ontologies 8(2), 2013

[12] K. Aslan, P. Molli, H. Skaf-Molli, and S. Weiss, C-Set : a Commutative Replicated Data Type for Semantic Stores, in RED : Fourth International Workshop on REsource Discovery, (Heraklion, Greece), 2011.

[13] Pascal Molli, Hala Skaf-Molli. C-Set: a Commutative Replicated Data Type for Semantic Stores may, In RED: Fourth International Workshop on Resource Discovery, Heraklion, Greece, 2011

[14] Khaled Aslan, Pascal Molli, Hala Skaf-Molli, Stephane Weiss, C-Set : a Commutative Replicated Data Type for Semantic Stores, in RED: Fourth International Workshop on REsource Discovery, 2011.

[15] Bouhenni Ouahiba, Fergane Khadidja, Éditeur collaboratif distribué pour les données sémantiques, Rapport de mini-projet, université de Mostaganem, 2020.

[16] H. Zarzour and M. Sellami, B-set: A synchronization method for distributed semantic stores, in IEEE International Conference on Complex Systems (ICCS), pp. 16, 2012

[17] Hafed Zarzour Mokhtar Sellami, B-Set: A synchronization method for distributed semantic stores, 2012.

[18] Hafed Zarzour, Mokhtar Sellami. srCE: a collaborative editing of scalable semantic stores on P2P networks, Int. J. Computer Applications in Technology, 2013

[19] Saito, Y., Shapiro, M., Optimistic replication, ACM Computing Surveys (CSUR), 2005

[20] Preguica, N., Marques, J. M., Shapiro, M., Letia, M. : A commutative replicated data type for cooperative editing. In : Distributed Computing Systems, 2009. ICDCS'09. 29th IEEE International Conference on. IEEE, p. 395-403, 2009

[21] https://github.com/t-mullen/observed-remove-set

[22] IoanaRobu , Valentin Robu, Benoit Thirion , An introduction to the Semantic Web for health sciences librarians , J Med Libr Assoc, 2006

[23] Mechaoui Moulay Driss, Le contrôle d'un modèle de Garbage Collection dans les réseaux Sans fils distribués pour l'édition collaborative. Thèse de doctorat, université d' USTO, (2017/2018).

[24] C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen. Achieving convergence, causality-preservation and intention-preservation in real-time cooperative editing systems. ACM Trans. Comput.-Hum. Interact, 1998.