

# Ré-identification efficace dans les types de données répliquées sans conflit (CRDTs)

## THÈSE

présentée et soutenue publiquement le 28 janvier 1986

pour l'obtention du

**Doctorat de l'Université de Lorraine**  
(mention informatique)

par

Matthieu Nicolas

### Composition du jury

|                                |                          |
|--------------------------------|--------------------------|
| <i>Président :</i>             | Stephan Merz             |
| <i>Rapporteurs :</i>           | Le rapporteur 1 de Paris |
|                                | Le rapporteur 2          |
|                                | suite taratata           |
|                                | Le rapporteur 3          |
| <i>Examineurs :</i>            | L'examineur 1 d'ici      |
|                                | L'examineur 2            |
| <i>Membres de la famille :</i> | Mon frère                |
|                                | Ma sœur                  |

Mis en page avec la classe thesul.

## Remerciements

Les remerciements.



*Je dédie cette thèse  
à ma machine.  
Oui, à Pandore,  
qui fut la première de toutes.*



# Sommaire

|                                                                       |          |
|-----------------------------------------------------------------------|----------|
| <b>Introduction</b>                                                   | <b>1</b> |
| 1 Contexte . . . . .                                                  | 1        |
| 2 Questions de recherche . . . . .                                    | 1        |
| 3 Contributions . . . . .                                             | 1        |
| 4 Plan du manuscrit . . . . .                                         | 1        |
| <b>Problématique</b>                                                  | <b>3</b> |
| <b>Chapitre 1</b>                                                     |          |
| <b>État de l’art</b>                                                  | <b>5</b> |
| 1.1 Transformées opérationnelles . . . . .                            | 5        |
| 1.2 Séquences répliquées sans conflits . . . . .                      | 5        |
| 1.2.1 Type de données répliquées sans conflits . . . . .              | 5        |
| 1.2.2 Approches pour les séquences répliquées sans conflits . . . . . | 5        |
| 1.3 LogootSplit . . . . .                                             | 5        |
| <b>Chapitre 2</b>                                                     |          |
| <b>Renommage dans un système centralisé</b>                           | <b>7</b> |
| 2.1 Approche . . . . .                                                | 8        |
| 2.2 RenamableLogootSplit . . . . .                                    | 8        |
| 2.2.1 Opération de renommage . . . . .                                | 8        |
| 2.2.2 Gestion des opérations concurrentes au renommage . . . . .      | 8        |
| 2.2.3 Récupération de la mémoire des états précédents . . . . .       | 8        |
| 2.2.4 Modèle de cohérence . . . . .                                   | 8        |
| 2.3 Évaluation . . . . .                                              | 8        |
| 2.3.1 Expérimentations . . . . .                                      | 8        |
| 2.3.2 Résultats . . . . .                                             | 8        |

|       |                                                               |   |
|-------|---------------------------------------------------------------|---|
| 2.4   | Discussion . . . . .                                          | 8 |
| 2.4.1 | Stockage d'anciens états sur disque . . . . .                 | 8 |
| 2.4.2 | Compression de l'opération de renommage . . . . .             | 8 |
| 2.4.3 | Limitation de la taille de l'opération de renommage . . . . . | 8 |

### Chapitre 3

#### Renommage dans un système distribué 9

|       |                                                                     |    |
|-------|---------------------------------------------------------------------|----|
| 3.1   | Approche . . . . .                                                  | 10 |
| 3.2   | RenamableLogootSplit v2 . . . . .                                   | 10 |
| 3.2.1 | Opération de renommage et inversion du renommage . . . . .          | 10 |
| 3.2.2 | Relation de priorité entre renommages concurrents . . . . .         | 10 |
| 3.2.3 | Identification des renommages à inverser et à appliquer . . . . .   | 10 |
| 3.2.4 | Récupération de la mémoire des états précédents . . . . .           | 10 |
| 3.3   | Évaluation . . . . .                                                | 10 |
| 3.3.1 | Expérimentations . . . . .                                          | 10 |
| 3.3.2 | Résultats . . . . .                                                 | 10 |
| 3.4   | Discussion . . . . .                                                | 10 |
| 3.4.1 | Définition de relations de priorité plus optimales . . . . .        | 10 |
| 3.4.2 | Report de la transition vers la nouvelle epoch principale . . . . . | 10 |

### Chapitre 4

#### Stratégies de déclenchement du renommage 11

|       |                                |    |
|-------|--------------------------------|----|
| 4.1   | Motivation . . . . .           | 11 |
| 4.2   | Stratégies proposées . . . . . | 11 |
| 4.2.1 | Propriétés . . . . .           | 11 |
| 4.2.2 | Stratégie 1 : ??? . . . . .    | 11 |
| 4.2.3 | Stratégie 2 : ??? . . . . .    | 11 |
| 4.3   | Évaluation . . . . .           | 11 |

### Chapitre 5

#### Conclusions et perspectives 13

|       |                                                              |    |
|-------|--------------------------------------------------------------|----|
| 5.1   | Résumé des contributions . . . . .                           | 13 |
| 5.2   | Perspectives . . . . .                                       | 13 |
| 5.2.1 | Définition de relations de priorité plus optimales . . . . . | 13 |



---

|       |                                                                                |    |
|-------|--------------------------------------------------------------------------------|----|
| 5.2.2 | Redéfinition de la sémantique du renommage en déplacement d'éléments . . . . . | 13 |
| 5.2.3 | Définition de types de données répliquées sans conflits plus complexes         | 13 |

|                                       |
|---------------------------------------|
| <b>Annexe A</b><br><b>Algorithmes</b> |
|---------------------------------------|



# Table des figures



# Introduction

- 1 Contexte
- 2 Questions de recherche
- 3 Contributions
- 4 Plan du manuscrit



# Problématique





# Chapitre 1

## État de l'art

### Sommaire

|            |                                                                 |          |
|------------|-----------------------------------------------------------------|----------|
| <b>1.1</b> | <b>Transformées opérationnelles . . . . .</b>                   | <b>5</b> |
| <b>1.2</b> | <b>Séquences répliquées sans conflits . . . . .</b>             | <b>5</b> |
| 1.2.1      | Type de données répliquées sans conflits . . . . .              | 5        |
| 1.2.2      | Approches pour les séquences répliquées sans conflits . . . . . | 5        |
| <b>1.3</b> | <b>LogootSplit . . . . .</b>                                    | <b>5</b> |

### 1.1 Transformées opérationnelles

### 1.2 Séquences répliquées sans conflits

#### 1.2.1 Type de données répliquées sans conflits

#### 1.2.2 Approches pour les séquences répliquées sans conflits

### 1.3 LogootSplit



# Chapitre 2

## Renommage dans un système centralisé

### Sommaire

---

|            |                                                     |          |
|------------|-----------------------------------------------------|----------|
| <b>2.1</b> | <b>Approche</b>                                     | <b>8</b> |
| <b>2.2</b> | <b>RenamableLogootSplit</b>                         | <b>8</b> |
| 2.2.1      | Opération de renommage                              | 8        |
| 2.2.2      | Gestion des opérations concurrentes au renommage    | 8        |
| 2.2.3      | Récupération de la mémoire des états précédents     | 8        |
| 2.2.4      | Modèle de cohérence                                 | 8        |
| <b>2.3</b> | <b>Évaluation</b>                                   | <b>8</b> |
| 2.3.1      | Expérimentations                                    | 8        |
| 2.3.2      | Résultats                                           | 8        |
| <b>2.4</b> | <b>Discussion</b>                                   | <b>8</b> |
| 2.4.1      | Stockage d'anciens états sur disque                 | 8        |
| 2.4.2      | Compression de l'opération de renommage             | 8        |
| 2.4.3      | Limitation de la taille de l'opération de renommage | 8        |

---

## 2.1 Approche

## 2.2 RenamableLogootSplit

### 2.2.1 Opération de renommage

Propriétés

Proposition

### 2.2.2 Gestion des opérations concurrentes au renommage

### 2.2.3 Récupération de la mémoire des états précédents

### 2.2.4 Modèle de cohérence

## 2.3 Évaluation

### 2.3.1 Expérimentations

Scénario d'expérimentation

Implémentation des simulations

### 2.3.2 Résultats

Convergence

Consommation mémoire

Temps d'intégration des opérations "simples"

Temps d'intégration de l'opération de renommage

## 2.4 Discussion

### 2.4.1 Stockage d'anciens états sur disque

### 2.4.2 Compression de l'opération de renommage

### 2.4.3 Limitation de la taille de l'opération de renommage

# Chapitre 3

## Renommage dans un système distribué

### Sommaire

---

|            |                                                           |           |
|------------|-----------------------------------------------------------|-----------|
| <b>3.1</b> | <b>Approche</b>                                           | <b>10</b> |
| <b>3.2</b> | <b>RenamableLogootSplit v2</b>                            | <b>10</b> |
| 3.2.1      | Opération de renommage et inversion du renommage          | 10        |
| 3.2.2      | Relation de priorité entre renommages concurrents         | 10        |
| 3.2.3      | Identification des renommages à inverser et à appliquer   | 10        |
| 3.2.4      | Récupération de la mémoire des états précédents           | 10        |
| <b>3.3</b> | <b>Évaluation</b>                                         | <b>10</b> |
| 3.3.1      | Expérimentations                                          | 10        |
| 3.3.2      | Résultats                                                 | 10        |
| <b>3.4</b> | <b>Discussion</b>                                         | <b>10</b> |
| 3.4.1      | Définition de relations de priorité plus optimales        | 10        |
| 3.4.2      | Report de la transition vers la nouvelle epoch principale | 10        |

---

## 3.1 Approche

## 3.2 RenamableLogootSplit v2

### 3.2.1 Opération de renommage et inversion du renommage

Propriétés

Proposition

### 3.2.2 Relation de priorité entre renommages concurrents

### 3.2.3 Identification des renommages à inverser et à appliquer

### 3.2.4 Récupération de la mémoire des états précédents

## 3.3 Évaluation

### 3.3.1 Expérimentations

Scénario d'expérimentation

Implémentation des simulations

### 3.3.2 Résultats

Convergence

Consommation mémoire

Temps d'intégration des opérations "simples"

Temps d'intégration de l'opération de renommage

## 3.4 Discussion

### 3.4.1 Définition de relations de priorité plus optimales

### 3.4.2 Report de la transition vers la nouvelle epoch principale

# Chapitre 4

## Stratégies de déclenchement du renommage

### 4.1 Motivation

### 4.2 Stratégies proposées

#### 4.2.1 Propriétés

#### 4.2.2 Stratégie 1 : ???

#### 4.2.3 Stratégie 2 : ???

### 4.3 Évaluation

#### Sommaire

---

|       |                                |    |
|-------|--------------------------------|----|
| 4.1   | Motivation . . . . .           | 11 |
| 4.2   | Stratégies proposées . . . . . | 11 |
| 4.2.1 | Propriétés . . . . .           | 11 |
| 4.2.2 | Stratégie 1 : ??? . . . . .    | 11 |
| 4.2.3 | Stratégie 2 : ??? . . . . .    | 11 |
| 4.3   | Évaluation . . . . .           | 11 |

---





# Chapitre 5

## Conclusions et perspectives

### Sommaire

---

|            |                                                                                       |           |
|------------|---------------------------------------------------------------------------------------|-----------|
| <b>5.1</b> | <b>Résumé des contributions . . . . .</b>                                             | <b>13</b> |
| <b>5.2</b> | <b>Perspectives . . . . .</b>                                                         | <b>13</b> |
| 5.2.1      | Définition de relations de priorité plus optimales . . . . .                          | 13        |
| 5.2.2      | Redéfinition de la sémantique du renommage en déplacement<br>d'éléments . . . . .     | 13        |
| 5.2.3      | Définition de types de données répliquées sans conflits plus com-<br>plexes . . . . . | 13        |

---

### 5.1 Résumé des contributions

### 5.2 Perspectives

#### 5.2.1 Définition de relations de priorité plus optimales

#### 5.2.2 Redéfinition de la sémantique du renommage en déplacement d'éléments

#### 5.2.3 Définition de types de données répliquées sans conflits plus complexes



# Annexe A

## Algorithmes



## Résumé

Afin d'assurer leur haute disponibilité, les systèmes distribués à large échelle se doivent de répliquer leurs données tout en minimisant les coordinations nécessaires entre noeuds. Pour concevoir de tels systèmes, la littérature et l'industrie adoptent de plus en plus l'utilisation de types de données répliquées sans conflits (CRDTs). Les CRDTs sont des types de données qui offrent des comportements similaires aux types existants, tel l'Ensemble ou la Séquence. Ils se distinguent cependant des types traditionnels par leur spécification, qui supporte nativement les modifications concurrentes. À cette fin, les CRDTs incorporent un mécanisme de résolution de conflits au sein de leur spécification.

Afin de résoudre les conflits de manière déterministe, les CRDTs associent généralement des identifiants aux éléments stockés au sein de la structure de données. Les identifiants doivent respecter un ensemble de contraintes en fonction du CRDT, telles que l'unicité ou l'appartenance à un ordre dense. Ces contraintes empêchent de borner la taille des identifiants. La taille des identifiants utilisés croît alors continuellement avec le nombre de modifications effectuées, aggravant le surcoût lié à l'utilisation des CRDTs par rapport aux structures de données traditionnelles. Le but de cette thèse est de proposer des solutions pour pallier ce problème.

Nous présentons dans cette thèse deux contributions visant à répondre à ce problème : (i) Un nouveau CRDT pour Séquence, *RenamableLogootSplit*, qui intègre un mécanisme de renommage à sa spécification. Ce mécanisme de renommage permet aux noeuds du système de réattribuer des identifiants de taille minimale aux éléments de la séquence. Cependant, cette première version requiert une coordination entre les noeuds pour effectuer un renommage. L'évaluation expérimentale montre que le mécanisme de renommage permet de réinitialiser à chaque renommage le surcoût lié à l'utilisation du CRDT. (ii) Une seconde version de *RenamableLogootSplit* conçue pour une utilisation dans un système distribué. Cette nouvelle version permet aux noeuds de déclencher un renommage sans coordination préalable. L'évaluation expérimentale montre que cette nouvelle version présente un surcoût temporaire en cas de renommages concurrents, mais que ce surcoût est à terme.

**Mots-clés:** CRDTs, édition collaborative en temps réel, cohérence à terme, optimisation mémoire, performance

## Abstract

**Keywords:** CRDTs, real-time collaborative editing, eventual consistency, memory-wise optimisation, performance

