République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université d'Abbès Laghrour Khenchela
Faculté des Sciences et de la Technologie
Département des Mathématiques et d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique
Spécialité : Génie logiciel et systèmes distribués.

Qualité des données, Elimination des doubles et des similaires

Présenté par :

Loubna BENDJADOU.

Nourhane BOUZIANE.

Dirigé par :

Dr.Mohamed Mahdi MALIK.

Jury :

Examinateur : Dr.Rafik MAHDAOUI.

Examinateur : M.Nabil AZIZI.

Encadreure : Dr.Mohamed Mahdi MALIK.

Promotion : 2019/2020

Table des matières

Table des figures

Liste des tableaux

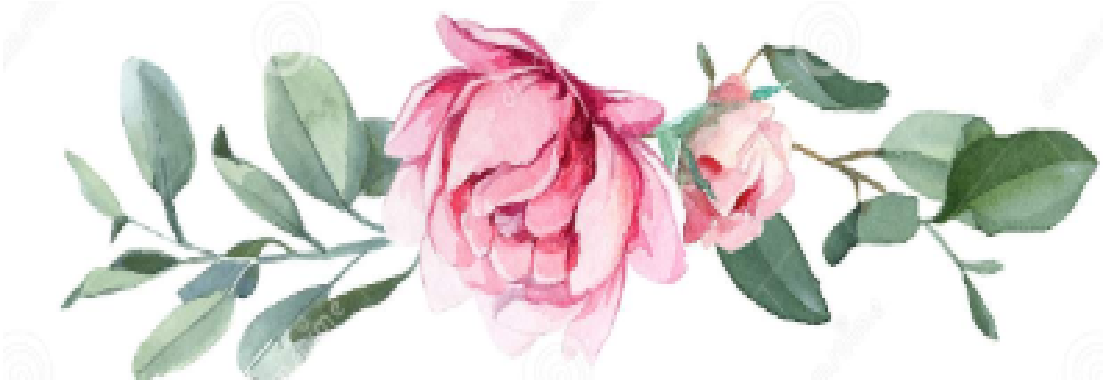
Liste des abréviations

ED	Entrepots de Données.
ETL	Extract Transform and Load.
SGBD	système de gestion de base de données
BD	Base de Données
TDI	Talend Data Integration.
PDI	Pentaho Data Integration.
IBAN	International Bank Account Number
JW	Jaro-Winkler
Jac	Jaccard
Lv	Levenshtein
Qg	Q-gram
Sdx	Soundex
DM	Double Metaphone
Ny	Nysiis
JVM	Machine Virtuelle Java
CDDL	Common Development and Distribution License
XML	eXtended Markup Language
HTML	HyperText Markup Language
IDE	Integrated Development Environment
SGBDRO	système de gestion de base de données relationnel-objet
SQL	Structured Query Language

Remerciment

Tout d'abord, nous remercions Dieu le tout-puissant qui nous a donné le courage,
la force et la volonté pour mener ce travail.
Nous adressons nos vifs remerciements à notre encadreur Mr. Mehdi Malik,
qui n'a pas cessé de nous donner les conseils et les bonnes orientations et nous prive pas de son temps et ses corrections nous ont été d'une grande utilité .
Nous tenons à remercier les membres de notre jury, pour avoir accepté de juger notre travail.
Leurs remarques et suggestions nous seront trés utiles pour la finalisation de ce mémoire.
Nous exprimons notre profonde gratitude à nos parents, qui ont fait de nous ce que nous sommes aujourd'hui.
Ils nous ont beaucoup apporté tout au long de notre existence,
ils nous ont offerts les moyens et tous les efforts qui vont avec pour nous voir réussir dans nos vies.
Nous remercions profondément nos frères,
qui nous ont soutenus et conseillés durant toutes ces années d'études. A nos chères amis qui ont toujours été présents et fidèles.
Nous exprimons notre profonde gratitude à toute personne qui,
de près ou de loin, a contribué à l'élaboration de ce travail .

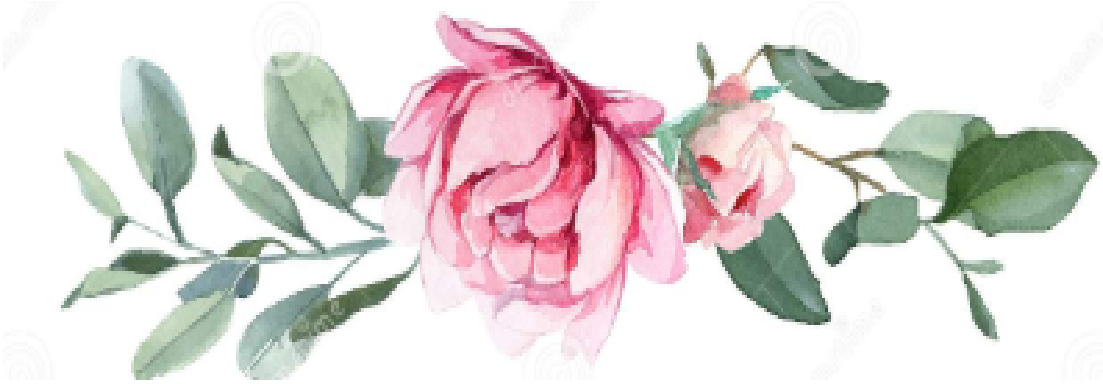
Loubna et Nourhane



Dédicace

Je tiens à dédier ce modeste travail à la lumière de ma vie : mes chers parents,
mon père Belkacem, et ma mère Zahia pour leur plus grand amour, soutien, encouragement de la patience et de l'aide continue pendant mes années d'études.
Ce travail est également dédié aux fleurs de ma vie :mon cher mari Abd elkader mahdaoui et ma petite princesse Lyne. .
(Que Dieu ,le tout puissant, vous préserve, vous accorde santé, bonheur, quiétude de l'esprit et vous protège de tout mal).
A mes chers frères :Abd elhak et Aymen et Badreddine.
A mes adorables amies :Amani,Houda,Chaima .
A mon cher binôme Nourhane.

Loubna



Dédicace

En premier lieu, je veux rendre grâce à Dieu, le Clément et le Très Miséricordieux pour son amour éternel. Je dédie cette thèse à :

ma mère fatima pour sa tendresse et à mon père abdelouadjel pour sa patience et son encouragement

mes très chers frères Abdelbasset , Dhiaa el din, Moslem

ma chère sœur Belkisse

mes oncles , mes tantes pour leurs conseils,

ma binome «Benjeddou loubna »

à tous ceux que j'aime,

à tous mes amis .

Nourhane

Résumé

Cet article présente la problématique et les défis liés à l'évaluation et l'amélioration de la qualité des données. Il décrit les solutions issues du monde de la recherche ainsi que les principales approches mises en œuvre en pratique pour gérer les problèmes de qualité des données que sont les données doubles ou les similaire...

Les principales techniques pour le diagnostic et la correction y sont présentées pour permettre la modélisation, la mesure, le contrôle et l'amélioration de la qualité des données dans les bases et les entrepôts de données structurées.

Abstract

This article presents the issues and challenges related to the evaluation and improvement of data quality. It describes solutions from the world of research as well as the main approaches implemented in practice to manage data quality problems such as double data or similar ... The main techniques for diagnosis and correction are presented there to allow modeling, measurement, control and improvement of data quality in databases and structured data warehouses.

ملخص

تعرض هذه المقالة التحديات المتعلقة بتقييم جودة و البيانات وتحسينها. كما تصف الحلول الرئيسية والأساليب المطبقة في عملية إدارة مشاكل جودة البيانات مثل البيانات المزدوجة أو ما شابه ذلك ...

تم تقديم التقنيات الرئيسية للتشخيص والتصحيح للسماح بنمذجة وقياس ومراقبة وتحسين جودة البيانات في قواعد البيانات

مستودعات البيانات المنظمة.

Chapitre 1

Introduction générale

1.1 Introduction

Les travaux actuels sur l'extraction de connaissances 'a partir d'un environnement informationnel, qui se caractérise par de trés grosses masses de données hétérogénes et distribuées dans les entrepots de données (ED), se focalisent principalement sur la recherche d'information potentielle, utile et préalablement inconnue.
La qualité de l'information recueillie depend de celles des données. Prendre des decisions 'a partir de mauvaises informations peut nuire 'a l'organisation, d'ou un cout de la non-qualité qui peut s'avérer trés élevé.
La construction d'un ED, issu de l'intégration de sources totalement hétérogènes de qualité variable, et d'outils d'aide à la decision issus de ces masses d'informations necessite developpement de nouveaux outils d'extraction et de transformation de données (ETL - Extract Transform and Load).
Ces derniers doivent, d'une part, prendre en compte l'hétérogénéité des données et leurs contraintes, et d'autre part, assurer la qualité du nouvel ensemble de données construit.
Dans ce papier nous nous intéressons à la problématique de la qualité des données dans ces ED et plus précisément au problème d'elimination des doublons et des données similaires , Les données similaires sont des données qui ont des ressemblances au niveau de leurs valeurs.
Le problème de l'elimination des données similaires (doublons non strictes) et de la fusion/intégration de données est très complexe. En effet, il s'agit de localiser, fusionner et enrichir des entités qui représentent le meme monde réel.
Dans ce cadre s'inscrit notre travail qui consiste à Implémenter des algorithmes parallèles qui permettent l'élimination des doublons et des similaires.

1.2 Qualité des données

1.2.1 Définition

La qualité des données est une mesure de l'état des données fondée sur divers facteurs : précision, exhaustivité, homogénéité, fiabilité et actualité.

Mesurer le niveau de qualité des données peut aider les organisations à repérer d'éventuelles erreurs qui doivent être corrigées, et à évaluer si les données présentes dans leurs systèmes informatiques sont adaptées à leurs besoins.
La qualité des données dans les systèmes d'entreprise prend une importance croissante depuis que le traitement des données devient un élément incontournable de l'exploitation et que des solutions analytiques sont de plus en plus utilisées comme outil d'aide à la prise de décision.

1.2.2 Cout la non qualité :

Le plus simple est peut-être de partir de ce que l'on entend par le terme de qualité.

La définition qu'en donne l'Organisation Internationale de Normalisation est des plus claires : « la qualité est l'aptitude d'un produit ou d'un service à satisfaire les exigences spécifiées ».

De façon plus simpliste, quand on achète un bien ou un service, on s'attend à ce qu'il réponde à certains critères plus ou moins subjectifs. Les tomates doivent être goûteuses, un smartphone offrir une bonne autonomie et un train arriver à l'heure.

Les entreprises ont bien compris les enjeux de la qualité et depuis des années mettent en œuvre tout un ensemble d'outils et méthodes destinés à offrir des produits et des services de qualité.

Avec une telle définition de la qualité, celle de la non-qualité semble évidente. La non qualité réside dans l'inaptitude à répondre aux attentes des entreprises, des consommateurs ou des usagers.

J'adore cette anecdote que j'avais entendue. Un stagiaire candide, demandait aux développeurs pourquoi ils mettaient des bugs dans leurs programmes.

Il ne s'agit pas ici de comprendre les raisons de la non-qualité. Elle peut être acceptée comme dans le cas du manager dont je parlais plus haut. Mais elle peut être subie suite à des défaillances de processus, une inadéquation de compétences ou de ressources, face à la contrainte de la concurrence ou la pression d'un client.

Le problème c'est que cette non-qualité a un coût ... Ou plutôt la non-qualité a DES coûts

Les 4 types de coûts de non qualité :

Les coûts de non-qualité, frais "cachés" issus des pertes provenant d'une mauvaise qualité, représentent un levier de compétitivité intéressant. On distingue 4 types de coûts de non-qualité :

1. Les coûts de non-qualité ou anomalies internes Ces coûts regroupent les frais relatifs à un produit non conforme aux exigences qualité avant qu'il n'ait quitté l'entreprise. Il peut par exemple s'agir des coûts liés à la mise au rebut, ou à l'arrêt de production.
2. Les coûts de non-qualité ou anomalies externes Ces coûts regroupent les frais relatifs à un produit non conforme aux exigences qualité après qu'il n'ait quitté l'entreprise. Ils comptent par exemple les coûts de reprise ou d'arrêt de production chez le client.
3. Les coûts de prévention Ces coûts sont ceux associés aux investissements de tout type (humains comme matériels) engagés pour prévenir et réduire les anomalies, et vérifier leur existence. Ils incluent notamment les frais de maintien du système qualité et les activités d'assurance qualité telles que la formation ou les charges du service Qualité liées à la prévention.

4. Les coûts de détection Ces coûts sont associés aux dépenses engagées pour vérifier la conformité du produit aux exigences Qualité. Il s'agit en d'autre termes des frais générés par la recherche de la non-qualité, tels que les coûts des machines de contrôle ou les charges du service Qualité affectées à cette activité.

1.2.3 Dimensions de la qualité

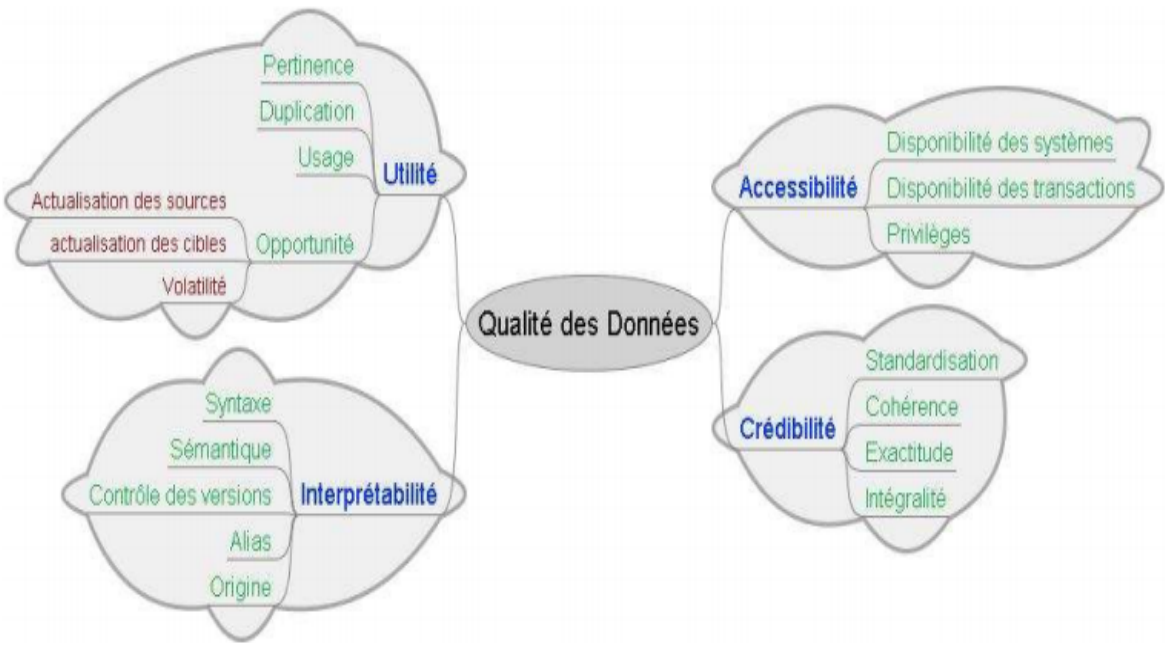


FIGURE 1.1 – Les dimensions de la qualité des données

la figure 1.1 ilustre les nombreux aspects de la qualité des données. Pour définir les problèmes de qualité des donnée, il est recommandé de définir les dimensions possibles et leur importance :

- ***Duplication :*** les données sont répétées.l'entté est gérée par plusieures systèmes d'informations sous des idantifiantes différents et donc sa vue n'est pas unifiée.
- ***Standard :*** les valeurs sont corrects par rapport à un intervalle de répartition ou à un domaines par manque de standard de condition.
- ***Intégralité :*** toutes les données nécessaires sont disponible pour le besoin de métier.
- ***Exactitude :*** les données représentent la réalité ou sont vérifiables à partir d'une source externe.
- ***Interprétabilité :***un donnée doit ètre représentée sous un format cohérent et sans ambiguité.
- ***Opportunité :*** les données sont à jour au moment de leur utilisation.[1]

1.2.4 Indicateurs et mesures de la qualité des données

Mesures de la qualité	*Indicateurs de qualité*
◦ Ce sont des variables qui mesurent directement un aspect particulier de la qualité (ex : décalage temporel entre la date de référence et la date de publication) ◦ La plupart d'entre elles sont en pratique coûteuses et difficiles à calculer	◦ Ils résument par un nombre l'information afin de fournir des indications sur le niveau de qualité des données ◦ Ils ne mesurent pas la qualité directement mais fournissent suffisamment d'information pour l'évaluation de la qualité (ex : le taux de réponse est un indicateur de qualité [un proxy] de la mesure bu biais lié au faible niveau relatif de réponses

TABLE 1.1 – Indicateurs de qualité vs Mesures directes de la qualité [2]

chaque dimension peut etre mesurée soit de manière subjectiv en recueillant la perception des utilisateurs , soit de manière objective au travers de suivis automatiques des indicateurs spécifiques. (TABLE 1.2) donne des exemples d'indicateurs de qualité suivant différents critères

Critères de Qualité des Données	Caractéristiques	Exemples d'Indicateurs
Opportunité	L'âge des données est-il conforme aux besoins métiers ?	Date de la collecte des données Date du dernier traitement Contrôle de la version
Intégralité / Complétude	Est-ce que toutes les données nécessaires sont disponibles ?	Intégralité des valeurs optionnelles Nombre de valeurs non renseignées Nombre de valeurs par défaut par rapport à la moyenne
Cohérence	Quelles sont les données sources des informations contradictoires ?	Vérification de plausibilité Valeur de la déviation standard
Exactitude	Les valeurs représentent-elles la réalité ?	Fréquence des changements de valeur Réaction (feedback) des clients
Interprétabilité	Les données sont-elles compréhensibles par les utilisateurs ?	Valorisation des données utilisateur Violation de domaines
Standardisation, conformité	Quelles sont les données saisies, stockées ou affichées dans un format non standard ?	Certificat de conformité
Duplication	Quelles sont les données répétées ?	Nombre d'enregistrements dupliqués

TABLE 1.2 – Exemple d'ndicateurs de qualité

une fois les indicateures définis il faut mettre en place un système de mesure qui permette de surveiller leur évolutions dans le temps. La publications des indicateurs de qualité , leur cible et leur évolution permettent de définir les plans d'action éventuels mettre en oeuvre pour corriger un situation .
(FIGURE 1.2) montre un exemple d'indicateur et son évolution dans le temps[1].

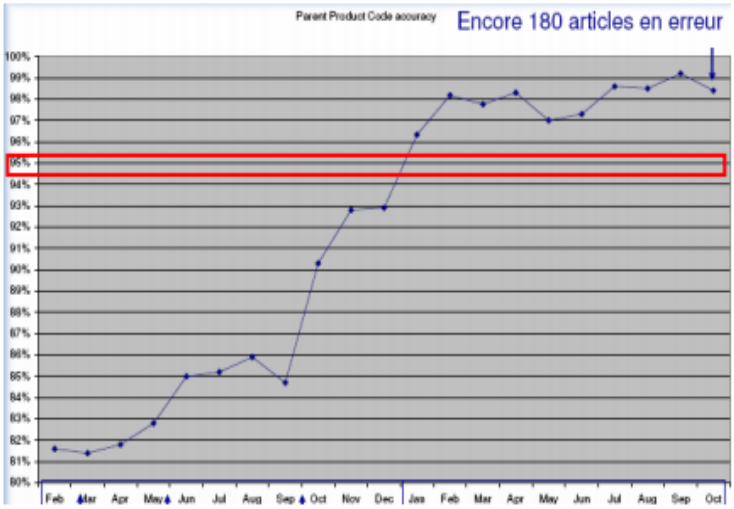


FIGURE 1.2 – Mesures d'un indicateure

1.3 problematique

Les problèmes de qualité des données stockées dans les bases et les entrepôts de données se propagent de façon endémique à tous les types de données (structurées ou non) et dans tous les domaines d'application : données gouvernementales, commerciales, industrielles ou scientifiques.
Il s'agit en particulier d'erreurs sur les données, de doublons, d'incohérences, de valeurs manquantes, incomplètes, incertaines, obsolètes, aberrantes ou peu fiables.
Les conséquences de la non qualité des données (ou de leur qualité médiocre) sur les prises de décision et les coûts financiers qu'elle engendre sont considérables.
Avec la multiplication des sources d'informations disponibles et l'accroissement des volumes de données potentiellement accessibles, la qualité des données et, plus largement, la qualité des informations ont pris une place de premier plan, d'abord, au sein des entreprises et, depuis ces dix dernières années, dans le monde académique .
Il est urgent de proposer des solutions théoriques et pratiques aux multiples problèmes de qualité des données , Quelles sont ces solutions ?

1.4 objectif

Notre objectif dans cette thèse est premièrement d'étudier la qualité de l'information à partir de toutes les informations disponibles sur les données (données, métadonnées, résultats d'analyses et informations fournies par l'utilisateur). Pour ce faire, il faudra modéliser la sémantique des données et le contexte de leur utilisation.
deuxièmement, corriger et nettoyer les données. Nous devrons élaborer des procédures permettant d'extraire et de mettre à jour ces méta-informations sur les données. Puis, il s'agira d'exploiter la sémantique des données reconstruite dans leur contexte pour aider à la correction des problèmes de qualité lors de processus d'intégration des données hétérogènes.
Notamment,des règles de qualité pourront être appliquées pour l'homogénéisation des données ou la fusion des données dans le cas de traitement des doublons. D'autres règles permettront de corriger les données en fonction du contexte d'application.
L'enrichissent des données par des informations sémantiques permet un meilleur nettoyage de ces dernières.
Le nettoyage des données se fait en deux étapes :
la première étape est la détection des anomalies.
la deuxième est la correction de ces dernières.
Quand la sémantique de ces données existe et bien présentée, une meilleure et plus simple détection est réalisée et de même pour le nettoyage.

Chapitre 2

Etat de l'art

2.1 Introduction

Les préoccupations stratégiques pour de nombreuses entreprises tournent de nos jour autour des données (bases et entrepôts de données, BigData). Ces dernières sont devenues incontournables pour une meilleure gestion, pour aider au développement d'outils d'aide à la décision et de prédiction. Les données sont très volumineuses, hétérogènes, distribuées et de qualité variable. Plusieurs types d'anomalies (notamment des doublons, des données similaires, des données aberrantes, des données obsolètes et des valeurs nulles) ont été relevés dans les données rassemblées. En conséquence, la qualité aussi bien de la gestion que de l'extraction des connaissances à partir de ces données peuvent s'avérer mauvaise. De surcroît, les coûts financiers qu'engendre la non-qualité des données, ou leur qualité médiocre, sur les prises de décision risquent d'être considérables (Toulemonde, 2008).
On peut facilement constater que les outils de manipulations de données tels que les Systèmes de Gestion de Bases de Données (SGBD) et les outils d'intégration (ETL) d'aujourd'hui sont très manuels. A la charge de utilisateur de définir le modèle de sortie (la structure détaillée des données résultats). Il devra ainsi être un expert ou une personne du domaine afin d'une part, de comprendre la définition des données sources, et d'autre part, de décider des différentes conversions et transformations à réaliser sur ces dernières. L'utilisateur ne bénéfice d'aucune aide sémantique pour effectuer cette tâche primordiale et ne bénéfice pas non plus de recommandations. Non seulement, les données sources hétérogènes sont généralement mal décrites, mais aussi elles peuvent être imparfaites et polluées par la présence de plusieurs types d'anomalies. Ceci complique énormément toute tâche de manipulation de données à savoir l'intégration des données, les calculs sur celles-ci et l'extraction des connaissances. Cette dernière, mal faite, pourrait contribuer à injecter dans le résultat de nouvelles anomalies.
L'opération de rassemblement des données afin de construire de nouvelles bases et entrepôts de données (Hamdoun and Boufarès, 2010), (Badri et al., 2009) ainsi que des outils d'aides à la décision issus de ces très grosses masses de données hétérogènes et distribuées nécessite le développement de nouveaux ETL . Ces derniers doivent, d'une part, prendre en compte l'hétérogénéité des données et leurs contraintes, et d'autre part, assurer la qualité du nouvel ensemble construit tout en redonnant un sens et une crédibilité aux données rassemblées.

Dans ce chapitre, nous commencerons par donner une liste non exhaustive des anomalies dans les métadonnées et dans les données elles-même. Dans un deuxième temps on abordera les solutions présentées dans la littérature afin de remédier à ces anomalies[5].

Définition 2.1 : Types de données

Dans un système de types, les types sont construits à partir des types de base (prédéfinis) BASE et de constructeurs de types CONST TYPES* . Un constructeur de type prend une séquence de types et construit un nouveau type.
$TYPE: := BASE|CONSTTYPES*$ [5]

Propriété de confluence de type

Soient dt_1 ,dt_2 ,dt_3 ,dt_4 des types de données. La notation $dt_1 \longrightarrow dt_2$ pour dire que $x: dt_1$ (x est de type dt_1) peut être vue comme $x: dt_2$ (x est de type dt_2).

si $dt_1 \longrightarrow *dt_3$ et $dt_2 \longrightarrow *dt_3$ alors il existe dt_4 tels que $dt_2 \longrightarrow *dt_4$ et $dt_3 \longrightarrow *dt_4$, avec $\longrightarrow *estlafermeturetransitivede \longrightarrow$.

Définition 2.2 : Compatibilité des types de données

Deux types dt_1 et dt_2 sont compatibles s'il existe un dt_3 tel que $dt_1 \longrightarrow *dt_3$ et $dt_2 \longrightarrow *dt_3$. Dans un système satisfaisant la propriété de confluence, la compatibilité est une relation d'équivalence.[5]

Définition 2.3 : Domaine de données

un dommaine de données est l'ensemble de valeures concrètes d'un type de données tels que :
- les châines de caractères alphanumériques (String,char,varchar)
- les types numériques (Integer,number,real)
- les dates
- les booléens
- les listes et les intervalles de valeurs

un domaine de données st identifée par un nom , soit D le nom attribué à un domaine le nom de domaine est alors un châine de caractères.Un tel nom n'est pas toujours significatif de sens de la donnée hors des personnes ayant présidé à leur création. [4]

exemple 2.1 : Domaine de données

- DomaineName20 est le nom donné au type String de 20 caractères.
- CityName30 est le nom donné au type String de 30 caractères.
- Age est le nom donné au type Integer de valeures comprises entre 0 et 150.

2.1.1 Définition 2.4 : Une source de données

Une source S est interprétée par l'ensemble $||T||$ de tous les$\{C_1 \dots C_n\}$ tuples définis sur $\prod_i^n = 1||D_n||$.

$||T||$ est l'ensemble des fonctions avec t: $\{C1...Cn\} \longrightarrow \prod_i^n = 1||D_n||$ tel que $t(C_i)$ noté $t.C_i$ est un élément de $||D_i||$.
Chaque élément t de $||T||$ est un tuple (enregistrement, record, ligne) de la source Set chaque $t.C_i$ est une valeur de la colonne dans t, que l'on notera aussi v.

2.1.2 Définition 2.5 : Schéma d'une source de données

Un schéma est une définition de la structure complète d'une source de données (Menard, 2008). Un schéma décrit toutes les propriétés d'une source : les noms des colonnes (attributs), les domaines syntaxiques de chacune (types de donnée) et éventuellement des contraintes et des commentaires sur certaines d'entre elles.
Les contraintes (Constraint) définies sur les colonnes expriment des règles sémantiques. Elles peuvent être de plusieurs types : clé primaire (Primary key), unique (Unique), clé étrangère (Foreign key), appartenance à un intervalle ou à une liste (Check), vérification de certaines règles de gestion et des déclencheurs (Triggers). Soit $Dom = \{D_1, D_2, \ldots, D_n\}$ l'ensemble des domaines des$C_i, i = 1; n.D_i$ présente le domaine syntaxique (type de données).
Soit$K = \{K_1, K_2, ..., K_n\}$ l'ensemble des contraintes sur les$C_i, i = 1; n.K$ présente le domaine sémantique d'une colonne.
Soit $Com = \{Com_1, Com_2, ..., Com_n\}$ l'ensemble des commentaires sur les $C_i, i = 1; n$.
On note un schéma
$S\{C_1: \{D_1, K_1, Com_1\}, C_2: \{D_2, K_2, Com_2\}, \cdots, C_n: \{D_n, K_n, Com_n\}\}$.
Par abus de langage on utilise la notation $S\{C_1, C_2, ..., C_n\}$ sachant que les domaines, les contraintes et les commentaires sont implicites, ou $S(C)$.
Une colonne Ci (un attribut) de S sera désigné par $S.C_i$ [5].

Exemple 2.2 : Exemples de schémas

$S1(FName, DateBirth, Contry)$.$S1$ est une source de données définie avec seulement les noms des attributs. C'est le format le plus souvent utilisé lors de la définition d'une source.
$0S2(FName, BirthDate: \{date, > 01/01/1980, //Dateofbirth\}, Country)$. $S2$ est définie avec des noms de colonnes significatifs et une contrainte et un commentaire pour la colonne BirthDate.
$S3(ABS1: \{string, cléprimaire\}, ABS2: \{string\}, ABS3: \{integer\}, ABS4: \{string\})$. $S3$ est une source avec des noms de colonnes ambiguës (meaningless).
Cependant, le type de données est présent pour chaque colonne et une contrainte est définie sur le premier attribut.
Notons que deux catégories de sources de données existent. Les sources qui sont accompagnées du schéma descriptif. Celui-ci peut être facilement compréhensible ou encore prêter à confusion.

Exemple 2.3 : Exemple d'une source où les noms des colonnes ne veulent rien dire

S				
	ABS1	ABS2	ABS3	ABS4
	string	string	integer	string
	clé primaire	??	??	??
	//	//	//	//
t1	LeBon Adam	0653545577	1000	
t2	LeBon A.	0653545555	2000	
t3	Lui Clémence	0607080911	1000	
t4	LeBon Adam	adam@yahoo.fr		Royaume-Uni
t5	LeBon A.	-		RoyaumeUni
t6	Lui Clémence	clui@gmail.com		France
t7	Traifor Eve	traifor@up13.fr		Chine

FIGURE 2.1 – Exemple de source S avec schéma

Le symbole " ? ?" est utilisé pour montrer l'absence de types de données ou de contraintes sur les colonnes. De même le symbole "//" montre l'absence de commentaires.
Les sources de données (sous forme d'un fichier plat (texte, csv)), peuvent ne pas être accompagnées d'un schéma descriptif (meaningless schema) (?).

Exemple 2.4 : Union des données sans schéma descriptif

S (TABLE 2.1) est le résultat de l'intégration de deux sources de données (française et anglaise). La source S peut être vue comme un ensemble de chaînes de caractères. Chaque tuple correspond à une ligne. Cette dernière contient plusieurs colonnes séparées par un point virgule.

S1 : Format CSV origine France
LeBon ; Adam ; 0653545577 ; adam@yahoo.fr ; M ; Mr ; Londres ; Royaume-Uni ; - ;1000 ; 31/03/2012 ; 0123435433
LeBon ; A. ; 0653545555 ; - ; M ; M. ; London ; Royaume-Uni ; - ; 2000 ; 31/03/2012 ;0123435432
Londres ;- ; - ; - ; F ; Mme ; Londres ; Royaume-Uni ; - ; 1000 ; 1-3-2002 ; 0411223344
Lui ; Clémence ; 0607080911 ; clui@gmail.com ; - ; Mme ; Epinay \ seine ; France ; - ; 02 mars 2014 ; 0120000022
Adamsss ; LeBon ; 0653545555 ; - ; - ; - ; - ; - ;- ; 31/3/2012 ;-
Lui ; Clémence ; 0607080911 ; clui@gmail.com ; F ; - ; Epinay sur seine ; France ; - ;2/3/2014 ;0120000022
Saint ; R. ; 0708091122 ; www.saint.fr ; M ; M. ; Epinay Villetaneuse ; Frence ; - ;3000 ;-
Tunsi ; Rahma ; - ;- ; - ; Mme ; Epinay sur seine ; France ; - ;1000 ; 31-11-2014 ; 0965321876
Riche ; Emir ; - ;- ; - ; M. ; Qatar ;- ; - ;200000000 ; 11/2/1955 ;-
Traifor ; Eve ; 0666622223 ; traifor@up13.fr ; F ; Mme ; Pékin ; Chine ; - ;1000 ; 30-2-2014 ; 0123987654
LeBon ; Adem ; - ; ada@obsolete.uk ; M ; Londre ; - ; - ;- ;-
S2 : Format CSV origine Angleterre
LeBon ; Adam ; 0653545577 ; adam@yahoo.fr ; M ; Mr ; London ; United-Kingdom ; - ;1000 ; 31/03/2012 ; 0123435433
LeBon ; Adam ; 0653545577 ; adam@yahoo.fr ; M ; Mr ; London ; United-Kingdom ; Europe ;1000 ; 31/03/2012 ; www.lebon.fr
LeBon ; A. ; 0653545555 ; - ; Male ; Mr ; London ; United-Kingdom ; Europe ;2000 ; 31/03/2012
London ; - ; 0766554425 ; - ; M ; Mr ; London ; United-Kingdom ; - ; - ; 11-12-1998
LeBon ; - ; 0653545577 ; adam@yahoo.fr ; 1 ; M. ; London ; UK ; - ;- ;-
Traifor ; Eve ; 0666622223 ; traifor@up13.fr ; Female ; Mrs ; Beijing ; China ; Asia ; 1000 ; 02/29/2014 ; 0123987654
Lebon ; Adel ; 0653545599 ; alebon@up13.fr ; M ;- ; Paris ; France ; Europe ; - ;45
Paris ; H. ; 0607080911 ; paris@live.com ; 0 ; Mrs ; LA ; USA ; America ;10000 ; 23-10-1992 ; 0987654321
Correia ; Sebastiao ; - ; scorreia@talend.com ; M ; - ; Suresnes ; - ; - ;- ; 9/30/2007 ; -

TABLE 2.1 – Exemple de source S avec schéma

2.2 Les différentes types des anomalies

Dans l'alittérature, il existe plusieurs travaux de recherche qui visent l'identification de différentes anomalies sur les données et sur leurs schémas. Ces anomalies peuvent être classées en deux groupes : les anomalies dans les métadonnées et celles dans les données.[4]

2.2.1 Anomalies dans les métadonnées :

Les anomalies dans les métadonnées sont diverses et sont causées par différents facteurs. On présente ci-dessous une liste non exhaustive de ces anomalies ainsi que leurs causes.
Rappelons que les métadonnées constituent tout ce qui est descriptif de données à savoir les noms des objets, leurs domaines de définition syntaxique (type de données : string, char, date, integer, number, real, boolean), leurs domaines de définition sémantiques (les contraintes), les commentaires et certaines analyses.
Les métadonnées sont souvent négligées, pourtant ces informations sont indispensables à une bonne compréhension du contenu sémantique des données et leurs contextes d'utilisation.
Parmi les anomalies constatées, on peut citer par exemple :

— Les noms des objets sont souvent non significatifs. Les noms des sources de données (table ou fichier) ne reflètent pas forcement leur contenu. Les noms utilisés dans le schéma descriptif, s'ils existent, de la source sont insignifiants tels que NP, Pnum ou Id. Ces derniers posent le problème de synonymie et d'homonymie.

— Les domaines de définition syntaxique des données ne sont pas suffisamment précis. Les

types de données (string, char, date, integer, number, real, boolean) ne suffisent pas pour comprendre la sémantique des données. Par exemple, les noms des personnes (définis sur des chaînes de caractères) ne sont pas comparables sémantiquement à des chaînes de caractères qui représentent les noms des entreprises.

— Les contraintes ne sont pas définies ou sont désactivées. Elles sont désactivées souvent pour des problèmes de performance ou parce que temporairement on a besoin de ne pas respecter ces contraintes. Par exemple, en l'absence du type intervalle, des valeurs numériques peuvent être aberrantes.

— Les commentaires ne sont pas maintenus à jour ou ne sont pas renseignés à cause de la fainéantise des utilisateurs ou bien parce que l'outil n'impose pas de les mettre.

Certaines entreprises définissent les bonnes pratiques, cependant elles ne sont pas souvent respectées à cause de diverses raisons. D'une part, les contraintes système imposées par les outils limitent à 30 caractères la taille des noms des objets, des noms courts sont requis. D'autre part, les règles de nommage imposées par l'entreprise ne permettent pas des noms significatifs.
. Ce qui complique d'avantage la compréhension du contenu et à fortiori toute manipulation telle que le processus d'intégration. Les anomalies dans les métadonnées, peuvent impacter donc toute opération sur les schémas des données et par conséquent les données elles-mêmes.
Il faudrait : (i) comprendre le sens de chaque colonne et de la source elle-même ; (ii) détecter les anomalies éventuelles au sein de chaque colonne et entre certaines d'entre elles ; (iii) et enfin corriger certaines de ces anomalies. L'exemple ci-dessous explicite ces problèmes.
Par exemple, lors d'une union, la compatibilité entre schémas de données ne peut être définie comme suit :
"deux schémas sont compatibles si et seulement si ils ont le même nombre de colonnes et toutes les colonnes sont définies deux à deux sur le même domaine syntaxique" (Codd, 1970).
Ainsi les opérations ensemblistes telles que l'union, l'intersection ou la différence devraient être basées sur une compatibilité plus riche sémantiquement. Les colonnes devraient être définies aussi sur le même domaine sémantique. Il en est de même des opérations binaires telles que la jointure et ses dérivés.
Force est de constater que les outils SGBD (Système de Gestion des Bases de Données) et ETL ne respectent pas cette compatibilité sémantique et n'assistent pas les utilisateurs.
Les exemples ci-dessous, traduisent certains conflits d'intégration de données dûs aux anomalies dans les schémas descriptifs (métadonnées).
Les définitions du nom de la donnée destination ainsi que de son type sont données manuellement. Les outils ETL ne font pas de recommandation et supposent donc une certaine expertise de l'utilisateur[5].

Exemple 2.5 : Intégration des données (Fusion-Union) (Contrainte non définie)

Etant donné deux sources de données $S1$ et $S2$. $S1$ regroupe la liste des clients. Elle est gérée par un SGBD. La source $S2$ représente la liste des patients. Elle est donnée sous forme d'un fichier plat (de format texte ou csv).
Les schémas des deux sources sont :

$S1(NomP: \{String(20)\}, Tel: \{String(10)\}, CA: Integer)$
$S2(NP: \{String\}, Tel: \{String\}, Pays: \{String\})$
Le résultat de la "Fusion-Union" est : R1= Fusion-Union(S1, S2) .
R1(NomP : $\{String\}, Téléphone: \{String\}, CA: \{Integer\}, Pays: \{String\}$) L'intégration de ces deux sources, peut engendrer différentes anomalies. La structure de la fusion est décrite par l'utilisateur. Le nom et le type de chaque colonne résultat sont renseignés. $R1.NomP$ sera construit à partir de $Client.NomP$ et de Patient.NP. Le type de cette colonne est celui qui doit couvrir $String(20)$ et $String$. Il faut remarquer que les données peuvent être tronquées si le type résultat n'est pas adéquat.
Il en est de même pour la colonne $R1.Téléphone$ (respectivement $R1.CA$ et $R1.Pays$) qui résulte de $Client.Tel$ et de $Patient.Tel$ (respectivement$Client.CA$ et $Patient.Pays$) .
Des valeurs nulles seront générées à cause du fait que l'information sur les données Pays n'existe pas dans la première source et la donnée sur le chiffre d'affaires CA non plus.

S1 : Client			
	NomP	Tel	CA
	String(20)	String(10)	Integer
	? ?	? ?	? ?
	//	//	//
t1	LeBon Adam	0653545577	1000
t2	LeBon A.	0653545555	2000
t3	Lui Clémence	0607080911	1000

S2 : Patient			
	NP	Tel	Pays
	String	String	String
	? ?	? ?	? ?
	//	//	//
t1'	LeBon Adam	adam@yahoo.fr	Royaume-Uni
t2'	LeBon A.	-	RoyaumeUni
t3'	Lui Clémence	clui@gmail.com	France
t4'	Traifor Eve	traifor@up13.fr	Chine

R1 : Clients (Résultat de Fusion-Union de Client et Patient				
	NomP	Téléphone	CA	Pays
	String	String	Integer	String
	? ?	? ?	? ?	? ?
	//	//	//	//
t1	LeBon Adam	0653545577	1000	
t2	LeBon A.	0653545555	2000	
t3	Lui Clémence	0607080911	1000	
t4	LeBon Adam	adam@yahoo.fr		Royaume-Uni
t5	LeBon A.	-		RoyaumeUni
t6	Lui Clémence	clui@gmail.com		France
t7	Traifor Eve	traifor@up13.fr		Chine

TABLE 2.2 – Intégration (Fusion-Union) des deux sources Client et Patient avec un ETL

Il est clair que les choix d'intégration, basés sur la synonymie/homonymie des colonnes de noms $S1.Tel$ et $S2.Tel$, introduit des anomalies dans le résultat. Des adresses mails sont présentes dans

la colonne nouvellement baptisée $R1.Téléphone$. L'utilisateur n'est pas assisté dans le processus d'intégration. Plusieurs questions se posent telles que :

— "Fallait-il comprendre les noms des colonnes $S1.TeletS2.Tel$ comme étant un téléphone ou un mail ?". D'une part, les valeurs n'étant pas affichées à l'écran, et d'autre part, aucune information n'est renseignée sur la sémantique des données. Le type String, à lui tout seul, ne permet pas d'assurer la cohérence sémantique des deux colonnes.
— "Quelle est la structure du résultat : s'agit-il d'une union ou d'une jointure et quel type de jointure (gauche, droite, externe, interne) ?".

Les outils devraient reconnaître la sémantique des deux colonnes avant la réalisation de l'opération. La sémantique de la première colonne $S1.Tel$ devrait être des numéros de téléphone, alors que celle de la deuxième colonne $S2.Tel$ devrait être des adresses mail. Les outils devraient alerter cette incompatibilité sémantique et interdire même l'opération, ce qui n'est pas toujours le cas de nos jours.

Exemple 2.6 : Intégration des données (Fusion-Jointure-Gauche)

La jointure de ces deux sources de données produit des résultats différents. Par exemple, une jointure à gauche (Codd, 1970) renvoie le résultat présenté dans la table 2.4. Cette jointure donne les personnes existantes dans les deux sources en enrichissant avec l'attribut Pays. Cependant, l'information sur le mail est perdue. $R2 = Fusion - Left - Join(S1, S2, S1.Tel = S2.Tel)$.
$R2(NomP: \{String\}, Tel: \{String\}, CA: \{Integer\}, Pays: \{String\})$

R2 : Clients (Résultat de la Fusion-Jointure-Gauche de Client et Patient)				
	NomP	Tel	CA	Pays
	String	String	Integer	String
	??	??	??	??
	//	//	//	//
t1	LeBon Adam	0653545577	1000	Royaume-Uni
t2	LeBon A.	0653545555	2000	RoyaumeUni
t3	Lui Clémence	0607080911	1000	France

FIGURE 2.2 – Intégration (Fusion-Jointure-Gauche) des deux sources Client et Patient avec un ETL

La jointure n'aurait pas dû être acceptée. L'utilisateur devrait être alerté lors du choix des colonnes de jointure.
Pour une meilleure intégration sémantique de données, la condition de jointure devrait porter sur des colonnes sémantiquement équivalentes.

Exemple 2.7 : Intégration des données (Fusion-Union) (Contrainte définie)

*R*est le résultat de l'intégration de deux sources de données (TABLE 2.3). S1 regroupe des données du mois de janvier sur la propreté et la température de la ville de Paris. $S2$ contient celles

du mois de février. $S1$ est une version francophone et $S2$ est version anglo-saxonne.
$S1(Note: \{[0 \dots 20\}, Température: \{[-40 \cdots 50], C\})$
La colonne $S1.Note$ représente l'ensemble des notes de satisfaction à propos de la propreté de la ville. Ces notes sont données sur une échelle de0à20. Alors que $S1.Température$ représente la température moyenne, en degré Celsius (C), de la ville de Paris pour certains jours du mois de janvier. La colonne$S2.Note$ pourrait représenter la même information que $S1.Note$. Elle est mal définie (de 0à 10 ou de 0 à 100). La valeur 100 est une donnée douteuse. $S2.Temp$ indique certaines moyennes de la température du mois de février de la ville de Paris, elles sont probablement en Fahrenheit (F) et l'intervalle de définition n'est pas précisée.
$S2(Note, Temp)$

S1	
Note	Température
??	??
[0..20]	[-40..50]
//	°C
12 ; -5 °C	
10 ; -1 °C	
8 ; 0 °C	
18 ; 6 °C	

S2	
Note	Temp
??	??
??	??
//	//
100 ; 30 °F	
9 ;40 °F	
5 ;40 °F	
3 ;20 °F	

a) R =Fusion-Union(S1,S2)	
Note	Température
Integer	Integer
??	??
//	//
12 ;-5 °C	
10 ; -1 °C	
8 ;0 °C	
18 ;6 °C	
3 ; 30 °F	
9 ; 40 °F	
5 ; 40 °F	
7 ; 20 °F	

b) R=Fusion-Union(S1,Transformation(S2))	
Note	Température
Integer	Integer
[0..20]	[-40..50]
//	°C
12 ; -5 °C	
10 ; -1 °C	
8 ; 0 °C	
18 ; 6 °C	
6 ; -1,11 °C	
18 ; 4,44 °C	
10 ; 4,44 °C	
14 ; -6,66 °C	

TABLE 2.3 – Intégration (Intégration des deux sources de données avec un ETL

Dans le premier cas de figure (cas (a) TABLE 2.3) , le résultat de l'intégration (Fusion-Union) est erroné. Les colonnes $S1$.Température et$S2.Temp$ ne sont pas homogènes car elles ne sont pas sémantiquement équivalentes. Les outils devraient détecter que la sémantique de la donnée est une

température dans les deux cas, alors que l'unité de mesure n'est pas la même.
$R = Fusion - Union(S1, S2)$
$R(Note$: $\{Integer\}$, Température : $Integer)$
Plusieurs questions se posent telles que : $transformations faut-il appliquer et sur quelles colonnes$?.
$C'est le deuxième cas(cas($ b $)TABLE2.3)$.
R=Fusion-Union(S1,Transformation(S2))
R(Note : $\{Integer, [0 \dots 20\}, Température$: $\{Integer, [-40 \dots 50], C\}$)
Dans la mesure où $S1.Température$ et $S2.Temp$ ont le même sens, leur intégration devrait homogénéiser l'unité de mesure (C, F) (cas (b) table2.5), sachant que 50 degrés Fahrenheit $(50F) = 10$ degrés Celsius $(10C)$.
Les valeurs de la colonne $S2.Note$ nécessite une transformation dans le domaine sémantique de la source$S1[0 \dots 20]$(cas (b) table 2.5).
Plusieurs questions se posent telles que : "Quelle est la transformation nécessaire et y-a-t-il des valeurs aberrantes ?".
Il faut remarquer que des analyses sur les données devraient enrichir les définitions syntaxiques et sémantiques des données avant de réaliser toute manipulation de données.
Les données étant hétérogènes et mal renseignées dès le départ, le processus d'intégration peut causer la dégradation des données. L'absence de référentiels et le manque de la sémantique peuvent introduire des valeurs nulles, des incohérences dans les données, voire des contradictions.
Une étude comparative sur les différents ETL commerciaux existants tels que Talend Data Integration (TDI) et Pentaho Data Integration (PDI), nous a permis de mettre l'accent sur ces difficultés d'intégration des sources de données hétérogènes. L'utilisateur ne bénéfice d'aucune recommandation ni assistance. Il en est de même des outils de gestion de la qualité (Talend Data Quality, DataCleaner, Datiris), des SGBD (Oracle) et des outils d'aide à la conception de BD (PowerAMC) .
En effet, lors de l'intégration des données en provenance de plusieurs sources hétérogènes, un grand nombre de types de conflits structurels peut survenir.
Un même nom peut être donné à deux objets différents dans chacune des sources (homonymes). Des appellations différentes d'un même objet peuvent exister (synonymes).
De surcroit, les contraintes définies sur les données peuvent ne pas exister. En effet, chaque donnée (colonne) définie syntaxiquement par un type (string, char, date, integer, number, real, boolean), devrait être définie par un domaine sémantique exprimé généralement par une contrainte telles que :

- L'appartenance à un intervalle de valeurs ou un ensemble de valeurs énumérées,
- l'unicité de valeurs pour une clé primaire (Primary Key) ou la contrainte unique (Unique).
- le sens sémantique de la donnée (une chaîne de caratères qui définit le nom d'un client est différente de celle qui désigne le nom d'une entreprise).
- les dépendances entre les données (le sex dépend de la civilité).

Signalons par ailleurs qu'aucune vision globale entre les colonnes n'est prise en compte. Les dépendances éventuelles entre les colonnes devront être étudiées afin d'éviter certaines incohérences. Il est en est de même concernant le rapprochement des lignes résultats et le traitement en général des redondances. Ce qui constitue l'objet du paragraphe suivant qui est les anomalies dans les

données[5].

2.2.2 Anomalies dans les données

Les anomalies dans les données sont très variées et sont causées par plusieurs facteurs.
Les données étant de plusieurs types ($string, char, date, integer, number, real, boolean$), elles peuvent être représentées dans plusieurs formats plus au moins équivalents.
Les données de type chaîne de caractères ($string$) peuvent représenter dans la réalité plusieurs sous-catégories de données telles que les données alphabétiques, numériques ou dates. Il convient donc de différencier ***les catégories*** et ***les sous catégories***, non seulement dans les types chaînes de caractères, mais aussi les autres types.
Les anomalies dans les données peuvent être perçues selon la classification cidessous. La double validité syntaxico-sémantique doit être prise en compte.
La codification des données Le format de données de type date devrait être validé par des expressions régulières jj/mm/aaaa, mm/jj/aaaa, aaaa/mm/jj. Le contenu ne devrait pas être aberrant tel que la date de naissance d'un client est le 14 janvier 3011 par rapport à la date système (15 décembre 2014).
Le format de données de type numérique devrait être précisé moyennant la taille de la partie entière et celle de la partie décimale.
Les chaînes de caractères telles que$-40C, 60$ oF,$100Kg, 1$ TO devront faire partie des données numériques. Elles sont présentées dans un format (unité de mesure). Une conversion StringToNumber et des expressions régulières sont nécessaires pour définir et contrôler au mieux les données.
Les chaînes de caractères doivent respecter les normes et standards existants tels que les normes ISO (International Standard Organisation), les codes pays, les codes postaux, le formatage des adresses, les dates, le numéro IBAN (International Bank Account Number), le numéro de SIREN (Système d'Identification du Répertoire des Entreprises).
Les chaînes de caractères "Epinay sur seine" et "Epinay/seine" doivent être unifiées.
La codification des données de type caractère selon le sens de celles-ci doit être également définie moyennant des expressions régulières. Par exemple, l'énumération des données se fait par une liste (sexe : $\{Homme, Femme, Male, Female, M, F, 0, 1\}$; size : $\{S, M, L, XL, XXL\}$; Blood group : $\{A+, B+, AB\}$)
L'encodage des caractères peut donner un sens sémantique à la donnée. Par exemple, les caractères spéciaux peuvent être interdits pour les noms des objets ("Fuentes HernÃ©ndez", EID^2).

Les valeurs aberrantes (Outliers) (Knox and Ng, 1998) qui présentent des écarts par rapport à la moyenne ou au modèle. Le problème consiste à définir un seuil à partir duquel les mesures peuvent être rejetées. La majorité des notes de propreté de la ville de Paris appartiennent à l'intervalle de définition sauf exception, la donnée "100", ce qui peut correspondre à une valeur aberrante.

Les valeurs nulles (Null Values) (Boufarès, 1983), (Berti-Équille, 2012) constituent une des causes principales des problèmes dans les rapports décisionnels. Plusieurs interprétations sont possibles pour celles-ci :

— valeurs applicables et non renseignées : la "ville de naissance" de Charlemagne ?
— valeurs non applicables (non signifiantes) : "nom de jeune fille" pour un homme !

Les valeurs par défaut (Default Values) proches des valeurs manquantes, sont plus difficiles à détecter. Par exemple, étant donnée la liste des villes pour lesquelles on affiche la température de saison, les valeurs dans le tableau ci-dessous (TABLE 2.4) donnent la température pour les villes de Paris et Londres. Une valeur par défaut, donnée à la ville de Londres, peut être mal interprétée. En effet, le 0 est-il la valeur par défaut (température non renseignée) ou la valeur après mise à jour ($température = 0$)?. S(NomVille : String, Température : $Integer, 0pardéfaut, C$) [5]

Exemple 2.8 : Les valeurs par défaut

S	
NomVille	Température en °C
String	Integer (0 par défaut)
??	??
//	//
Paris	5,4
Londres	0

TABLE 2.4 – – Les valeurs par défaut

L'obsolescence des données (Peralta, 2006) : qui peut être mesurée par deux facteurs. Le premier facteur de résistance des données par rapport aux changements à partir de la date d'extraction à la date de livraison. Le deuxième facteur exprime l'ancienneté des données : c'est l'écart entre la création et la mise à jour des données par rapport à la livraison sans se soucier de la date de création.
Il faut remarquer que les anomalies citées ci-dessus ne portent que sur une colonne à la fois. Il existe cependant, d'autres anomalies portant sur plusieurs colonnes en même temps à savoir les liens entre colonnes pour exprimer les dépendances entre elles et les lignes en doubles ou similaires.

Les dépendances (Simonenko and Novelli, 2012), (Dallachiesa et al., 2013) entre les colonnes telles que les dépendances fonctionnelles constituent un autre volet d'anomalies. Non seulement il faut les découvrir mais aussi les vérifier. Les contraintes définies sur le schéma conceptuel ne suffissent pas pour éviter les données erronées dans la base de données. La détection des redondances inutiles n'est qu'une partie du problème. La question sémantique reste totalement ouverte pour savoir quelles sont les lignes à éliminer.

Les redondances (doubles et similaires) entre les lignes doivent être détectées et corrigées. Ce problème est appelé aussi "Record Linkage", "Duplicates Data" (Hernandez and Stolfo, 1995), (Sarawagi and Bhamidipaty, 2002), (Koudas et al., 2006), (Benjalloun et al., 2009), (Boufarès et al., 2012a), (Boufarès et al., 2012b), (Boufarès et al., 2013). La similarité entre les données nécessite un

calcul de similarité entre les données. Différentes méthodes de mesure existent dans la littérature (Bilenko and Mooney, 2003), (Hernandez and Stolfo, 2004), (Cohen and Richman, 2004), (Koudas et al., 2006), (Winkler, 2006), (Benjalloun et al., 2009), (Berti-Équille, 2007). Les chaînes de caractères "'Le Bon Adam" et "Le Bon Adan" sont proches lexicographiquement. Elles sont donc équivalentes syntaxiquement et sémantiquement. Alors que les chaînes "London" et "Londres" ainsi que les chaînes "Pékin" et "Beijing" sont éloignées lexicographiquement. Elles sont syntaxiquement différentes et sémantiquement équivalentes. Ces quatre dernières chaînes de caractères appartiennent à la ***catégorie sémantique*** City. Notons que les méthodes de mesure de similarité actuelles sont pauvres en sémantique.

Les principales causes des anomalies dans les données résident principalement lors de la saisie manuelle ou de la génération automatique et au cours des transformations et agrégations. Les utilisateurs ne sont pas vraiment assistés dans cette tâche. En effet, d'une part, les formulaires de saisie sont très mal conçus : peu ou pas d'utilisation de référentiels afin d'éviter par exemple les erreurs typographiques, les abréviations et les mélange entre colonnes. La définition syntaxique des données (type de données), n'est pas toujours accompagnée des contraintes (constraints) ni des déclencheurs (triggers) permettant de filtrer les données.
En effet, les formulaires Web, source d'anomalies, contiennent d'une part les noms des colonnes à remplir et d'autre part, les données renseignées par l'utilisateur.
Les noms de colonnes peuvent être la source du problème dans la mesure où ils prêtent à confusion ou ils sont mal compris[5].

Exemple 2.9 : Les anomalies dans les formulaires Web français

Nom du champ	Donnée-Saisie	Nom du champ	Donnée-Saisie
Civilité	**Mme**	Email	**adem.lebon**
Prénom	**Adem**	Adresse	**18 rue Carnot**
Nom	**LeBon**	Adresse2	
Sexe	**Masculin**	Ville	**Paris**
Téléphone1	**0155555599**	Code Postal	**92150**
Téléphone2	**0653545599**	Pays	**France**
Fax	**0148904890**		

TABLE 2.5 – Formulaire Web source d'anomalies

Les champs *Téléphone*1 et *Téléphone*2 prêtent à confusion. Il est en de même des champs *Adresse* et *Adresse*2.
Les seules valeurs admises dans civilité devront appartenir à (M., Mme). Le champ sexe est dépendant de la civilité. Il devrait être déduit automatiquement. Le champ pays doit être renseigné avant la ville. Celui-ci devrait servir pour établir automatiquement les contrôles nécessaires du téléphone et fax.

De nombreuses questions relatives à la qualité de données doivent être posées, tout au long du cycle de vie de la donnée et surtout au niveau du processus d'intégration, sur :

1. les contraintes sur les données
2. la compatibilité des types syntaxiques de données,
3. la compatibilité de types sémantiques des données,
4. la transformation, la correction et la validation des données,
5. la déduplication des données et l'élimination des redondances.
6. comment le savoir ?.

Pour aider à résoudre ces problèmes, l'ajout de la sémantique aux métadonnées peut être une solution afin de comprendre les données, détecter et localiser les anomalies existantes. Ainsi, des actions automatiques de correction et de nettoyage peuvent être proposées.
Dans ce qui suit, un parcours des différents travaux existants va nous permettre d'établir un bilan sur les démarches proposées et prouver l'originalité de notre contribution.
Rappelons qu'à l'aire du BigData, les sources de données à manipuler n'ont pas forcément un descriptif (meaningless schema). Ainsi, une étape supplémentaire de reconstruction du schéma à partir de l'analyse des données nous semble déterminante pour aborder l'approche de nettoyage de données. C'est ce qui caractérise notre approche.

2.3 Traitement des anomalies

Le présent paragraphe exposera l'essentiel des travaux présentés dans la littérature sur sur le rapprochement de schémas, la détection et la correction des anomalies.
La manipulation des sources de données dans le but de les explorer et d'en extraire des connaissances, nécessite donc de reconnaître le sens de chaque colonne (reconstruction sémantique) et éventuellement le lien entre elles afin de remédier aux anomalies. Le lien entre les colonnes pourrait donner une idée sur le contenu globale de la source (le concept).

2.3.1 Rapprochement de schémas

Le rapprochement de schémas (Schema Matching) est le processus d'identification des objets sémantiquement similaires (figure ? ?) (Rahm and Bernstein, 2001). Le principe consiste à identifier et caractériser les correspondances entre deux schémas qui existent. Ils se basent essentiellement sur la comparaison lexicographique des noms des colonnes.
Le rapprochement de schéma permet de faciliter le processus d'intégration et la création d'un entrepôt de données à cause de l'hétérogénéité des sources.
Différentes méthodes existent dans la littérature (Rahm and Bernstein, 2001), (Sais, 2007), (Do and Rahm, 2002), (Madhavan et al., 2001), (Castano et al., 2001), (Li and Clifton, 2000) (FIGURE 2.4).

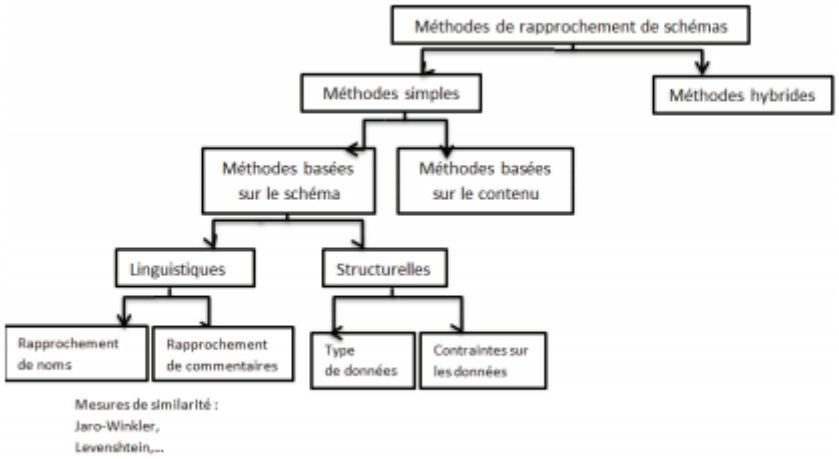


FIGURE 2.3 – Approches de rapprochement de schéma

Parmi ces méthodes, on peut citer :

Approche linguistique (non basée sur la définition syntaxique et sémantique de la colonne) (Glenn and Sethi, 2001) : elle utilise les noms des colonnes et leurs commentaires pour trouver des éléments sémantiquement similaires. Deux niveaux sont définis :

- rapprochement de noms : calculer la similarité entre eux (en prenant en compte les synonymes, les homonymes 5 , les hyperonymes 6 existants dans des dictionnaires de données tel que WordNet 7) avec les mesures de distance d'édition telles que Levenshtein, Jaro-winkler et la mesure phonétique Soundex (Bilenko and Mooney, 2003).
- . rapprochement des commentaires : exploiter la sémantique des commentaires associés à chaque colonne, afin de trouver une similarité entre les colonnes des deux schémas.

Exemple 2.10 : Rapprochement des commentaires

Soit $S1$ et$S2$ deux sources de données. Des commentaires ont été renseignés sur les deux colonnes $S1.BirthD$ et $S2.DateB$. On rapproche les deux commentaires (FIGURE 2.4) afin de détecter que ces deux colonnes sont similaires.

S1			≈	S2		
Name	Sex	BirthD		DateB	City	Country
??	??	??		??	??	??
??	??	??		??	??	??
//	//	//BirthDate	⟷	//Date of birth	//	//

FIGURE 2.4 – Rapprochement de commentaires de deux attributs

Approche structurelle basée sur la définition des colonnes (type et contraintes) (Larson et al., 1989) : la similarité est calculée à la base de la correspondance entre :

— les types de données : la notion de compatibilité est utilisée entre les types pour le rapprochement. Par exemple pour une chaîne de caractères, les types existants dans (FIGURE 2.6) sont équivalents

Type de données compatibles
String, varchar, char
Number, integer, real
Date
Boolean

TABLE 2.6 – Compatibilité des types de données

— les contraintes sur les données : rapprocher les contraintes définies sur chaque colonne. Par exemple, rapprocher la contrainte clé primaire à la contrainte Unique.

Exemple 2.11 : Exemple de rapprochement en utilisant les contraintes ainsi que le type des données

Soit deux schémas $S1$ et $S2$ avec (FIGURE 2.5) : $S1(NomPrénom: \{varchar(20)\}, CA: \{integer\}, Civilité: \{varchar(10)\}, EDate: \{date, > 01/01/1998\})$. $S2(NomP: \{string\}, CA: \{real\}, SDate1: \{date, > 01/01/1998\}, Sexe: \{string\}, CI: \{string\})$

S1			
NomPrénom	**CA**	**Civilité**	**EDate**
varchar(20)	integer	varchar(10)	date
??	??	??	>1/1/1998
//	//	//	//

S2			
NomP	**CA**	**Date1**	**Sexe**
string	real	date	string
??	??	>1/1/1998	??
//	//	//	//

FIGURE 2.5 – Rapprochement structurelle

$S1.CA$ et$S2.CA$ sont similaires puisqu'ils ont le même type de données (integer). $S1.EDate$ et $S2.Date1$ ont le même type "date" et vérifient la même contrainte ($> 01/01/1998$), donc ils sont proches.
Notons aussi que souvent les descriptifs et les contraintes sur les données sont peu ou mal définis. Peu d'utilisateurs remplissent les commentaires ou spécifient des contraintes sur les données.

Approche hybride (Doan et al., 2001) : celles-ci combinent les deux approches précédentes. En effet, ni la reconnaissance lexicographie du nom de la colonne ni les contraintes ne peut suffire à elles seules de rapprocher les schémas. L'approche linguistique combinée avec l'approche basée sur les contraintes permet une meilleure correspondance entre les deux schémas.

Exemple 2.12 : Approche hybride

Dans l'exemple précédent (exemple 2.10), les colonnes $S1.NomPrénom$et $S1.Civilité$ et $S2.NomP$, $S2.Sexe$ et $S2.CI$ ont le même type (string). Ici le type ne suffit pas pour décider de la correspon-

dance. Un rapprochement en fonction des noms peut être appliqué pour rapprocher $S1.NomPrénom$ à $S2.NomP$. Par contre, les autres colonnes $S1.Civilité$, $S2.Sexe$ et $S2.CI$ ne sont pas rapprochables.
Le rapprochement linguistique ne prend pas en compte la langue des colonnes. Tous les noms des colonnes sont des chaînes de caractères, sans aucune sémantique, à rapprocher.
L'originalité de notre approche est que nous construisions un schéma sémantique. Les colonnes du schéma sont bien définies. Chaque colonne a une catégorie et une sous-catégorie (telle que la langue). Donc c'est peu envisageable de rencontrer un nom colonne tel que CI.
La détection des relations entre colonnes pourra faciliter la détection des colonnes similaires. Dans l'exemple 2.10, il existe une dépendance entre la colonne S1.Civilité et $S2.Sexe$.

Approche basée sur le contenu en détectant les doublons d'une source (Bilke and Naumann, 2005), (Bilke et al., 2005). L'approche consiste à comparer le contenu de deux enregistrements et si les valeurs de deux colonnes sont similaires alors ces deux derniers sont rapprochables. Une matrice de similarité est calculée pour chaque paire d'enregistrements (TABLE 2.7).

		FirstName	**Phone**	**Sex**	**Civility**	**City**
		Jean	0666554426	M	Mister	Paris
FName	Paris	0	0	0	0	1
Ph	0666554425	0	0,96	0	0	0
S	M	0	0	1	0	0
Adr	Paris	0	0	0	0	1

TABLE 2.7 – Matrice de similarité pour une paire d'enregistrements

Une nouvelle matrice est proposée contenant la moyenne des similarité des doublons, pour les différentes colonnes (TABLE 2.8). A partir de cette matrice, le rapprochement de schémas est déduit.

	FirstName	**Phone**	**Sex**	**Civility**	**City**
FName	0	0	0	0	1
Ph	0	0,96	0	0	0
S	0	0	1	0	0
Adr	0	0	0	0	1

TABLE 2.8 – Matrice des moyennes de similarités

Notons que l'on peut rapprocher des colonnes différentes mais avec un contenu similaire. Par exemple, on rapproche les colonnes FName à City vue que la valeur "Paris" représente d'une part le nom d'une personne et d'autre part, le nom de la capitale de France.
Deux colonnes sont similaires à la même colonne. La valeur "Paris" apparaît dans les deux attributs.

Alignement des ontologies On retrouve le rapprochement de schémas dans le domaine des ontologies. Le concept "alignement des ontologies" est utilisé pour le rapprochement des ontologies.
Les travaux existants (Euzenat and Valtchev, 2006), (Shvaiko and Euzenat, 2013) définissent les mêmes méthodes de rapprochement de schémas (linguistique, structurelle, hybrides). Ils ajoutent cependant, des méthodes qui permettent d'exploiter la particularité des ontologies qui est les relations entre entités.
Le mot "entité" est utilisée dans le monde ontologie pour désigner les colonnes et les noms des sources.
Les liens entre entités (équivalence, transitivité, symétrie, disjonction) sont utilisés pour rapprocher les entités entre elles (Giunchiglia et al., 2005).
En conclusion, l'alignement en fonction des relations et des instances est très peu utilisé. La comparaison terminologique et structurelle des noms des objets prend le dessus sur les différentes approches (Shvaiko and Euzenat, 2013). Cependant, ces noms peuvent être peu significatifs ou contiennent peu de sémantique[5].

2.3.2 Détection des anomalies

Le ***profilages*** de données représente une étape primordiale pour la détection des anomalies dans les données sur une colonne. Le profilage est une collection de statistiques. Il consiste en une analyse exploratoire des données sur trois niveaux :

1. ***analyse*** analyse des colonnes indicateurs de fréquence, les valeurs nulles pour détecter des données douteuses comme les valeurs aberrantes,
2. analyse des dépendances (les dépendances fonctionnelles par exemple) ;
3. analyses des doublons et des similaires (Johnson and Dasu, 2001), (Berti-Équille, 2006).

Le profilage est très utilisé dans le monde industriel. Force est de constater que c'est une tâche très manuelle. Les utilisateurs ne bénéficient d'aucune assistance ni aide. L'utilisateur est censé connaitre la sémantique des colonnes et avoir une idée sur contenu de la source manupilée !
Une étude comparatif des trois outils open source existants : "Talend Data Quality (TDQ)", "DataCleaner (DC)" et "DatirisProfiler (DP)", nous permet d'avoir un aperçu sur les fonctionnalités proposées (TABLE 2.9).
Les statistiques simples comportent des indicateurs tels que le nombre de valeurs total, nombre de valeurs nulles, nombre de valeurs en doubles et autres statistiques appliquées sur tout type de données[5].

Analyses de colonnes (une à une)	Indicateurs	TDQ	DC	DP
Statistiques simples	Nombre total des valeurs	X	X	X
	Nombre de valeurs nulles	X	X	X
	Nombre de valeurs distinctes	X	X	X
	Nombre de valeurs doubles	X	X	X
	Table de fréquence	X		
Statistiques sur les chaînes	Longueur maximale des chaînes	X	X	X
	Longueur minimale des chaînes	X	X	X
	Longueur moyenne des chaînes	X	X	X
Statistiques sur les numériques	Valeur maximale des numériques	X	X	
	Valeur minimale des numériques	X	X	
	Moyenne des numériques	X	X	
	Écart type des numériques	X	X	
Statistiques sur les dates	Fréquence des années	X	X	
	Motif de fréquence de Date	X		
Format des chaînes	Motif de fréquence	X		
Nature des chaînes				
Langue des chaînes				
Nature de la langue des chaînes (Latin, Arabe)			X	
Nombre de chaînes valides syntaxique				
Nombre de chaînes valides sémantique				

TABLE 2.9 – Tableau comparatif des différents outils de profilage (analyse de colonnes)

Analyses de la source		TDQ	DC	DP
Détection des doublons et	Choix des attributs de déduplication	X		
	Choix des algorithmes de similarité	X		
	Choix des seuils de similarité	X		
Dépendances fonctionnelles	Dépendances simples	X		
	Dépendances multiples			
	Détection manuelle	X		
	Détection automatique			
Détection des anomalies syntaxiques				
Détection des anomalies sémantique				
Assistance utilisateur				
Recommandation				

TABLE 2.10 – Tableau comparatif des différents outils de profilage (analyse de ource)

Les outils de profilage existants sur le marché fournissent des résumés statistiques sur les données. Ces statistiques concernent exclusivement la syntaxe des données. Elles ne sont pas facilement interprétables et ne permettent pas de détecter les différentes anomalies dans une source de données. Des règles métiers sont à définir en fonction des besoins utilisateurs telles que :

- Une personne de sexe masculin, n'a pas de nom marital.
- La date de recrutement doit être inférieure à la date de démission.
- Deux attributs sont considérés comme dépendants l'un de l'autre si et seulement si les valeurs de deux colonnes vérifient la dépendance fonctionnelle à plus de 80.

Des seuils sont aussi à définir par les utilisateurs sur les indicateurs de l'analyse. Par exemple, on n'accepte pas plus de 40de valeurs nulles dans une colonne donnée.
Ces outils n'assistent pas donc l'utilisateur dans sa démarche, qui ne bénéfice d'aucune recommandation. C'est à lui seul d'analyser les résultats de profilage et d'en déduire les actions de nettoyage sur les données.
Les données ne sont pas catégorisées. Aucun indicateur n'existe pour déterminer la nature de la colonne traitée.
Les outils offrent des analyses statistiques sur les données pour chaque colonne et pour chaque type de donnée. Par contre, aucune analyse sémantique sur les données n'est effectuée.

Cependant il reste à se poser beaucoup de questions telles que : "Est ce que les résultats donnés par les analyses permettent de spécifier le type de traitement à proposer pour l'utilisateur ? ", "Est ce que les résultats d'analyses sont facilement interprétables ? ", "Comment peut-on déduire les attributs clés pour une élimination des doublons ? ".
Les contraintes définies éventuellement sur les données ne sont pas prise en compte. Il faudrait ajouter des vérifications sur ces contraintes tels que les check de SQL ou encore les triggers[5].

2.3.3 Correction des anomalies

Afin de remédier à certaines anomalies, différents travaux ont été proposés dans littérature. On s'intéresse dans ce qui suit aux deux actions de nettoyage : l'homogénéisation des données et la déduplication.
Une étude bibliographique sur la problématique d'élimination des doublons et similaires dans une source a été menée en profondeur.
L'élimination des similaires est le processus de comparaison de lignes d'une source de données afin de déterminer celles qui présentent le même objet du monde réel. Il est appelé encore le processus de dé-doublonnage (déduplication), de résolution d'entité (Entity resolution), de rapprochement de données (Entity Matching), de couplage d'enregistrements (record linkage) (Hernandez and Stolfo, 1995), (Sarawagi and Bhamidipaty, 2002), (Koudas et al., 2006), (Benjalloun et al., 2009), (Boufarès et al., 2012a), (Boufarès et al., 2012b), (Boufarès et al., 2013). Il consiste en deux étapes : la comparaison (Match) et la fusion (Merge).
(Boufarès et Aicha Bensalem) résumons les différentes étapes de l'algorithme d'élimination des doublons et similaires en cinq étapes (algo 1) :

Algorithm 1 Deduplication
Input : Data Source S
Output : Cleaned Target Data SC
Begin
Choose key attributes (deduplication attributes)
Choose similarity distance
Choose Match method
Choose Merge approach
Evaluate the deduplication rate
return SC
End Algorithm Deduplication

Choix des attributs de dédoublonnage

Peu de travaux portent sur le choix des attributs clés. Parmi ces travaux, citons celui (Fan et al., 2009) qui propose de choisir les attributs candidats pour le dédoublonnage en utilisant les relations de dépendances entre les attributs. Les outils de profilage ne permettent pas de bénéficier des résultats des analyses, à savoir de ne pas utiliser les colonnes contenant plusieurs valeurs nulles comme attribut clé.

Choix d'un algorithme de similarité

La similarité des données est définie comme étant la distance entre deux valeurs, de même type, de deux enregistrements (comme illustré dans la figure 2.4).

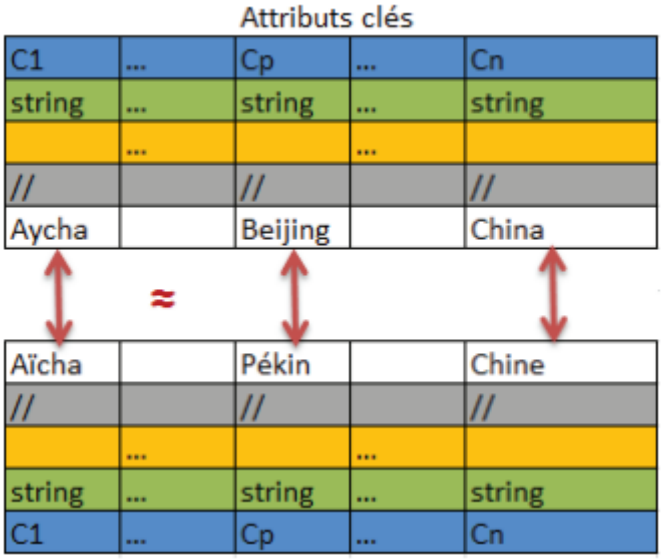


FIGURE 2.6 – Similarité entre les données

Plusieurs questions ont besoin d'une réponse justifiée : quelle mesure utiliser pour calculer la

similarité entre les noms des personnes, les noms des villes, les adresses mail, les bibliographies ? Est-ce que c'est la même mesure utilisée pour tous ? Après quel seuil à fixer pour chaque donnée et pour chaque mesure ?.
Pour répondre à une partie de ces questions, plusieurs méthodes de comparaison des attributs d'un enregistrement existent dans la littérature (Bilenko and Mooney, 2003), (Hernandez and Stolfo, 2004), (Cohen and Richman, 2004), (Koudas et al., 2006), (Winkler, 2006), (Benjalloun et al., 2009), (Berti-Équille, 2007). Nous avons classé ces différentes méthodes, qui traitent des chaînes de caractères, en deux groupes : se prononce comme et s'écrit comme. Une liste non exhaustive, présentée ci-après, des méthodes de calcul de distance de similarité classées en fonction du type des données.
Dans cette étude, nous n'allons-nous intéresser qu'à certaines méthodes. Le but étant de montrer l'apport d'une sémantique supplémentaire sur les données pour de meilleures comparaisons.

1. Mesures pour les chaînes de caractères : Il existe deux types d'algorithmes lexicographiques et phonétiques ***Mesures lexicographiques*** : les chaînes de caractère s'écrivent d'une manière similaire (Elmagarmid et al., 2007) :
   — La distance Levenshtein (Levenshtein, 1966) calcule la distance de similarité en fonction du nombre des opérations d'édition (insertion, suppression et remplacement).
   — La distance de Q-gram (Ukkonen, 1992) divise les chaînes en q-caractères. Deux chaînes sont similaires si elles partagent le plus grand nombre de Q-gram.
   — La distance de Jaro-Winkler (Winkler, 2006), (Cohen, 1998) est souvent utilisée pour les noms et prénoms. Elle identifie les caractères en commun entre deux chaînes et le nombre de transposition.

   Mesures phonétiques : les chaînes peuvent être similaires phonétiquement sans être similaires en fonction des caractères (Elmagarmid et al., 2007).
   — Soundex (Cohen, 1998) regroupe des consonnes phonétiquement identiques.
   — Double Metaphone (Philips, 2000) sont une alternative de Soundex, appliquées à d'autres langues autre que l'anglais.
   — NYSIIS (New York State Identification and Intelligence System) est un algorithme phonétique conçu en 1970. Il permet d'améliorer la précision de 2,7par rapport à l'algorithme Soundex.

   Une mesure lexicographique, telle que Jaro-Winkler ou Levenshtein, est appliquée sur le code généré par ces mesures phonétiques. Nous utilisons la notation suivante : mesure phonétique + combinée + avec la mesure lexicographique.
   — SoundexLevenshtein pour la combinaison des deux mesures Soundex et Jaro-Winkler
   — SoundexJaroWinkler : Soundex est combinée à Jaro-Winkler
   — de même pour NYSIISLevenstein, NYSIISJaroWinkler,
   DoubleMetaphoneLevenshtein et DoubleMetaphoneJaroWinkler Dans le papier (Elmagarmid et al., 2007), on conclut que la mesure JaroWinkler est mieux utilisée pour les noms.

Exemple 2.14 : Mesures de similarité sur différentes chaînes de caractères

Nous présentons dans le tableau suivant (TABLE 2.11), des distances de similarité (en) pour différentes chaînes selon les mesures de similarité présentées ci-dessus. On commence avec les distances textuelles avec Jaro-Winkler (JW), Jaccard (Jac), Levenshtein (Lv) et Q-gram (Qg). Ensuite, les mesures phonétiques telles que Soundex (Sdx), Double Metaphone (DM) et Nysiis (Ny). Ces mesures ont été combinées avec les distances Jaro-Winkler et Levenshtein.

ChaineI	Chaine2	JW	Jac	Lv	Qg	SdxLv	SdxJW	DMJW	NyLv	NyJW
Loubna	Loubna	82,20	0,00	66,60	50,00	I00,00	I00,00	I00,00	66,60	6I,I0
La belle	Labelle	79,00	0,00	57,I0	50,00	I00,00	I00,00	I00,00	I00,00	I00,00
Jérémy	Jérémi	84,90	0,00	7I,40	80,00	I00,00	I00,00	I00,00	83,30	96,60
Pékin	Beijing	54,90	0,00	25,00	25,00	50,00	50,00	66,60	33,30	57,70
Londres	London	7I,40	0,00	42,80	60,00	75,00	88,30	88,30	66,60	89,30
loubna@gmaiil.com	loubna@gmaiil.com	98,40	66,60	92,30	I00,00	I00,00	I00,00	I00,00	I00,00	I00,00
La belle@gmail.com	labelle@gmail.com	97,80	66,60	92,80	9I,60	I00,00	I00,00	I00,00	66,60	89,90
0234546456	0234546466	96,00	0,00	90,00	87,50	0,00	I00,00	I00,00	0,00	I00,00
0789923457	0289923457	87,30	0,00	90,00	77,70	0,00	I00,00	I00,00	0,00	I00,00
I5°C	-I5°C	93,30	75,00	80,00	I00,00	I00,00	I00,00	I00,00	I00,00	I00,00
90°F	90°FF	96,60	50,00	83,30	I00,00	I00,00	I00,00	I00,00	I00,00	I00,00

TABLE 2.11 – Mesures de similarité sur différentes chaînes de caractères

Les chaînes "Loubna" et "Loubna" (avec une différence au début de la chaîne) sont proches selon Jaro-Winkler à 82,20 . Elles se prononcent de la même façon : la similarité est égale à 100avec les mesures phonétiques.
Toutes les mesures textuelles ainsi que phonétiques renvoient que les deux chaînes "Pékin" et "Beijing" ne sont pas assez proches. Par exemple, avec la distance Jaro-Winkler (qui est recommandée dans la littérature pour les noms (Elmagarmid et al., 2007)), elles sont similaires à 54,90seulement. Cependant, ces deux chaînes présentent le nom d'une même ville en deux langues différentes (français et anglais). Ces deux chaînes sont dites égales sémantiquement. De même pour les chaînes "Londres" et "London"

2. Mesures pour les numériques : les numériques peuvent être comparés comme des chaînes de caractères (Elmagarmid et al., 2007).
   Cependant si on compare les deux numériques "123,0000001" et "123" en les transformant en chaînes de caractères, la distance de similarité sera très grande et donc ils seront différents. Par contre, un calcul de différence entre ces deux nombres renvoie un score très faible et donc les deux numériques sont proches.

3. Mesures pour les dates : un calcul de différence entre les jours, les mois et les années.
   Cependant, ce qui manque à ces mesures est la prise en compte de la sémantique des don-

nées dans le choix des méthodes et des seuils adéquats. Le problème devient plus complexe puisque de nos jours avec la notion du BigData, souvent les métadonnées des sources sont peu significatives ou encore inexistante.
Jusqu'à ce jour le choix de la mesure de similarité repose seulement sur quelques critères telles que la longueur de la chaîne à comparer (Sais, 2007) ou bien la distance Jaro-Winkler est mieux utilisée avec les noms (Cohen and Richman, 2004). Souvent aussi ces mesures sont choisies en fonction de l'expérience utilisateur.
L'originalité de notre travail est de recommander ces mesures en fonction de la nature/catégorie et la langue des données. Donc un de nos objectifs alors est de reconnaître la catégorie et/ou la sous-catégorie sémantique des données afin de proposer les meilleures méthodes[5].

Choix d'une approche de correspondance (Fonction Match)

Différentes approches existent pour la comparaison d'enregistrements (Kö and Rahm, 2009) : les approches probabilistes et celles déterministes. Nous ne nous sommes intéressés, dans cette étude, qu'aux approches déterministes. Parmi ces dernières, citons les techniques les plus utilisées : Techniques basées sur les distances : Le principe est de calculer la distance pondérée entre les données d'un enregistrement (Guha et al., 2004). La formule présente la somme pondérée des similarités calculées (d_a) pour les attributs clés et un poids (p_a) est donné à chacun d'eux. Cette somme devrait être inférieure à un seuil global (ε) (Dey et al., 1998). Les poids et le seuil sont définis par un expert. Aucune assistance ou recommandation pour l'utilisateur dans le choix des méthodes de similarité ainsi que les seuils

$\sum_{n-1}^{n} p_a \cdot d_a \leq \varepsilon$

Techniques basées sur les règles : Ces techniques sont un cas particulier du cas précédent tel que la distance entre deux enregistrements est comprise entre 0 et 1. Elles ont besoin de l'expertise utilisateur pour décider des règles de correspondance (Matching Rules). Parmi les travaux utilisant cette approche, il existe (Wang and Madnick, 1989), (Lim et al., 1993), (Hernandez and Stolfo, 1995), (Hernandez and Stolfo, 1998), (R. Ananthakrishna and Ganti, 2002) et (Galhardas et al., 2000). Ils utilisent une base de règles de connaissances afin de mettre en oeuvre une équation théorique. Cette dernière est l'ensemble de règles logiques qui définissent si deux enregistrements sont similaires en fonction du calcul de distance de similarité.
Vu le nombre important des méthodes de mesure de similarité, le choix de la meilleure mesure ainsi que du meilleur seuil reste toujours un casse-tête pour les utilisateurs. Ce choix devient plus compliqué quand les attributs clés, utilisés pour le dédoublonnage, n'ont pas de noms compréhensibles. Une information sémantique sur les métadonnées pourrait faciliter ce choix et lui donner un sens.
Techniques basées sur les relations entre données :
Parmi ces travaux, citons l'approche proposée dans (?) qui s'intéresse aux relations hiérarchiques existantes entre les colonnes. Celles-ci peuvent être exploiter pour rapprocher les contenu de ces colonnes.
Par exemple, il existe une relation hiérarchique entre Ville et Pays. Si pour deux valeurs distinctes

de Pays, on a un ensemble important des noms de villes en commun, alors les deux pays sont identiques.
Dans le même contexte, le travail de (Kalashnikov and Mehrotra, 2006) propose d'utiliser les relations d'une BD relationnelle afin de rapprocher les données. La problématique est de gérer l'ambiguïté des références. Chercher à associer à chaque référence son auteur. Par exemple, pour réconcilier les noms d'auteur "D. White" avec "Don White" ou "Dave White", ils proposent de comparer le contenu des autres colonnes, en relation avec la colonne Auteur, telles que Affiliation ou Co-auteur afin de reconnaître la "vraie" valeur du "D. White".
Le travail dans la thèse (Sais, 2007) et (Saïs and R.Thomopoulos, 2014), propose aussi une méthode de réconciliation des références décrites relativement à un même schéma. Elle permet de déterminer celles qui réfèrent la même entité du monde réel et celles qui réfèrent des entités différentes du monde réel. Cette approche consiste à exploiter en plus des informations syntaxiques présentes dans les données, les connaissances du domaine déclarées explicitement dans une ontologie. Dans cette approche, l'exploitation du contenu de l'ontologie a deux rôles : (i) homogénéisation des données hétérogènes en les décrivant relativement au contenu d'une même ontologie en les enrichissant par les concepts, les termes et les relations du domaine représentés dans l'ontologie et (ii) renforcer la tâche de réconciliation par des connaissances du domaine.
Cette approche propose une première méthode logique N2R (Réconciliation Numérique de références) qui repose sur les relations existantes entre les données (telles que l'équivalence, la transitivité, la symétrie et la disjonction). Ces relations sont traduites en règles de réconciliation. Un algorithme d'inférence est appliqué à ces règles afin d'en déduire la correspondance. Une deuxième méthode numérique L2R (Réconciliation Logique de références) calcule un score de similarité entre les enregistrements. L'utilité de cette approche est la combinaison des deux méthodes N2R et L2R pour décider de la réconciliation ou non des données. Exemple, si une donnée X appartient à une classe C1 et une donnée Y appartient à une classe C2, et une relation de disjonction existe entre les deux classes C1 et C2, alors X est différent de Y même si une similarité existe entre eux. Cette approche repose sur la présence de relations entre les données, cependant peu d'utilisateurs définissent ces relations. Proposer des analyses en fonction de la sémantique des métadonnées permettra la reconnaissance de certaines relations entre données. Toutes ces approches sont plutôt manuelles. Elles se basent sur le fait de connaître non seulement le sens des colonnes mais aussi les relations qui peuvent exister entre elles notamment les clés primaires et étrangères et les dépendances fonctionnelles. Par ailleurs, ces différentes approches manquent de sémantique. Aucune de ces dernières ne peut dire à la fois que les deux chaînes de caractères "London" et "Londres" sont d'un côté similaires et d'un autre coté différentes. Les deux chaînes sont similaires si et seulement si elles appartiennent à une même catégorie sémantique "City". Elles sont différentes si le sens de ces deux chaînes était "Name" ou "FirstName" (figure 2.7).

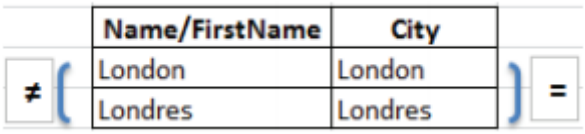


FIGURE 2.7 – Similarité sémantique entre les données

Une fois deux enregistrements sont définis comme similaires, la fusion de ces deux derniers est proposée afin de créer un enregistrement résultat plus riche.

Choix de la stratégie de fusion des tuples similaires (Fonction Merge)

La fonction Merge correspond la fusion des enregistrements. La fusion permet de créer un nouvel enregistrement contenant plus d'informations. Merge permet l'enrichissement des enregistrements : renseigner les valeurs nulles, enrichir les données (une personne peut avoir deux numéros de téléphone différents ou plus), corriger les formats des données. Cette étape demande la définition d'expression ou de règles de fusion. Quelques travaux abordent cette problématique.
Les travaux (Hernandez and Stolfo, 1995), (Hernandez and Stolfo, 1998) fusionnent deux chaînes de caractères en gardant la plus longue. Cette proposition devrait respectée certaines règles de cohérence telle que une expression régulière. Ceci permettrait d'éviter d'introduire des incohérence dans le résultat, par exemple, si "Adamsss LeBon" ne devrait pas remplacer "Adam Lebon". Qu'est ce qu'il on est des données de type date ou de type numérique ? Ou aussi que faire pour les attributs qui n'ont pas servi pour le dédoublonnage ? Comment devront-ils être fusionnés ? Des règles pour leur fusion devront être définies. Des réponses restent à apporter dans les sections suivantes.

Évaluation du taux d'élimination des similaires et des doublons

Cependant, la problématique dans la fonction Match reste les vrai/faux doublons : comment évaluer les résultats fournis par les mesures de distance de similarité et les règles de décision ? La problématique des**vrais/faux doublons** présente un vrai enjeu dans le processus d'élimination des données similaires.

Une clé de rapprochement (Matching key) est l'ensemble des attributs clés, défini par l'utilisateur et utilisé pour le dédoublonnage.

Les faux négatifs apparaissent quand la clé de rapprochement étant trop discriminante. C'est le cas où deux enregistrements représentent le même objet, mais ne sont pas identifiés comme similaires (Stricker, 2000).

Les faux positifs apparaissent quand la clé de rapprochement est non suffisamment discriminante (Stricker, 2000). C'est le cas où deux enregistrements sont identifiés similaires alors qu'ils ne représentent pas le même objet. Exemple, "Dupont" et "Dupond" sont identifiés similaires alors

qu'ils représentent deux personnes différentes.

Afin de traiter cette problématique, on propose dans la littérature un ensemble de mesures telles que la précision, le rappel, F-mesure. Le rappel=$\frac{n1}{n1+n3}$ et la précision=$\frac{n1}{n1+n2}$ (Stricker, 2000) (Table 2.12).

	Enregistrements Similaires	Enregistrements Non Similaires
Nombre d'enregistrements Supprimés	n1	n2
Nombre d'enregistrements Non Supprimés	n3	n4
Nombre Total	n5	n6

TABLE 2.12 – Le rappel et la précision

F-mesure est la mesure qui allie la précision et le rappel. C'est la moyenne harmonique de la précision et du rappel.
F-mesure=$\frac{2.precision.rappel}{precision+rappel}$.

Notons que la comparaison des enregistrements entre eux se fait de deux manières différentes : séquentielle ou par blocs[5].

Les méthodes de comparaison des tuples

Deux types de comparaison sont définis : (i) la comparaison de tous les enregistrements entre eux (la production d'un produit cartésien) et (ii) la comparaison par blocs d'enregistrements.
La première approche présentée dans (Benjalloun et al., 2009) présente trois algorithmes comparant chacun tous les enregistrements n existants dans une source. La complexité de cette approche est de l'ordre $O(n^2)$.
Quant à la deuxième approche, le principe de regroupement des données (Köpcke and Rahm, 2009) est de partitionner la source de données en des blocs selon une clé générée à partir des attributs d'un enregistrement. La clé de regroupement (Blocking key) peut être un seul attribut ou construite à partir de plusieurs attributs. Chaque méthode de regroupement définit sa propre clé. Les enregistrements sont triés selon la clé. Ils sont comparés entre eux au sein d'un même bloc. La complexité est de l'ordre de $O(n^2/b)$ pour n enregistrements et b blocs. Chaque bloc est de taille n/b en moyenne. D'autres travaux proposent $\cdots$

Outils d'élimination des doublons et similaires Différents outils existent dans la littérature telle que (Elmagarmid et al., 2007) :
SERF (Stanford Entity Resolution Framework) (Benjalloun et al., 2009) : les auteurs considèrent les fonctions Match et Merge sous forme d'une boite noire. Différents algorithmes sont proposés afin de réduire le nombre d'appel aux boites noires en gardant trace de la dernière comparaison.

Différents comparateurs sont combinés dans une disjonction des règles de correspondance.
FEBRL (Christen, 2008) est un outil open source permettant le rapprochement des enregistrements biomédicaux. Il permet la détection des doublons en utilisant des différentes mesures de distance de similarité telles que Jaro, Levenshtein, Q-gram et favorise les mesures phonétiques pour les noms. Il utilise des approches numériques et celles basées sur règles pour la décision de correspondance.
De même l'outil **TAILOR** (Elfeky et al., 2002) repose sur les mêmes décision que FEBRL (des approches numériques et celles basées sur les règles) . Il utilise cinq mesures de distance :
distance de Hamming, Levenshtein, Jaro, q-gram et soundex. Il fournit aussi des mesures statistiques telle que l'exactitude et la complétude afin d'évaluer la qualité des données étudiées.
WHIRL (Elmagarmid et al., 2007) est aussi un outil open source. Il combine les mesures cosine similarity avec TF-IDF afin de calculer la similarité entre deux listes. Toute valeur d'attribut est décomposée en un ensemble de sous chaînes. L'approche consiste à calculer une distance de similarité entre les sous chaînes. Calculer après une moyenne de ces distances et enfin comparer cette moyenne à un seuil de réconciliation.
Les différents outils existants traitent la problématique de dédoublonnage sans tenir compte de la sémantique des données.
Le choix des attributs clés, des algorithmes de similarité ainsi que les seuils de correspondance, repose sur l'expérience utilisateur. Celui-ci n'est pas assisté dans sa démarche.

2.4 Conclusion

Nous avons présenté dans ce chapitre les différentes anomalies dans les données et dans les schémas des données .Les travaux existantes dans la littérature sur les traitements des anomalies ne permettent pas de résoudre les incohérences entre les colonnes ni meme les anomalies sémantique dans un même colonne. Cette dernière peut, elle aussi reflèter la dépendance sémantique entre les données qui a toujours été négligée .À la fin de ce chapitre nous avons présenté les résultata de létude sur les outils ETL. Nous avons déduit que nous avons besoin de sémentique sur les données pour guider l'utilisateure dans les processus de l'intégration , de détection des anomalies et de nettoyage des données[4].

Chapitre 3

Méthodologie des algorithmes parallèles :

3.1 Introduction :

Ce chapitre présente tout d'abord une «méthodologie» Le parallélisme est utilisé depuis longtemps en informatique pour résoudre des problèmes scientifiques (e.g. simulation, météorologie, biologie, jeux vidéo) le plus rapidement possible. Le principe de base du parallélisme est d'utiliser plusieurs ressources (e.g. processeurs) qui fonctionnent concurremment pour accroître la puissance de calcul pour la résolution d'un même algorithme pour un problème donné. L'objectif du parallélisme est non seulement de résoudre les problèmes le plus rapidement, mais aussi de pouvoir résoudre des problèmes de plus grande taille.
Pour pouvoir mettre en œuvre le parallélisme en pratique, des systèmes ou architectures ont été développés et ne cessent d'évoluer. Ces avancées dans le domaine des architectures parallèles au cours de ces dernières années, ont permis le développement considérable de nouvelles architectures parallèles et distribuées.
Ainsi, les architectures de processeurs multi-cœurs, les processeurs graphiques (GPUs), les machines SMP (Symetric MultiProcessors), les grappes ou les grilles de calcul sont apparues. Ce qui fait qu'aujourd'hui l'utilisation du parallélisme devient de plus en plus omniprésente dans tous les systèmes.[6]

3.2 Définition des algorithmes parallèles

Dans l'informatique , un algorithme parallèle , par opposition à un traditionnel algorithme de série , est un algorithme qui peut être exécuté un morceau à la fois sur de nombreux appareils de traitement, puis combiné à nouveau ensemble à la fin pour obtenir le résultat correct.
De nombreux algorithmes parallèles sont exécutées en même temps - mais en général des algorithmes concurrents sont un concept distinct - et donc ces concepts sont souvent confondus, avec quel aspect d'un algorithme est parallèle et qui est en même temps ne pas être clairement distingués. En outre, non parallèles, les algorithmes non concurrentes sont souvent appelés « algorithmes séquentiels », par contraste avec des algorithmes concurrents. [7]

La conception et l'analyse d'algorithmes parallèles est une pierre angulaire du domaine du parallélisme. Des algorithmes parallèles rapides sont nécessaires si l'on veut atteindre une réduction significative des temps de calcul dans la résolution de problèmes complexes sur des ordinateurs parallèles. Dans cette section, la notion d'algorithme parallèle sera définie. Ensuite, les principaux thèmes relatifs à l'algorithmique parallèle, tels que la conception, l'analyse et les facteurs de performance des algorithmes parallèles, seront abordés.[8]

3.2.1 Architectures parallèles

Définition :

Les ordinateurs parallèles sont des machines qui comportent une architecture parallèle, constituée de plusieurs processeurs identiques, ou non, qui concourent au traitement d'une application. La performance d'une architecture parallèle est la combinaison des performances de ses ressources et de leur agencement. (latence, débit)

— La latence du traitement : temps nécessaire pour l'exécution d'un traitement.

— le débit du traitement :
nombre de traitement exécutable par unité de temps .[9]

Architectures parallèles

— Pas de limite de mémoire.
— Pas de limite de processeurs.
— Accélération des calculs complexes ou coûteux en temps d'occupation CPU(calcul matriciel, simulation numérique,...).
— Calcul répétitif sur un large ensemble de données structuré.
— traitement indépendant.

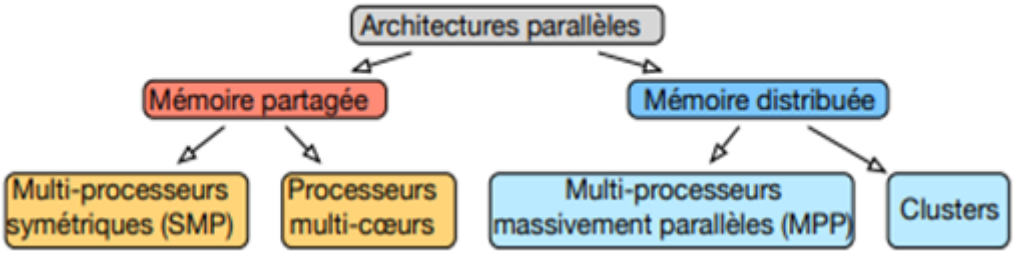


FIGURE 3.1 – Une classification des architectures parallèles [10]

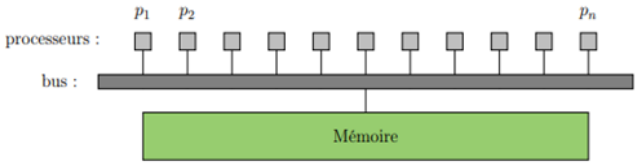


FIGURE 3.2 – Processeurs reliés à une mémoire centrale par une bus [10]

Processeur vectoriel : pipeline d'instructions

3.2.2 Les différentes sources de parallélismes [12]

Le parallélisme de données

la même opération est effectuée par chaque processeur sur des données différentes.

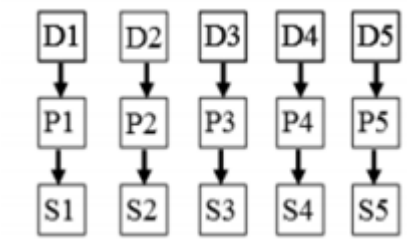


FIGURE 3.3 – Le parallélisme de données

Parallélisme de contrôle :

Des opérations sont réalisées simultanément sur plusieurs processeurs. Le programme présente des séquences d'opérations indépendantes qui peuvent être exécutées en parallèle.

Temps	Tache1	Tache2	Tache3	Tache4	Tache5
t1	Donnée1				
t2	D2	D1			
t3	D3	D2	D1		
t4	D4	D3	D2	D1	
t5	D5	D4	D3	D2	D1
t6	D6	D5	D4	D3	D2

TABLE 3.1 – Parallélisme de contrôle

Parallélisme de flux :

Les opérations sur un même flux de données peuvent être enchaînées (pipeline) .[11]

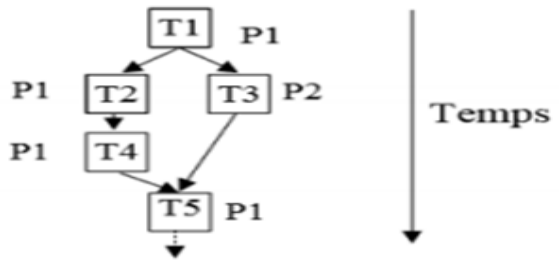


FIGURE 3.4 – Parallélisme de flux

3.3 Une démarche générale pour la méthodologie de conception d'un algorithme parallèle :

Nous décrivons ici une démarche générale de conception d'algorithmes parallèles proposée initialement par I. FOSTER [13] et détaillée par M.-J.
QUINN dans [14]. Cette démarche utilise un modèle de tâches communicantes : le programme est perçu comme un ensemble de tâches s'exécutant en parallèle et pouvant échanger des données par envoi et réception de messages. Dans ce modèle, l'envoi de données est considéré comme non-bloquant :
une fois le message envoyé, le processeur peut continuer à exécuter le code associé à la tâche courante. À l'inverse, la réception de message est bloquante :
une tâche reste en attente d'un message en provenance d'une autre tâche tant que cette dernière ne l'a pas envoyé. La démarche proposée par I. FOSTER pour concevoir un algorithme parallèle consiste à rechercher le maximum de scalabilité en traitant le plus tard possible les questions relatives à la machine utilisée. Elle se décompose en quatre étapes :
le partitionnement, où le problème est découpé en tâches élémentaires, la structuration des communications, où l'on spécifie les échanges d'informations entre tâches élémentaires nécessaires à l'exécution de l'algorithme, l'agglomération, qui consiste à regrouper les tâches élémentaires en macro–tâches de façon à minimiser les communications, et enfin le placement, qui affecte chaque macro–tâche à une unité de calcul de façon à équilibrer la charge. (FIGURE 3.5) représente schématiquement le rôle de chacune de ces étapes dans la construction d'un algorithme parallèle.
Nous allons maintenant détailler les principes de chacune de ces étapes.

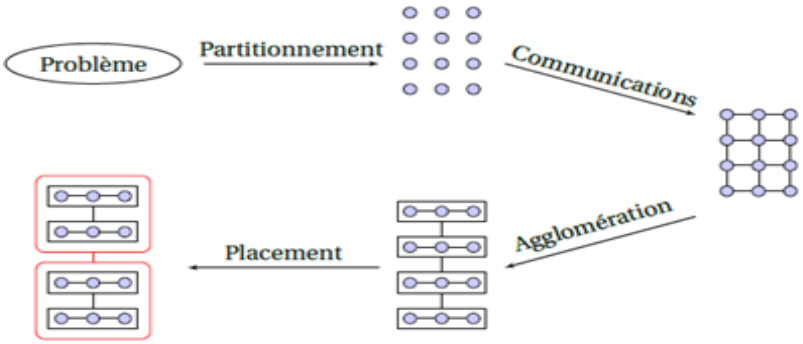


FIGURE 3.5 – La méthode de conception d'algorithmes parallèles de I. FOSTER

3.3.1 Les 4 étapes dans la conception d'un algorithme parallèle [15]

Partitionnement

Division des opérations et des données en morceaux

- Peut être centré sur les données ou sur les instructions .
- Décomposition de domaine
  - Division des données
  - Association des instructions aux données
- Décomposition fonctionnelle
  - Division des instructions
  - Association des données aux instructions
- But : Identifier un max. de tâches primitives
- Les critères pour évaluer la qualité d'un partitionnement
  - Le nombre de tâches primitives est considérablement plus grand que le nombre de processeurs de l'ordinateur utilisé
  - Les instructions et données redondantes sont minimisées (extensibilité $\longrightarrow tailledeproblème)Lestâche$
- Le nombre de tâches primitives augmente en fonction de la taille du problème (extensibilité $\longrightarrow nombredeprocesseurs$)

Communication :

- formes de communication entre les tâches
- Locale
  - Une tâche nécessite des données provenant d'un petit nombre d'autres tâches.
  - Création de canaux partant des tâches fournissant les données, vers la tâche qui a besoin de ces données.
- Globale
  - Une tâche nécessite des données provenant d'un grand nombre d'autres tâches

  - Gestion des canaux plus complexe et généralement traitée plus tard dans le processus de conception
- Les communications entre tâches d'un algorithme parallèle sont une source de coûts et doivent être minimisées autant que possible
- critères permettent d'évaluer la structure des communications d'un algorithme parallèle
  - Communications équilibrées entre les tâches
  - Chaque tâche communique avec un petit nombre de tâches voisines
  - Les tâches peuvent communiquer de façon concurrente
  - Les tâches peuvent faire leurs calculs de façon concurrente

Agglomération :

- À partir de cette étape, une architecture cible doit être identifiée
- Regroupement de tâches primitives en tâches de plus grande taille
  - Amélioration de la performance
  - Simplification de la programmation
- L'agglomération de tâches primitives communicantes :
  - Réduit la charge de communication
  - Augmente la localité de l'algorithme parallèle

  7 critères pour évaluer la qualité de l'agglomération :
- Augmentation de la localité
- La réplication de calcul prend moins de temps que les communications qu'elle remplacent
- La quantité de données répliquée est assez petite pour permettre l'extensibilité
- Les tâches agglomérées ont des coûts de calcul et de communication similaires
- Le nombre de tâches augmente avec la taille de problème
- Le nombre de tâches est aussi petit que possible, tout en étant plus grand que le nombre de processeurs
- L'agglomération choisie pour la parallélisation d'un code séquentiel implique des coûts raisonnables de modification

Assignation (mapping) :

- Buts :
  - Maximiser l'utilisation des processeurs (‰moyen du temps où les processeurs sont actifs durant l'exécution)
  - Minimiser les communications entre processeurs
- Ces 2 buts sont souvent contradictoires, un compromis est alors nécessaire

  4 règles pour évaluer la qualité de l'assignation
- Considération d'une seule tâche par proc et de plusieurs tâches par proc
- Considération des allocations statiques et dynamiques des tâches aux procs

- Si une allocation dynamique a été choisie, vérification qu'elle n'est pas un goulot d'étranglement au niveau du système
- Si une allocation statique a été choisie, vérification que le ratio tâche/proc est sufissament grand

3.3.2 Partitionnement du problème :

L'objectif principal de la phase de partitionnement consiste à identifier le maximum de parallélisme possible en décomposant le problème en tâches élémentaires. Cette décomposition porte à la fois sur les données du problème et sur les instructions agissant sur ces données. Ainsi, cette phase peut être abordée de deux façons, selon que l'on cherche avant tout à partitionner les données du problèmes (on parle alors de parallélisme de données, ou de parallélisme structurel), ou bien les intructions du programme (on parle alors de parallélisme de tâches, ou de parallélisme fonctionnel). Quelle que soit la méthode considérée, l'objectif est de maximiser le nombre de tâches élémentaires, car il s'agit d'une borne supérieure sur les capacités d'exécution en parallèle de l'algorithme. Un bon partitionnement devrait vérifier les propriétés suivantes :

— le nombre de tâches élémentaires doit être supérieur d'au moins un ordre de grandeur au nombre de processeurs de la machine cible (ceci afin de faciliter les choix d'implémentation ultérieurs)

— la redondance des instructions et des données au sein des tâches élémentaires doit être minimale (dans le cas contraire, l'algorithme obtenu pourrait ne pas pouvoir supporter l'augmentation de la taille du problème)

— les tâches élémentaires doivent représenter des charges de travail sensiblement équivalentes (si tel n'est pas le cas, il sera difficile par la suite d'équilibrer la charge entre les processeurs)

— le nombre de tâches élémentaires doit être une fonction croissante de la taille du problème (sinon, il pourrait s'avérer impossible d'utiliser plus de processeurs pour résoudre un problème plus grand). [16]

3.3.3 Définition des schémas de communication :

Une fois les tâches élémentaires identifiées, il s'agit de déterminer le schéma de communication qu'il est nécessaire de mettre en place pour implémenter l'algorithme.
Nous pouvons distinguer essentiellement deux types de communications :
les communications locales, qui n'impliquent qu'un nombre restreint de tâches, et les communications globales, faisant intervenir la plupart, si ce ne sont toutes les tâches élémentaires. Bien qu'il soit pertinent de noter la présence de communications globales, ces dernières ne sont pas réellement importantes à ce stade de la conception de l'algorithme. Leur effet principal est d'introduire un point de synchronisation dans l'algorithme, mais comme une large majorité des tâches sont impliquées dans une communication globale, elles n'introduisent pas de réel déséquilibre, et la seule façon de les optimiser réside dans le câblage des connexions physiques du réseau reliant les unités de calcul. À l'inverse, les communications locales peuvent introduire un réel déséquilibre et un surcoût important à l'exécution si elles ne sont pas optimisées.
Nous devons donc les prendre en compte, et pour cela, nous construisons un graphe dans lequel

chaque nœud représente une tâche, et chaque arête reliant une tâche à une autre correspond à l'introduction d'une communication locale entre ces deux tâches. Les communications inter–processeurs représentent une part non–négligeable du surcoût introduit par le parallélisme, en ce sens qu'un algorithme séquentiel n'en génère pas. Il est donc très important d'évaluer la structure du schéma de communication pour se faire une idée des propriétés de l'algorithme. Ainsi, un schéma de communication sera considéré efficace si :

— les communications sont réparties équitablement entre les tâches
— les communications peuvent être réalisées de façon concurrente
— les temps de communication peuvent être rec
ouverts par des phases de calcul. [16]

3.3.4 Agglomération des tâches élémentaires :

L'objectif des deux premières étapes est de fournir une description de l'algorithme à implémenter en identifiant le maximum de sources de parallélisme possible.
Cependant, si le nombre de tâches élémentaires est très supérieur au nombre de processeurs disponibles sur la machine cible, une telle granularité peut singulièrement nuire à l'efficacité de l'algorithme.
Le simple fait de générer la totalité de ces tâches élémentaires peut s'avérer coûteux, et le schéma de communication devient également prohibitif. C'est pourquoi nous introduisons une phase d'agglomération, qui consiste à regrouper les tâches élémentaires en tâches plus importantes, en prenant en compte les caractéristiques de la machine cible.
L'un des objectifs de cette phase d'agglomération est de préserver la localité de l'algorithme parallèle :
l'idée consiste à regrouper ensemble des tâches élémentaires qui communiquent entre elles. De cette façon, on minimise les communications entre les tâches agglomérées. Une autre idée retenue pour cette phase d'agglomération consiste à regrouper d'un côté les tâches qui envoient des messages, et de l'autre celles qui les reçoivent. De cette façon, les messages élémentaires peuvent être regroupés en un unique message, ce qui permet de minimiser la latence des communications. Ces deux idées sont illustrées sur(FIGURE 3.6).
Une façon simple de réaliser cette phase consiste à utiliser un algorithme de partitionnement de graphe (voir par exemple [20, 19, 18, 17]). La qualité de l'agglomération des tâches peut être évaluée à partir des critères suivants :

— l'agglomération a augmenté la localité de l'algorithme.

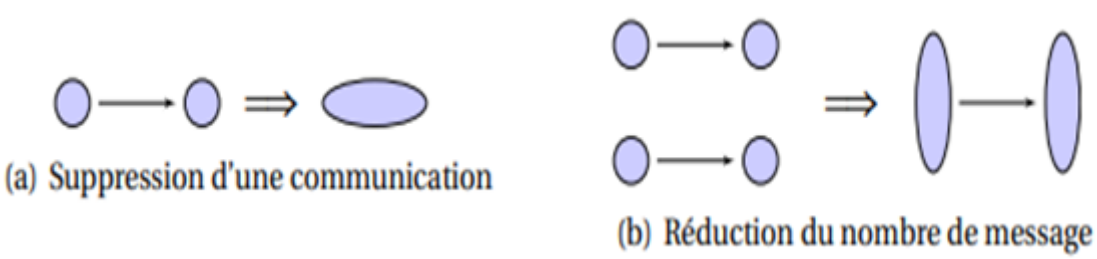


FIGURE 3.6 – Exemples de techniques d'agglomération de tâches élémentaires

— la quantité de données dupliquées reste faible.
— les coûts d'exécution et de communication des tâches sont comparables.
— le nombre de tâches est une fonction croissante de la taille du problème. [16]

3.3.5 Placement des tâches :

Le placement des tâches consiste à distribuer les tâches agglomérées sur les processeurs disponibles. Bien évidemment, si le nombre de tâches est fixe, et que la phase d'agglomération a permis de construire autant de tâches que de processeurs, cette phase devient triviale.
Il n'est cependant pas rare de rencontrer des algorithmes dont le nombre de tâches varie au cours de l'exécution, et dans ce cas, la phase de placement prend tout son sens.
L'objectif consiste généralement à trouver un équilibre entre une utilisation maximale des capacités de la machine, et une minimisation du volume de communications.

En général, le placement des tâches sera considéré comme pertinent si les conditions suivantes sont vérifiées :
— dans le cas d'un placement statique, le nombre de tâches par processeurs devrait être de l'ordre de la dizaine au plus .
— dans le cas d'un placement dynamique, le processus de distribution des tâches ne doit pas constituer un goulot d'étranglement lors de l'exécution. [16]

3.4 Techniques d'implémentation d'algorithmes parallèles

La programmation effective d'un algorithme sur une machine parallèle peut être réalisée à l'aide de différents outils. Comme nous allons le voir, certains de ces outils sont particulièrement adaptés à un type de machine donné, alors que d'autre sont plus généraux et permettent d'utiliser plusieurs modèles de programmation sur différents types de machines.
Ainsi, il existe essentiellement deux outils permettant d'implémenter un programme parallèle destiné à être exécuté sur un multiprocesseur. Le premier repose sur la notion de processus légers, aussi appelés threads selon la terminologie anglo–saxonne.
Le standard POSIX définit une spécification précise de l'interface de programmation de ces processus légers, l'implémentation étant généralement réalisée par le concepteur du système d'exploitation de la machine.
Dans ce modèle, l'accès à la mémoire est partagé, et les communications sont réalisées par l'utilisation explicite de variables communes dont l'accès est règlementé par l'introduction de verrous.
. Une différence essentielle entre ces deux outils se situe au niveau du mode de compilation du programme. En effet, l'approche par processus légers est généralement mise en oeuvre par liaison avec une librairie binaire, alors que l'utilisation du standard OpenMP repose sur l'appel à un compilateur spécifique capable d'interpréter les directives de parallélisation. Ces directives permettent de spécifier la distribution des données sur les processeurs et de paralléliser l'exécution des itérations sur les données, chaque processeur n'itérant que sur les données qui lui ont été attribuées.
Pour la programmation sur multicalculateur, l'approche usuelle consiste à développer le programme

à l'aide d'un modèle de passage de messages tel que celui proposé par le standard MPI. Selon ce modèle, chaque unité de calcul est perçue comme un processeur pouvant envoyer (resp. recevoir) des messages à destination (resp.en provenance) des autres processeurs.
Pour cela, le standardMPI fournit de nombreuses fonctions de communication permettant l'élaboration de schémas de communication très variés :
communications point–à–point ou globales, communications synchrones ou asynchrones, communications bloquantes ou non–bloquantes, etc. Nous renvoyons à ainsi qu'à la spécification du standard MPI pour le détail des possibilités offertes.
La flexibilité de ce système est à l'origine de sa portabilité et de sa popularité. Il est en effet possible de l'utiliser sur la quasi–totalité des machines parallèles, et les constructeurs fournissent généralement une implémentation optimisée de ce standard pour chacune de leurs machines. Notons que bien que la sémantique des fonctions définies dans le standard MPI soit particulièrement adaptée au modèle d'exécution des multicalculateurs, il est tout–à–fait possible de développer un programme parallèle utilisantMPI pour l'exécuter ensuite sur un multiprocesseur.
L'efficacité du programme dépendra alors de la qualité de l'implémentation du standard sur cette machine.
Enfin, si l'on désire utiliser le modèle de programmation BSP évoqué plus haut, il existe essentiellement deux librairies fournissant une interface de programmation adaptée à sa mise en oeuvre. Il s'agit de la BSPlib et de la Paderborn University BSP Library .Ces librairies proposent chacune une petite collection de fonctions de communication et de synchronisation de haut niveau reflétant les fonctionnalités requises par le modèle BSP.
L'implémentation de ces fonctions reposent sur le standard MPI, ce qui assure leur portabilité.[21]

3.5 Les méthodes proposées pour extraire les données parallèles à partir d'un corpus [22]

La meilleure façon d'extraire les données parallèles à partir d'un corpus comparable est encore une question ouverte qui a reçu beaucoup d'attention de la part de la communauté de chercheurs dans les années récentes. Plusieurs méthodes sont proposées pour extraire (ou chercher) à partir de ce type de corpus les documents comparables [23], [24], etc. ; les phrases parallèles [25], [26], [27], etc. ; et des fragments ou des dictionnaires bilingues [28], [29], [Cettolo 2010], etc. Les méthodes de recherche peuvent être regroupées de la façon suivante :
- les approches de recherche qui utilisent des caractéristiques générales des documents (telles que le titre du document, le lien du document, l'information dans le lien, la structure du document, etc.) ainsi que des informations lexicales du document ; - les approches qui utilisent des techniques de recherche d'information translingue (crosslanguage information retrieval) nécessitant un module de traduction.
Dans nos travaux, nous nous concentrons sur l'extraction de documents parallèles et de phrases parallèles pour construire un grand corpus d'apprentissage afin de développer un système de traduction probabiliste dans le cas d'une langue peu dotée. Les deux approches ci-dessus sont considérées. Au point de départ, nous n'avons qu'un seul site web multilingue de cette langue peu dotée, nous

n'avons pas de données parallèles disponibles ou de données supplémentaires. Les trois chapitres suivants présentent nos contributions et méthodes proposées, qui sont au nombre de trois.
La première méthode concerne la première approche pour extraire à la fois les documents comparables et les phrases parallèles. Cette méthode requiert des données supplémentaires sur la paire de langues (telles qu'un dictionnaire bilingue, une liste de mots d'arrêt, etc.) Lorsque ces données supplémentaires sont disponibles, nous pouvons appliquer cette méthode pour toute paire de langues.
La deuxième méthode suit l'approche de recherche d'information translingue mais nous proposons une méthode non supervisée qui peut s'appliquer dans le cas de langues peu dotées (où l'application directe de la technique de recherche d'information translingue est impossible à cause des données requises en entrée qui ne sont pas disponibles).
Le processus d'extraction ne requiert aucune donnée supplémentaire comme dans la première méthode.
Cette méthode peut être appliquée pour toute paire de langues, pour un corpus comparable en entrée seulement, et aucune donnée supplémentaire n'est requise sur les deux langues considérées.
La dernière méthode aborde l'utilisation d'une troisième langue dans l'extraction du corpus bilingue. C'est une application de l'approche de triangulation, introduite pour la traduction automatique, au processus d'extraction de données parallèles.
L'approche de triangulation a été utilisée récemment pour augmenter la qualité d'un système de traduction.
Nous proposons d'utiliser une troisième langue dans le processus d'extraction des données parallèles à partir d'un corpus comparable.Les détails de ces approches sont présentés dans les chapitres suivants avec des expériences appliquées sur la langue peu dotée vietnamienne et les langues française et anglaise.

3.6 Les facteurs de performance des algorithmes parallèles [8]

L'approche la plus naturelle pour développer un algorithme parallèle est de prendre un algorithme séquentiel, d'identifier les étapes indépendantes et d'assigner ces étapes à processeurs.
Ce faisant, nous obtenons une version parallèle de cet algorithme. Cependant, cette simple conversion donne souvent des algorithmes possédant un faible degré de parallélisme et qui ne sont pas efficaces (Codenotti et Leoncini (1993)).
Il ne faut pas non plus oublier que l'extraction du parallélisme du meilleur algorithme séquentiel n'aboutit pas forcément au meilleur algorithme parallèle, chaque algorithme présentant un profil plus ou moins favorable à une parallélisation (Gentler et al. (1996)).
Lorsque que l'on considère un algorithme parallèle ou lorsqu'on envisage la parallélisation d'un algorithme, il convient d'étudier certains paramètres importants. Cette section présente ceux qui sont les plus pertinents à ce travail.

3.6.1 Le grain de parallélisme

Le grain de parallélisme est la taille moyenne des tâches élémentaires qui ont guidé la parallélisation.

On le mesure souvent en nombre d'instructions machine, en taille de la mémoire employée ou en durée d'exécution.
Le choix de ce grain est fortement lié aux caractéristiques de la machine utilisée (architecture, nombre de processeurs, etc.).
De manière sommaire, on parle souvent de gros grain coarse grain) et de grain fin (fine grain) pour indiquer la taille des entités de parallélisation. Schématiquement, on considère la hiérarchie suivante (de gros grain à grain fin) :
programmes, procédures, instructions, expressions, opérateurs et bits.
Mesure complémentaire très étroitement liée au grain de parallélisme, le degré de parallélisme correspond à une mesure du nombre d'opérations exécutées simultanément et reflète le nombre de processeurs que l'on pourra utiliser.
Il peut être évalué systématiquement à partir de la description de l'algorithme parallèle ou évalué dynamiquement par des mesures pendant l'exécution. Le degré de parallélisme peut varier largement dans les différentes parties d'un programme.
Ainsi, on est souvent amené à considérer les degrés de parallélisme maximal, minimal et moyen d'un algorithme Comme le degré maximal constitue une borne supérieure du nombre de processeurs, le choix d'utiliser ce maximum pour un algorithme donné permet d'obtenir une bonne rapidité d'exécution durant certaines parties de l'algorithme, mais peut donner une très mauvaise efficacité dans le cas où le degré maximal excède de loin le degré moyen. Dans ce cas, un grand nombre de processeurs sont laissés inactifs pendant une partie considérable de la durée d'exécution du programme.
La détermination du nombre optimal de processeurs, du point de vue de l'efficacité obtenue, est souvent difficile.

3.6.2 Les communications

Une partie du temps d'exécution d'un algorithme est consacrée aux échanges d'information entre deux ou plusieurs processeurs.
L'ensemble de ces échanges représente la charge de communication de cet algorithme, charge qui peut être plus ou moins prohibitive sur l'efficacité réalisable selon le cas. Dans un modèle à mémoire partagée, l'échange d'information entre les processeurs se fait par l'entremise de la mémoire commune qui est lue et écrite par les processeurs.
La charge de communication d'un algorithme dans ce modèle est donc représentée par la taille des données transférées entre la mémoire partagée et les mémoires locales (registres) de tous les processeurs (JàJà (1992)). Cette charge dépend entre autres des temps d'accès à la mémoire et des mécanismes de gestion des lectures/écritures simultanées.
Dans un modèle à mémoires distribuées, les communications sont effectuées par passage de messages, c'est-à-dire que les processeurs envoient/reçoivent explicitement des données et des instructions aux/des autres processeurs. Des algorithmes de routage sont utilisés pour faire parvenir les messages d'un processeur à un autre.
Le coût des communications d'un algorithme dans ce modèle dépend, entre autres, de la topologie du réseau de processeurs, de la bande passante disponible sur les canaux de communication reliant ces processeurs, de la taille des messages et de la quantité de messages envoyés durant l'exécution

de l'algorithme.
Les deux derniers éléments sont directement reliés à la structure de l'algorithme et au nombre de processeurs utilisés.

3.6.3 La synchronisation

Dans un environnement parallèle où des opérations sont effectuées simultanément, il est souvent nécessaire d'établir un certain ordre dans le déroulement des événements qui sont dépendants les uns des autres.
Les opérations de synchronisation permettent, entre autres, de garantir que tous les processeurs ont terminé une partie de leurs calculs à un moment donné, ou qu'une opération est effectuée par un seul processeur à la fois si nécessaire.
Cette opération est utile, par exemple, dans le cas où le processeur px a besoin d'une information qui est calculée par le processeur py. Il faut s'assurer que l'information a été effectivement calculée par py avant son utilisation par px pour que l'algorithme s'exécute correctement.
Dans un modèle à mémoire partagée, les synchronisations peuvent être effectuées, entre autres, par des barrières ou des sémaphores.
Si une barrière est placée dans un programme, aucun processeur ne pourra continuer son exécution au-delà de ce point tant que tous les processeurs ne l'auront pas atteint.
Les sémaphores permettent de gérer les entrées et les sorties dans les zones critiques, c'est-à-dire les parties de l'algorithme qui ne peuvent être exécutées que par un seul processeur (ou un nombre maximal déterminé) à la fois.
Dans un modèle à passage de messages, les communications sont effectuées par des opérations d'envoi et de réception de messages.
Une façon de réaliser une synchronisation dans ce modèle est en utilisant des envois et des réceptions bloquants.
Par exemple, au moment où les processeurs ont fini leur calcul et sont prêts à être synchronisés, ils envoient un message indiquant ce fait à un processeur chargé de gérer la synchronisation. Ensuite, les processeurs attendent (n'effectuent aucun traitement) tant qu'ils n'ont pas reçu le message de fin de synchronisation.
Durant une synchronisation, les processeurs demeurent inactifs durant un certain délai qui peut être plus ou moins long selon l'algorithme et le modèle utilisé.
Ce délai peut représenter un obstacle considérable à l'atteinte d'une efficacité parallèle satisfaisante, donc doit être minimisé autant que possible.

3.6.4 Le placement des tâches

La conception d'un algorithme parallèle implique la répartition de calculs et de données de tailles possiblement différentes sur l'ensemble des processeurs.
Cette répartition des tâches doit être la plus équitable possible afin de réduire au minimum le temps où les processeurs sont improductifs et ainsi maximiser l'efficacité de l'algorithme.
Il arrive cependant que la charge de calcul d'une tâche n'est pas connue à l'avance, que le nombre

de tâches varie durant l'exécution ou que des contraintes système ou technologiques font en sorte de ralentir le temps d'exécution de certaines tâches.
Selon Calégari (1999), trois stratégies d'allocation des tâches sont possibles tout dépendant du moment où l'allocation et le nombre des tâches sont déterminés.
S'ils sont tous les deux déterminés au moment de la compilation du programme, alors l'allocation est statique.
Si le nombre de tâches est déterminé au moment de la compilation et l'allocation se fait durant l'exécution, alors l'allocation est dynamique.
Finalement, s'ils peuvent tous les deux varier durant l'exécution du programme, alors l'allocation est adaptative.
Dans les deux derniers cas, un algorithme d'équilibre de charges (load balancing) est nécessaire pour allouer les tâches aux processeurs au moment de l'exécution de l'algorithme.
Si la charge de calcul des différentes tâches est plus ou moins la même, alors une stratégie d'allocation statique peut être suffisante si le nombre de tâches est connu.
Si, à l'opposé, les tâches sont hétérogènes, alors une stratégie dynamique sera probablement nécessaire si l'on veut atteindre une bonne efficacité. L'architecture utilisée peut influencer le choix de la stratégie de placement des tâches.
Par exemple, si l'environnement parallèle utilisé est un réseau de stations dont la charge est variable, alors l'allocation dynamique des tâches aux processeurs pourrait être la plus appropriée même si leur hétérogénéité est relativement faible. Notons également que la granularité de la parallélisation et le placement des tâches sont considérablement reliées.
Si l'allocation des tâches est coûteuse et qu'il y a plus de tâches concurrentes que de processeurs, il peut être intéressant de regrouper un certain nombre de tâches et, par le fait même, augmenter la granularité et la performance.
Afin d'utiliser de façon pratique, sur des ordinateurs réels, les algorithmes parallèles conçus, il existe divers environnements et langages de programmation parallèles.
La prochaine et dernière section de ce chapitre sur le parallélisme sera consacrée au survol des principaux modèles de programmation existants.

3.7 conclusion

Dans ce chapitre, nous avons présenté quelques notions sur la programmation parallèle.
La programmation parallèle est plus complexe que la programmation séquentielle car elle nécessite la prise en compte d'éléments supplémentaires comme la gestion des synchronisations entre les ressources de l'environnement d'exécution ou les copies de données

Chapitre 4

Implémentation de la solution

4.1 Introduction

Nous présentons dans ce chapitre, les outils exploités pour le développement du logiciel tels que le choix du langage de programmation, l'environnement de programmation, ainsi que l'ensemble des solutions des expérimentations par toutes les approches proposées. Nous terminons par une conclusion.

4.2 Environnement et matériel :

Pour la réalisation de notre application, nous avons eu recours à plusieurs moyens matériels et logiciels :

4.2.1 Environnement :

Le langage Java :

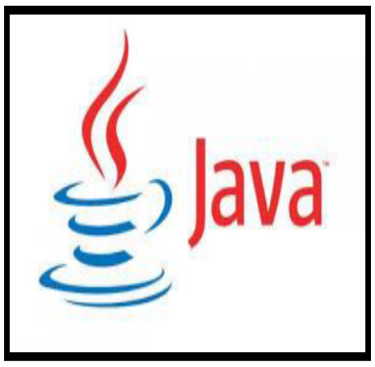


FIGURE 4.1 – NetBeans

Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld.
Une particularité de Java est que les logiciels écrits dans ce langage sont compilés vers une représentation binaire intermédiaire qui peut être exécutée dans une machine virtuelle Java (JVM) en faisant abstraction du système d'exploitation[31].
Nous avons implémenté l'algorithmes en Java pour les raisons suivantes :

Excellente documentation.

Environnement de développement confortable .

Nombreuses librairies disponibles.

NetBeans :

FIGURE 4.2 – NetBeans

Notre application est implémentée sous l'environnement Netbeans qui est un environnement de développement intégré (IDE) dont le but est de fournir une plate-forme modulaire pour permettre de réaliser des développements informatiques.
IL placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres (dont Python et Ruby) par l'ajout de greffons. Il offre toutes les facilités d'un IDE moderne (éditeur avec coloration syntaxique, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Compilé en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

NetBeans constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate forme.

L'IDE Netbeans s'enrichit à l'aide de greffons[32].

PostgreSQL :

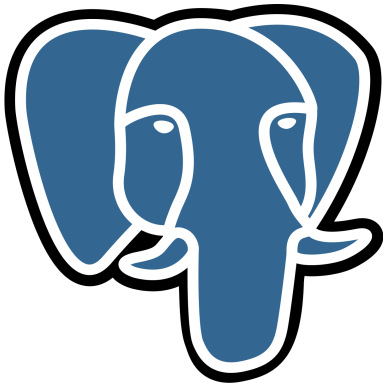


FIGURE 4.3 – SQL Server

Notre base de donnnnées est créer par PostgreSQL qui est désigne couramment un serveur de base de données. La définition du SQL server est étroitement liée à celle du langage SQL (Structured Query Language), un langage informatique permettant d'exploiter des bases de données.

Concrètement, un SQL server est un outil qui possède toutes les caractéristiques pour pouvoir accompagner l'utilisateur dans la manipulation, le contrôle, le tri, la mise à jour, et bien d'autres actions encore, de bases de données grâce au langage SQL [33].

4.2.2 Matériel :

Le développement de l'application est réalisé via un ordinateur portable ayant les caractéristiques suivantes :

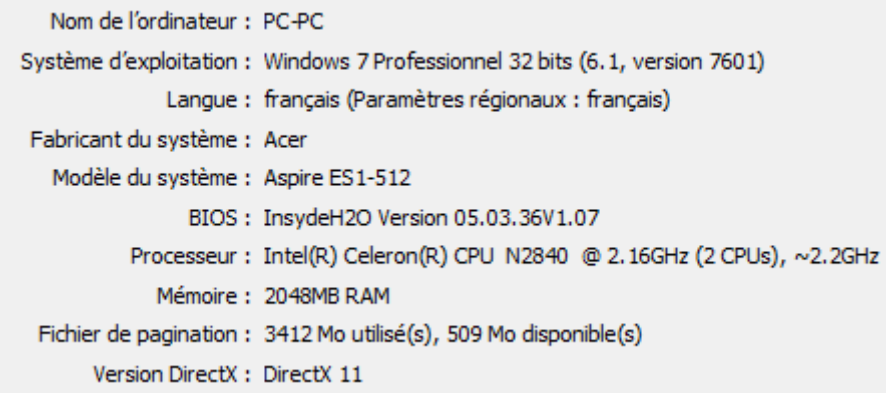


FIGURE 4.4 – l'ordinateur

4.3 Corpus utilisés :

Nous avons utiliser le corpus de (F.boufares et A.Bensalem) présenter dans la figure (FIGURE 4.5) avec quelque modification.

	Nom et Prénom	Téléphone	Mail
R1	Le Bon Adam	0666600007	lebon@yahoo.fr
R2	Le Bon A.	0666677777	
R3	Le B. Adam	0666677777	lebon@yahoo.fr
R4	Grande Clémence	0666688887	grande@yahoo.fr
R5	Adam LeBon	0666677777	
R6	Unique Eve	0666622227	unique@gmail.com
R7	Le Bon Adam	0666600007	lebon@yahoo.fr
R8	Le Bon	0666600007	lebon@yahoo.fr
R9	LeBon Adeline	0666611117	Alebon@yahoo.fr

FIGURE 4.5 – corpus

4.4 Elimination des doublons et similaires

4.4.1 Introduction

Nous allons maintenant nettoyer les données ligne par ligne. Pour cela, nous nous sommes intéressés dans cette thèse à la problématique qui préoccupe de plus en plus les travaux de recherche ainsi que les organisations, qui est le dédoublonnage des données. Nous avons alors en entrée à ce processus, des données homogènes (standardisées, unifiées dans un même format, code et langue). L'ordre de ce traitement (homogénéisation suivi de déduplication des données) est justifié par des expérimentations. Soit S une source qui contient doublons et similaires.On va éliminer les doubles sur deux versions de la table Client. Comme présentée dans l'état d'art(chapitre 1), la déduplication des données repose sur deux fonction Match et Merge.

4.5 Calcul de distance de similarité

Plusieurs méthodes de calcul de distance de similarité ont été présentées dans la littérature selon les types des données à comparer. Nous présentons, ci-dessous, les méthodes les plus connues. Pour plus de détails, on pourra consulter par exemple . Les types de données peuvent être résumés en deux grandes catégories (simples et complexes) :

Il existe deux types d'algorithmes de comparaison de données de types chaînes de caractères ou textes. Le choix est basé sur deux critères :
s'écrirecomme et *seprononcercomme*.
Nous présentons, dans ce qui suit, cinq méthodes pour le calcul de la distance de similarité de données. Les trois premières méthodes se basent sur le concept *s'écrirecomme*. Alors que les deux autres se basent sur le concept "*seprononcercomme*".

La distance de *Levenshtein*[1] est égale au nombre minimal de caractères qu'il faut supprimer, insérer, ou remplacer pour passer d'une chaîne à une autre.

La distance de Jaro-Winkler est adaptée au traitement de chaînes courtes comme des noms ou des mots de passe .

La distance de Jaccard mesure la diversité entre des chaînes longues. En testant ces méthodes, Jaro-Winkler semble mieux fonctionner pour les chaînes de caractère de différents types et de longueur variable.

L'algorithme Soundex produit une clé phonétique d'un mot. Il fonctionne sur la consonance anglo-saxonne du langage.

L'algorithme *Metaphone*[4] est un algorithme pour indexer les mots selon leur sonorité lorsqu'ils sont prononcés en anglais. La version double tient compte de la sonorité des langues étrangères comme le français, le grec et le latin.
L'algorithme Soundex produit de nombreux faux doublons, principalement parce qu'il est basé sur les premiers caractères de la chaîne pour générer le code soundex. Une autre limitation est que c'est un algorithme phonétique, de sorte que le résultat dépend de la langue. Le soundex est adapté à l'anglais.
Les algorithmes de comparaison de données numériques et binaires permettent de calculer la distance de similarité entre des données binaires, tels que Jaccard, Indice de Sokal Michener, Indice de Rogers et Tanimoto et Hamming, et des données numériques, tels que la distance Euclidienne, la distance Manhattan, Bennani (2006).

On peut trouver dans la littérature plusieurs algorithmes pour comparer des données de type complexe telles que les images, les sons et les textes. On peut citer, d'une part, ceux basés sur l'approche bayésienne, Bennani (2006) ou la distance de Hausdorff pourd manipuler des images, et d'autre part, le double Metaphone, Soundex, Phonex, Similar Text, Cosinus similarité, SVM pour les autres types de données.
Plusieurs travaux sur l'élimination des données similaires et des doublons existent dans la littérature (Benjalloun et al., 2009), (Cohen et Richman, 2004), (Bilenko et Mooney, 2003), (Hernandez et Stolfo, 1998). Les fonction de comparaison utilisent des règles de décision portant sur les n attributs. Il existe deux types de décision :

dichotomique (similaires ou non) -c'est notre cas ; ou

trois réponses (similaires ou non et besoin de plus de révision) . La fonction de comparaison peut êre basée, soit sur les valeurs d'attributs, soit sur le concept de contexte . Plusieurs approches existent pour fusionner différents tuples telles que les approches basées sur les règles ou sur les workflows ou encore les approches numériques. Concernant les approches basées sur les règles, la fusion se fait selon une combinaison logique des conditions de correspondance. Alors que celles basées sur les workflows permettent des combinaisons aléatoires. [82]

4.6 Processus de comparaison et de fusion (Match and Merge)

L'élimination de données similaires nécessite le développement de deux fonctions "Match" (comparer) et "Merge" (fusionner) (Hernandez et Stolfo, 1998). La fonction Match permet de définir si deux tuples dans une table sont similaires et la fonction Merge permet d'en générer un nouveau en fusionnant les deux supposés similaires.
Introduisons à présent quelques notations. Soit T une table et C l'ensemble de ses attributs, on note T(C). On considère que$C = A \cup B$.
A est l'ensemble des attributs qui servent pour éliminer les tuples similaires dans la table T.

$B = C - A$ et $A \cap B = \phi$ (B peut être vide).

Soit $[[D_k]]$ l'ensemble de valeurs concrètes d'un tuple de données tels que les chaînes de caractères alphanumériques (String), les numériques (Number), les dates, les bouléens (Boolean) ou encore des listes et des intervalles de valeurs. Une table T est interprétée par l'ensemble [[T]] de tous les $\{A_1 \cdots A_n, B_1 \cdots B_m\}$ -tuples définis sur $[[D_k]]_{k=1;n+m}$. [[T]] est l'ensemble des fonctions avec $t : \{A_1 \cdots A_n, B_1 \cdots B_m\} \rightarrow \cup[[D_k]]_{k=1;n+m}$ tel que $t(A_i)$ noté $t.A_i$ $(respect.t.B_j)$ est un élément de $[[D_i]]$ (respect. $[[D_j]]$). Chaque élément t de [[T]] est un tuple de la table T et chaque $t.A_i$ est une valeur de l'attribut dans t, que l'on notera aussi υ.

Définition 1 :

Similarité entre valeurs (*notée* $\approx$) : Deux valeurs υ et $\upsilon^{'}$ sont similaires, on note $(\upsilon \approx \upsilon^{'})$, ssi la distance de similarité d , calculée entre ces deux valeurs, vérifie une condition k. Pour deux tuples t et $t^{'}$, pour un attribut $A_{i(i=1;n)}$, $t.\ A_i \approx t^{'}.\ A_i$ ssi la condition k_i est vérifiée.

la condition k_i se base sur le calcul de la distance d_i de similarité selon le type des données.

Plusieurs cas de figures peuvent être envisagés. L'utilisateur peut ainsi fixer (donner) un ou plusieurs seuils. k_i : d_i est inférieur à un seuil maximal ;$(d_i < s_i)$ avec $s_i \in [0..1]$.

La similarité ($\approx$) vérifier évidemment les propriétés de réflexivité, commutativité et d'associativité. C'est-à-dire $[\upsilon \approx \upsilon]$, $[(\upsilon \approx \upsilon^{'}) \Leftrightarrow (\upsilon^{'} \approx \upsilon)]$ et $[\upsilon \approx (\upsilon^{'} \approx \upsilon^{''}) \Leftrightarrow (\upsilon \approx \upsilon^{'}) \approx \upsilon^{''}]$.

Exemple 1 :

$Adr1 \leftarrow t1.$Adr et $Adr2 \leftarrow t2.Adr$; $Nom1 \leftarrow t1.$Nom et $Nom2 \leftarrow t2.$Nom ;$Mail1 \leftarrow t1.$Mail et $Mail2 \leftarrow t2.$Mail ; $Tel1 \leftarrow t1.$Tel et $Tel2 \leftarrow t2.$Tel.

Les deux adresses suivantes Adr1= "133 BD MARCEL EPINAY/SEINE" et Adr2= Adr2="133 BD MARCEL 93800 EPINAY-SUR-SEINE" sont similaires, selon la méthode de Jaro-Winkler car la distance calculée est de 0,057 (inférieure au seuil s=0,2).

Définition 2 :

Règle de similarité : Une règle r de similarité est une conjonction de similarités qui portent sur des attributs $A_{i(i=1;n)}$ de l'ensemble A de la table T : $r = (t.A_1 \approx t^{'}.A_1) \wedge (t.A_2 \approx t^{'}.A_2) \wedge (t.A_i \approx t^{'}.A_2) \cdots \wedge (t.A_n \approx t^{'}.A_n)$.

Exemple 2 :

Deux règles de similarités r_1 *et* r_2 : r1 = (Nom12)$\wedge$($Mail12$) $\wedge$ ($Adr12$),$r2 = (Mail1 \approx Mail2) \wedge (Tel1 \approx Tel2)$.

Définition 3 :

Similarité entre tuples : Deux tuples t et $t^{'}$ sont similaires ssi la disjonction de toutes les règles de similarité définies sur la table T est vraie. On note $t \approx t^{'}$ ssi $r_1 \ldots r_2 \vee \cdots \ldots r_k \cdots r_q$ est vraie avec q étant le nombre de règles de similarité.

Exemple 3 :
t1 et t2 sont similaires ssi $r_1 \vee r_2 : t1 \approx t2$ ssi$((Nom1 \approx Nom2) \wedge (Mail1 \approx Mail2) \wedge (Adr12)) \vee ((Mail1 \approx Mail2) \wedge (Tel1 \approx Tel2))$.

Remarque :
Ces règles peuvent présenter toutefois des incohérences ou des contradictions qui seront à vérifier.

4.6.1 Fonction Match

La fonction Match compare deux tuples t1 et t2 : Match$(t1, t2)$=Vrai ssi $t1 \approx t2$. L'étude de similarité porte sur les attributs de l'ensemble A selon les règles de similarité. La fonction Similaire $(v1, v2; k)$ permet de dire si deux valeurs v1 et v2 sont similaires selon une condition k. En effet, et selon le type des données, une des méthodes de calcul de distance de similarité, vue ci-dessus, est appliquée. Par exemple, pour les chaînes de caractères un algorithme tel que celui de Levenshtein ou Jaro-Winkler sera déclenché. La distance trouvée d est comparée au seuil s donné (exemple de condition $k : d < s$). L'algorithme est présenté ci-dessous (TABLE 4.1).[84]

Algorithme Match
Input : Two tuples, t1 and t2 $\in T$;
S (thresholds)
Output : Result := True if $t1 \approx t2$
Begin
Result := True
For all rj j from 1 to q Do
$Rule_j$:= True ; i :=1
While $Rule_j$ and $i \leqslant n$ **Do**
v1 :=t1.Ai ; v2 :=t2.Ai
If v1 or v2 = NULL Then Res = True
Else Result = Similar (v1,v2, si) **End If**
$Rule_j = Rule_j$ and Result ; i :=i+1
End While
Result := Result or $Rule_j$
End for
End Algorithm Match

TABLE 4.1 – Algorithm de Match

4.6.2 Fonction Merge

La fonction Merge traite deux tuples, supposés similaires, tel que Match(t1,t2)=Vrai. Elle retourne un nouveau tuple qui est le résultat de la fusion des deux précédents. La fonction Merge retourne un tuple qui apporte "plus" d'information. Un enrichissement des données peut être réalisé à la demande de l'utilisateur sur les attributs de l'ensemble A, c'est-à-dire ceux qui servent pour étudier la concordance (match). Le principe général de l'algorithme de fusion de deux tuples est donné ci-dessous (TABLE 4.2). La fonction fusion permet de combiner deux valeurs selon le type de donnée de l'attribut en question :
La fonction Merge traite deux tuples, supposés similaires, tel que Match(t1,t2)=Vrai. Elle retourne un nouveau tuple qui est le résultat de la fusion des deux précédents. La fonction Merge retourne un tuple qui apporte "plus" d'information. Un enrichissement des données peut être réalisé à la demande de l'utilisateur sur les attributs de l'ensemble A, c'est-à-dire ceux qui servent pour étudier la concordance (match). Le principe général de l'algorithme de fusion de deux tuples est donné ci-dessous (tab 1). La fonction fusion permet de combiner deux valeurs selon le type de donnée de l'attribut en question :

- si l'attribut est du type chaîne de caractères, *expr* présente la plus longue chaîne des deux, si celle-ci vérifie un ensemble d'expressions régulières proposé pour l'utilisateur. La fusion d'une chaîne quelconque avec une valeur nulle (NULL) donne la chaîne elle-même ;

- si l'attribut est du type numérique, la fusion peut correspondre à des calculs demandés par l'utilisateur tels que la somme, la moyenne, le minimum ou le maximum des valeurs. Des transformations peuvent êtres réalisées telles que des conversions dues à des changements d'unité. Les valeurs nulles sont considérées égales à 0 ;

- si l'attribut est du type date, la fusion peut permettre de garder la plus récente des deux, selon le choix de l'utilisateur, après avoir unifié les formats et vérifié leurs validités , L'algorithme est présenté ci-dessous (TABLE 4.2). [30].

Algorithm Merge
Input : Two tuples, t1 and t2 T **Output :** A tuple t **Begin For all** Attribute Ai in A i from 1 to n **Do** v1 :=t1.Ai ; v2 :=t2.Ai v :=fusion (expr, v1, v2) t.Ai :=v **End for** **End** Algorithm Merge

TABLE 4.2 – Algorithm de Merge

4.7 Elimination des similaires

Le principe de l'élimination des données similaires consiste à croiser tous les tuples d'une table. Il s'agit donc d'algorithmes très coûteux pour de gros volumes de données. L'idée est de réduire le nombre de comparaisons en fusionnant les données qui concordent. Pour entamer cette étude, nous nous sommes inspirés des trois algorithmes présentés dans (Benjallounet al., 2009). Dans ce qui suit, nous présentons et comparons ces trois algorithmes modifiés.
Le principe de partitionnement des tables dans les SGBD est utilisé afin d'améliorer leurs performances, plusieurs versions sont donc disponibles[30].

4.7.1 Algorithme G

L'algorithme G est séquentiel (TABLE 4.3). Il permet d'éliminer les similaires (ou quasi doublons) d'un ensemble de tuples grâce à la "boite noire" composée des deux fonctions Match et Merge.
L'algorithme fonctionne à l'aide de deux ensembles de tuples I et I'. I est l'ensemble des tuples initiaux qui n'ont pas encore été comparés, et I' contient le résultat final.
Tout tuple t de I est comparé à tous ceux de I'. Dans le cas où t correspond à t' de I', un nouveau tuple t"(résultat de la fusion) est créé et inséré dans I. Ensuite, le tuple dominée t' est supprimé de I'.
Dans le cas inverse, t est ajouté à I'. De nombreuses versions de cet algorithme sont mises en oeuvre en utilisant le concept de partitionnement du SGBD Oracle.
Les améliorations apportées à l'algorithme initial se résument comme suit :
(i) l'élimination des tuples dominés se fait dès qu'ils sont trouvés et non à la fin du processus, puisque les tuples dominés ne peuvent pas générer des tuples non dominés ; (ii) l'ajout de la condition t 6= t' pour le parcours des tuples de I' afin de satisfaire les propriétés d'ICAR (Idempotence, Commutativity, Associativity, Representativity) (Benjalloun et al., 2009) (celle de l'idempotence vu qu'il n'est pas nécessaire de faire correspondre t à lui-même)[30].

4.7.2 Mesures des performances :

(F.Boufarès et A.Ben Salem ; S.Correia) testé une table CLIENT dont la description SQL est la suivante :
"$CREATETABLECLIENT(PRENOM\ varchar2(100), NOMvarchar2(100), RUEvarchar2(100), CODEPOSTALvarchar2(10), VILLEvarchar2(50), DATNAISDate, TELvarchar2(100))$;". Les attributs Prenom, Nom, Dat- Nais et Tel ont été choisis pour l'élimination des similaires. Les algorithmes Jaro-winkler, Levenshtein et Jaccard ont été utilisés pour le calcul de la distance de similarité. Un seuil a été fixé pour chacun des attributs respectivement $(0,2; 0,2; 0; 0,1)$. L'utilisateur peut demander l'enrichissement de ses données afin d'éliminer, d'une part, d'éventuelles valeurs nulles, et d'autre part regrouper ou corriger certaines informations. Par exemple, dans la mesure où l'on veut avoir plusieurs numéros de téléphone pour un même client, la définition de l'attribut concerné (résultat de la fusion) devrait être suffisamment élargie pour mettre x valeurs au lieu d'une seule ($Telvarchar2(100)$ au lieu de $Telvarchar2(10)$). De ce fait, dans le cas où l'utilisateur demande à enrichir ses données, il lui suffit de fixer le nombre maximal x d'occurrences des valeurs pour les attributs concernés. La variable x peut être fixée par l'utilisateur selon une règle métier. Dans notre exemple, nous avons fixé x à 3 pour l'attribut Tel. Ainsi, on pourra avoir au plus trois numéros de téléphone pour chaque client après exécution de l'algorithme d'élimination des similaires. Deux règles de similarité entre tuples ont été utilisées pour réaliser la fusion (Merge) :
$\mathrm{R}_{ules} = r_1 \vee r_2 avec r_1 : (d1 \leq s1) \wedge (d2 \leq s2) \wedge (d7 \leq s7); r_2 : (d2 \leq s2) \wedge (d6 \leq s6)$

```
Algorithme G
input : a set I of tuples
output : a set I' of tuples
I' → ∅
while I ≠ ∅ do
 t → a tuple from I
 remove t from I
 If I' ≠ ∅ then
  for all tuples t' in I' and t ≠ t' do
   if Match (t, t') then
    t" → Merge (t, t')
    if t" ∉ I U I' U t then
    add t" to I
    endif
   remove t' from I'
   endif   endfor
 enfif
 add t to I'
endwhile
return I'
```

TABLE 4.3 – Algorithme G

4.7.3 Mesures des performances pour l'algorithme G [30] :

	G		
Milliers de lignes	10	20	50
des tuples éliminés	19,79	16,5	59 ,4
des tuples similaires	19,79	16,5	59 ,4

TABLE 4.4 – Mesures des performances[30]

4.8 Conclusion

Actuellement, un grand nombre d'applications utilisent des données hétérogènes et distribuées de qualité variable. Le besoin d'intégration de données et d'évaluation de la qualité des données se fait de plus en plus ressentir. Afin de redonner d'avantage de sens dans les données rassemblées, nous abordons la problématique complexe de l'élimination de similaires . En effet, le développement de nouveaux outils pour traiter les données similaires nécessite l'approfondissement, d'une part, des deux concepts de base **Match & Merge**,et d'autre part, le développement d'autres algorithmes

pour parcourir et comparer les données dans des environnements séquentiels . Les types de données traités dans notre étude sont les types caractère, numérique, date et logique. La concordance des données se fait selon des règles proposées par l'utilisateur.
Nous avons durant ce chapitre exposé l'algorithme G de déduplication. L'originalité de ce algorithme réside dans le fait de procéder à l'élimination des données redondantes au sens strict et au sens large. Les fonctions Match et Merge sont appliquées simultanément. On traite de données de types caractère, numérique et date. La concordance des données se fait selon des combinaisons de règles proposées par l'utilisateur, celles-ci devront être dynamiques et réutilisables. La sémantique des données traitées est prise en compte dans les fonctions Match et Merge.
En éliminant les doublons et les similaires dans une source de données, nous veillons à valider la dimension Duplication de la qualité de données . Cette dimension assure la présence de données non répétées dans la source et donc une meilleure qualité.
La dimension Complétude est assurée avec la fonction Merge que nous proposons. Elle permet l'enrichissement des données et ainsi traiter les valeurs nulles.

Chapitre 5

Résultat et Discution

5.1 Introduction

Après avoir élaboré la conception de notre projet, nous abordons dans ce chapitre le dernier volet de ce rapport, qui a pour objectif d'exposer la phase de quelques interfaces réalisées pour illustrer le fonctionnement de quelques activités du système.puis nous présentons les résultas de l'implimentation de l'algorithme G.

5.2 Description de l'application

- chargement de source de données.
- appliquer des différents traitements ,sur notre source de données.
- appliquer un algorithme qui permet d'éliminer les doubles et les similaires dans une SD.

5.3 Objectif :

Pour mettre en place notre application, nous avons réalisé une application sous l'environnement NetBeans qui utilise l'algorithme G offert via une interface facile à utilisé. Donc, notre application appel les packages et les différentes classes nécessaire NetBeans dans un programme Java afin d'assurer les fonctionnalités de prédiction fixés par notre étude.

l'algorithme d'élimination des doubles et des similaires dans une SD utilisés dans notre application est :

- Algorithme G

5.4 Utilisation du package JDBC.jar

5.4.1 Définition du package JDBC.jar

La technologie JDBC (java data base connectivity)est une API fourni avec Java permettant de se connecter à des bases de données.

5.4.2 Importation du package JDBC.jar

Sous NetBeans, si on veut créez un projet qui aura besoin des classes de JDBC, on doit indiquer ca par l'importation du package JDBC.jar dans le programme Java.
Le package JDBC.jar est accessible et téléchargeable gratuitement et il suffit de l'ajouter à la bibliothèque de l'application comme montre la figure suivant :

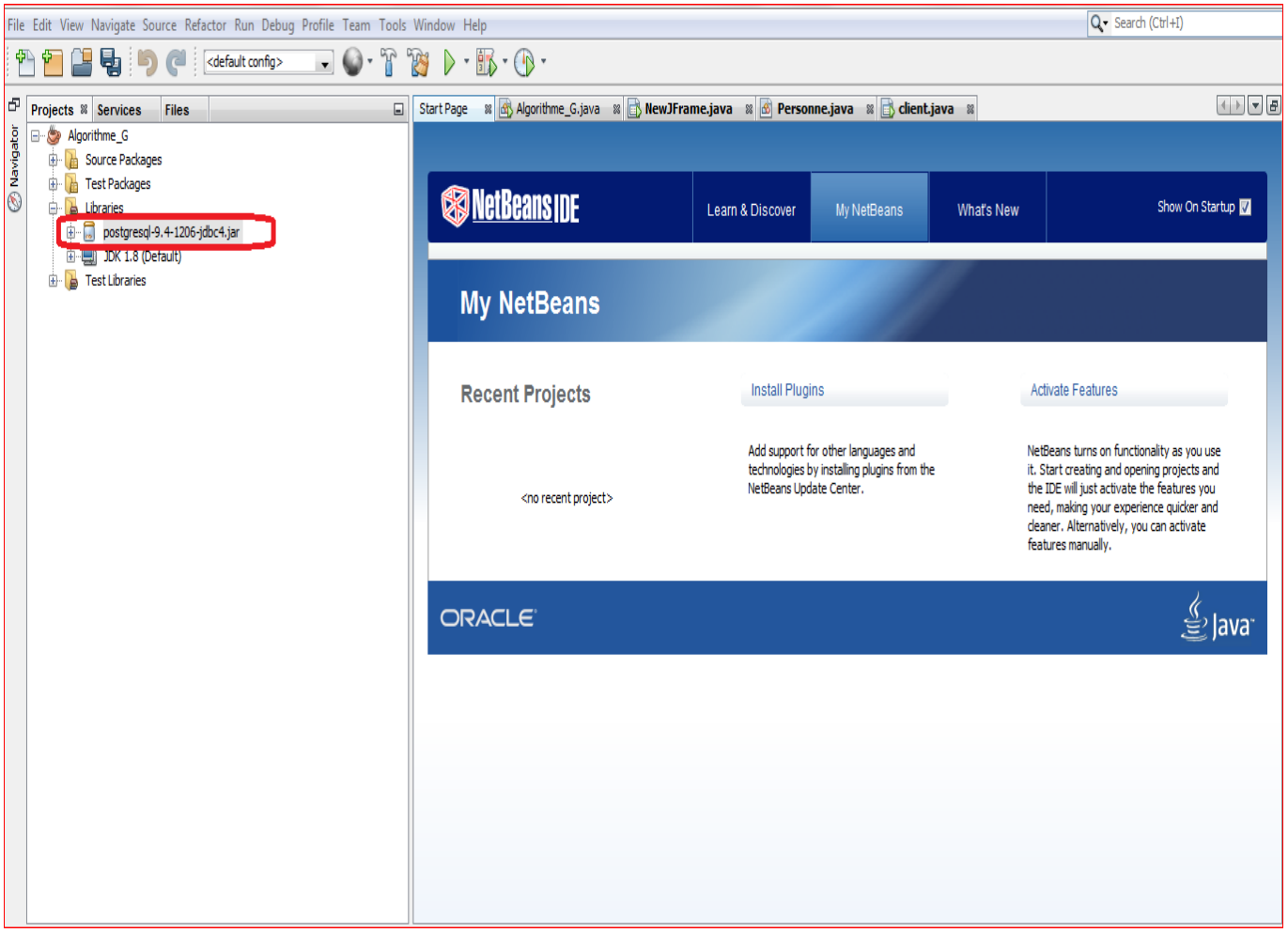


FIGURE 5.1 – package JDBC.jar

5.5 Utilisation d'une source de données

Notre source de données n'est pas une base réelle, alors c'est une SD virtuelle.

ID	Nom	Prénom	Téliphone	Mail
1	Le Bon	Adam	0666600007	LeBon_Adam...
4	Grande	Clèmence	0666688887	grande@yaho...
5	Adam	Le Bon	0666677777	AdamLeBon...
5	Adam	Le Bon	0666677777	AdamLeBon...
7	LeBon	Adeline	0666611117	LeBon_Adeli...
2	Le Bon	A.	0666677777	LeBon_A.@y...
3	Le B.	Adam	0666677777	LeB_Adam@...
5	Adam	Le Bon	0666677777	AdamLeBon...
7	Unique	Eve	0666622227	unique@gmai...
7	LeBon	Adeline	0666611117	LeBon_Adeli...
6	Unique	Eve	0666622227	unique@gmai...

FIGURE 5.2 – Notre source de données

5.6 Les interfaces de l'application développée

Nous présentons dans cette section des captures d'écran de l'application développée.

5.6.1 L'Interface d'accueil :

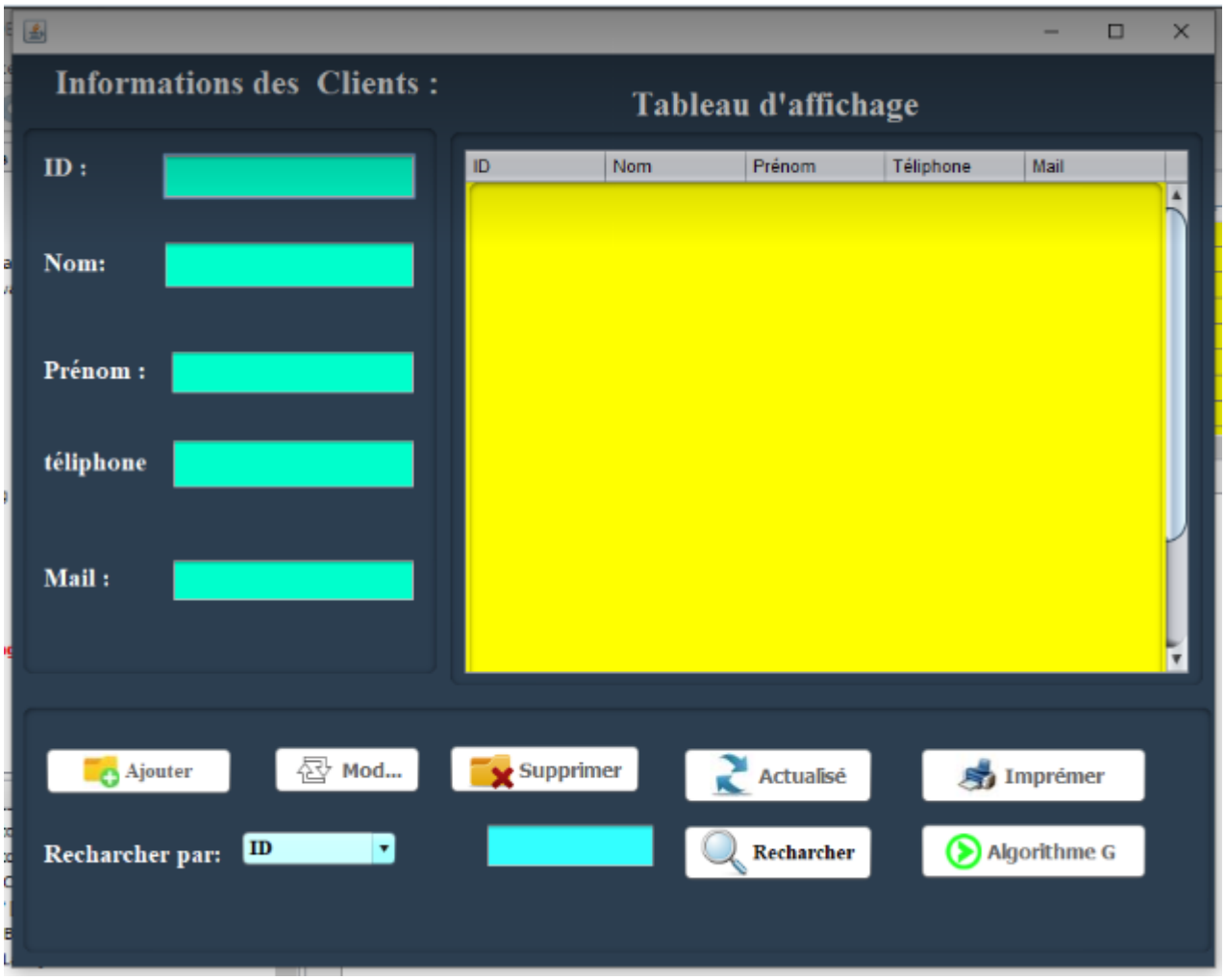


FIGURE 5.3 – Fenêtre principale de l'application.

— Le bouton Ajouter permet d'ajouter un nouveau Client dans notre source de données.
— Le bouton Modifier permet de modifier les données d'un Client existe déja dans notre SD.
— Le bouton Supprimer permet de supprimer 'un Client existe déja dans notre SD.
— Le bouton Actualiser permet d'actualiser notre SD.
— Le bouton Rechercher permet de rechercher un Client existe dans notre SD apartire de leur ID .
— Le bouton Imprimer permet d'imprimer notre SD.
— Le bouton Algorithme G permet de passe ver la 2eme interface (FIGURE 5.2).
— La case ID permet d'entrer un ID pour ajouter un nouveau Client.
— La case Nom permet d'entrer un Nom pour ajouter un nouveau Client.
— La case Prénom permet d'entrer un Prénom pour ajouter un nouveau Client.
— La case Télephone permet d'entrer un numéro de telephone pour ajouter un nouveau

Client.

— La case mail permet d'entrer une adresse E-mail pour ajouter un nouveau Client.

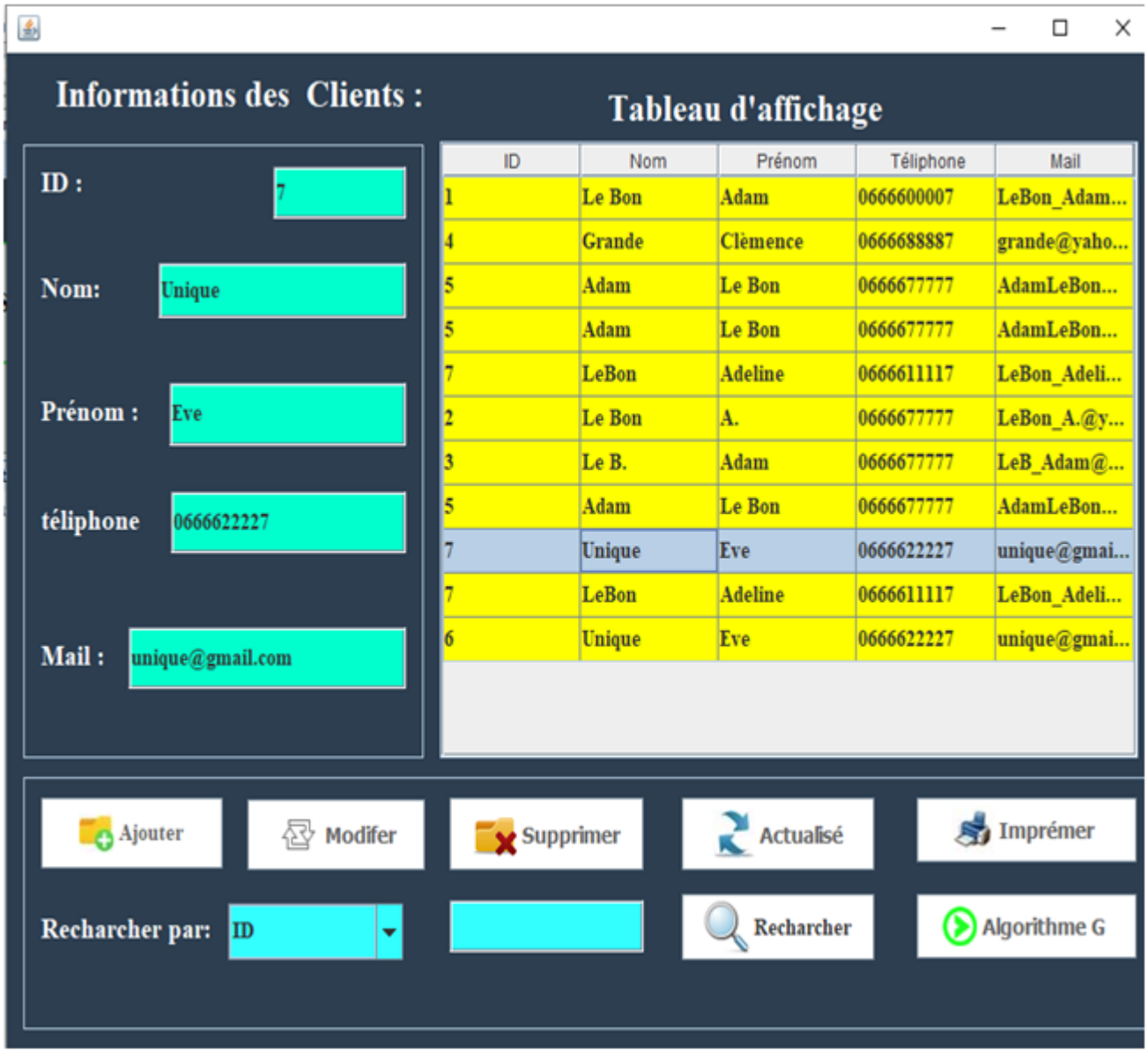


FIGURE 5.4 – entré les données dans une source de donnée.

5.7 interface d'Algorithme G

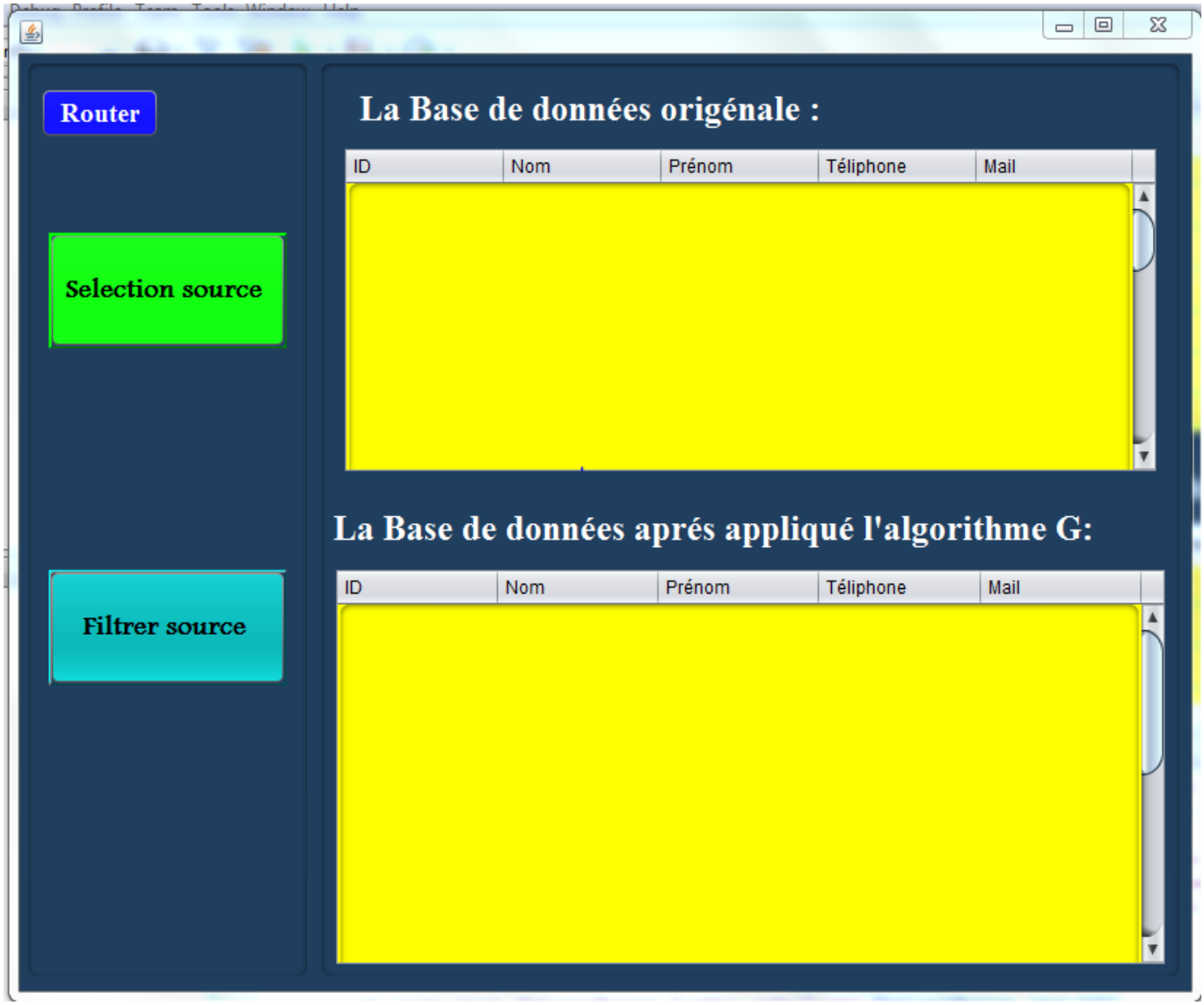


FIGURE 5.5 – interface Algorithme G

— Le bouton Retour de retour a l'interface précédent.
— Le bouton Selectioner source permet d'afficher notre SD.
— Le bouton filtrer source permet d'appliquer l'algorithme G sur notre SD.

5.7.1 Selectioner source

Quand on cliquons sur le bouton Selectioner , j'obtiens les résultats suivants (FIGURE 5.4)

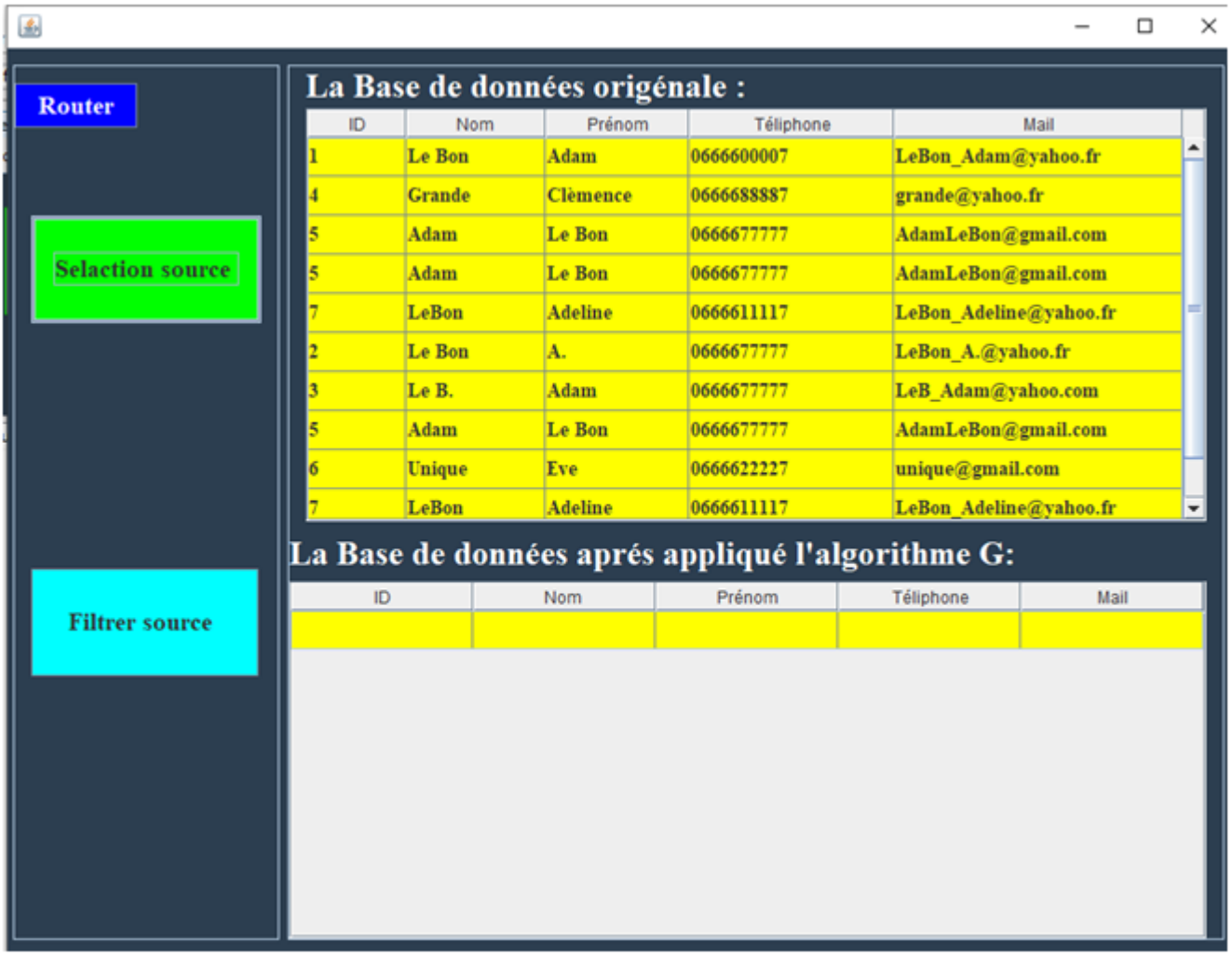


FIGURE 5.6 – Résultat de Selectioner source source

dans cette étape ,on a comme résultat une source de données non filtré contiene plusieure données doubles et similaires.

5.7.2 Filtrer source

Quand on cliquons sur le bouton Filtrer source , j'obtiens les résultats suivants (FIGURE 5.5)

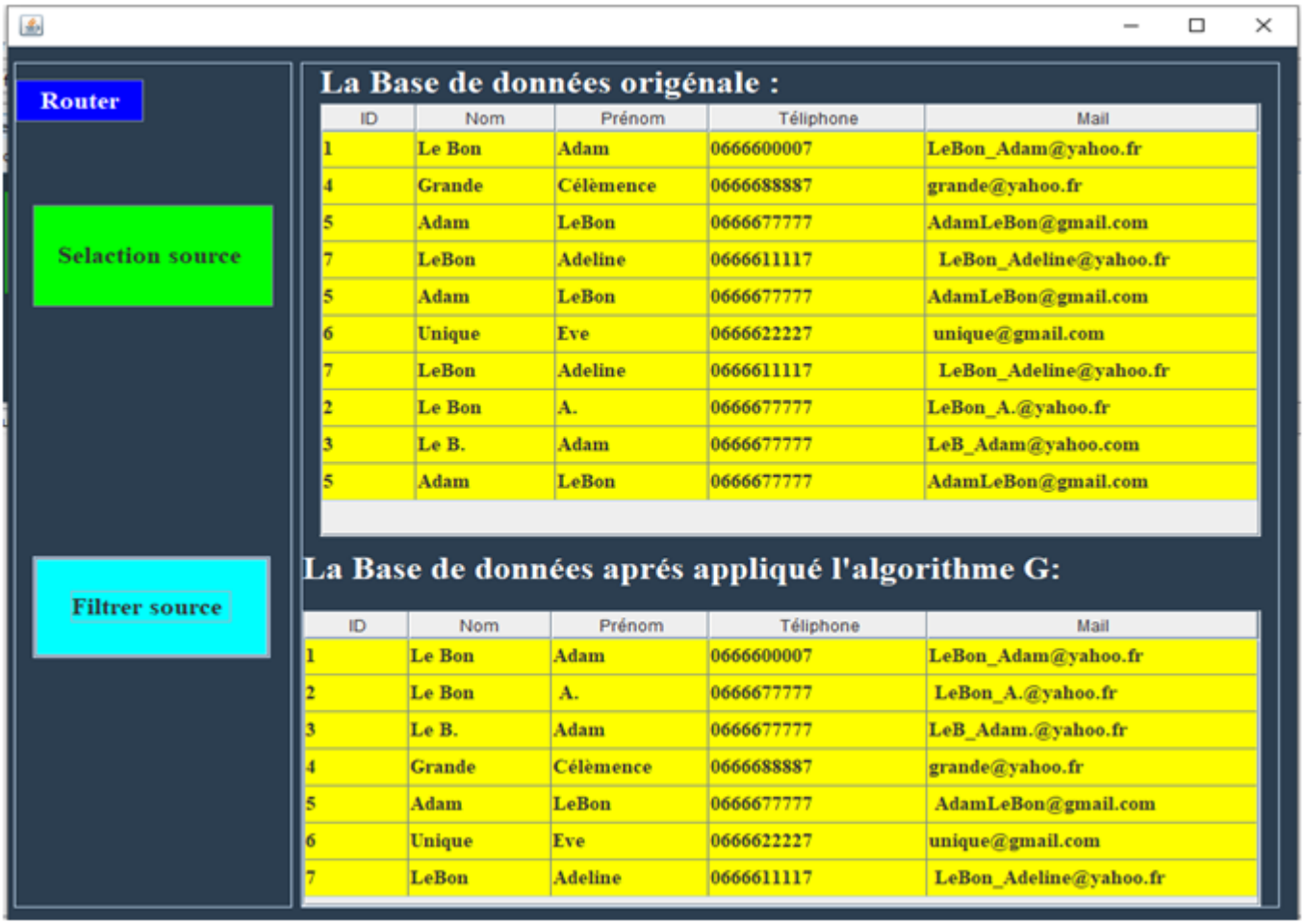


FIGURE 5.7 – Résultat de Filtrer source

dans cette étape ,on a comme résultat une source de données filtré ne contienne pas les données doubles et similaires.

5.8 Conclusion

À la fin de ce chapitre, nous avons réussi à présenter les résultats obtenus de l'implémentaion de l'algorithme G utilisé ainsi que les algorithmes Merge Match en relation entre ils, Nous disposons ainsi des interfaces de l'application développée de notre algorithme pour bien illustrer le travail qui a été fait.
D'après les résultats des algorithmes , nous avons remarqué que des tests de validation ont été concluant .
Enfin, nous avons constaté que notre application d'algorithme G est efficace pour permet de conaitre les données sans doublons et des similaires.

Chapitre 6

Conclusion Génirale

De nombreux cas décrits dans la littérature et dans la presse scientifique, révèlent de nombreuses situations alarmantes liées aux enjeux de la qualité des données dans les bases, entrepôts de données et systèmes d'information commerciaux, médicaux, du domaine public ou de l'industrie. Les approches jusqu'ici mises en oeuvre sont le plus souvent ad hoc, fragmentées et spécifiques à des domaines d'application relativement cloisonnés. Des solutions théoriquement fondées et validées en pratique sont aujourd'hui très attendues pour évaluer et contrôler la qualité des données. Plusieurs verrous scientifiques dans ce domaine ont été identifiés et ils expliquent d'ailleurs les limitations des outils actuellement disponibles. A titre indicatif, en voici les principaux :
- l'hétérogénéité et la diversité des données multi-sources à intégrer : les données sont extraites de différentes sources d'informations puis intégrées alors qu'elles possèdent différents niveaux d'abstraction (d'une donnée brute à un agrégat). Les données sont intégrées au sein d'un même jeu de données qui contient donc potentiellement une superposition de plusieurs traitements statistiques. Ceci nécessite une très grande précaution dans l'analyse de ces données. Les jointures entre les différents jeux de données sont difficiles (quand elles ne sont pas totalement faussées) par le problème de l'identification non ambiguë des données et l'impact des données manquantes,

- la dynamique des données et de leur qualité : les bases de données modélisent une portion du monde réel en constante évolution. Or, dans le même temps, la qualité des données peut devenir obsolète. Le processus d'estimation et de contrôle doit être constamment reconduit en fonction de la dynamique du monde réel modélisé ainsi que des changements des centres d'intérêt ou de l'attention particulière portée à certaines données (car des données jugées critiques à un instant donné ne le restent pas indéfiniment)...

Pour conclure, les multiples problèmes évoqués dans ce dossier offrent plus que jamais d'intéressantes perspectives de recherche pour les différentes communautés scientifiques travaillant autour de la qualité des données en Statistiques, Bases de Données, Ingénierie des Connaissances et Gestion de Processus. Pour les entreprises et industriels, détenteurs et garants de leurs masses de données, la qualité des données demeure un problème récurrent, les guidant ponctuellement vers des choix pragmatiques, souvent à court terme par manque de moyens et d'appuis hiérarchiques. Il est alors

clair que pour apporter des solutions concrètes, opérationnelles sur le long terme et théoriquement fondées, des collaborations étroites entre le monde académique et les industriels sont nécessaires.

bibliographie

[**2**] Alain GAUGRIS *Améliorer la qualité des statistiques du commerce de distribution*,Bamako, Mali,17-20 juin 2008.

[**4**] Aicha Ben Salem *Qualité contextuelle des données : détection et nettoyage guidés par la sémantique des données.*,Université Sorbonne Paris Cité,2015.

[**5**] Houda Zaidi *Amélioration de la qualité des données : correction sémantique des anomalies intercolonnes. Base de données.*.Conservatoire national des arts et metiers - CNAM ; École Nationale des Sciences de l'Informatique (La Manouba, Tunisie),2017.

[**6**] Daouda Traoré. *Algorithmes parallèles auto-adaptatifs et applications. Informatique* . Institut National Polytechnique de Grenoble - INPG, 2008. Français.

[**8**] Pierre Delisle .*« Parallélisation d'un algorithme d'Optimisation par Colonies de Fourmis pour la résolution d'un problème d'ordonnancement industriel »* ,UNIVERSITÉ DU QUÉBEC À CHICOUTIMI COMME EXIGENCE PARTIELLE DE LA MAÎTRISE EN INFORMATIQUE , 6 juin 2002.

[**9**] Chafik A. *« Chapitre 3 : Les architectures parallèles »*.

[**11**] Aurélia Marchand. *« Les architectures parallèles - MPI »*. 2010.

[**12**] Abderrahimi Hidayet khadidja, Benali Amar samia *« Algorithmique génétique parallèle appliqué a un problème d'optimisation combinatoire»*, Mémoire de Master, Informatique, 2016-2017.

[**13**] I. FOSTER *Designing and Building Parallel Programs.* Addison–Wesley, 1995.

[**14**] M.-J. *QUINN : Parallel Programming in C with MPI and OpenMP.* McGraw–Hill, 2003.

[**17**] K. SCHOEGEL, G. KARYPIS et V. KUMAR . *Multilevel diffusion schemes for repartitioning of adaptive meshes*. Rapport technique 97-013, Department of Computer Science, University of Minnesota, Minneapolis, 1997.

[**18**] G. KARYPIS et V. KUMAR . *A coarse–grain parallel formulation of multilevel k–way graph partitioning algorithm*. In Eigth SIAM Conference on Parallel Processing for Scientific Computing, Philadelphia, 1997.

[**19**] G. KARYPIS et V. KUMAR *A coarse–grain parallel formulation of multilevel k–way graph partitioning algorithm*. In Eigth SIAM Conference on Parallel Processing for Scientific Computing, Philadelphia, 1997.

[**21**] M. Éli LAUCOIN *« Développement du parallélisme des méthodes numériques adaptatives pour un code industriel de simulation en mécanique des fluides»*, Thèse doctorat dans le cadre de l'École Doctorale : «Mathématiques, Sciences et Technologies de l'Information, Informatique » , Spécialité : «Mathématiques appliquées », UNIVERSITÉ JOSEPH FOURIER GRENOBLE 1 , Thèse soutenue le 24 octobre 2008 .

[**23**] Philip Resnik, *Mining the Web for bilingual text*, In proceedings of ACL 1999.

[**24**] Philip Resnik and Noah A. *Smith, The Web as a parallel corpus*, Computational Linguistics, vol. 29, no. 3, 2003.

[**25**] Zhao B., Vogel S., *Adaptive parallel sentences mining from Web bilingual news collection*, In proceedings of IEEE International Conference on Data Mining 2002.

[**26**] P. Fung, P. Cheung, *Mining very-non-parallel corpora : parallel sentence and lexicon extraction via bootstrapping and EM*. In proceedings of EMNLP, 2004.

[**27**] Munteanu D.S., Marcu D. *Improving machine translation performance by exploiting non-parallel corpora*, Computational Linguistics 2005.

[**28**] Munteanu D.S., Marcu D. *Extracting parallel subsentential fragments from non-parallel corpora*, In proceedings of ACL 2006.

[**29**] Quirk C., Udapa R., *Menezes A. Generative models of noisy translations with applications to parallel fragment extraction*, In proceedings of MT Summit, EAMT 2007.

[**30**] Faouzi Boufarès, Aïcha Ben Salem ; Sebastiao Correia *Qualité de données dans les entrepôts de données : élimination des similaires* Laboratoire LIPN - UMR 7030 -

CNRS, Université Paris 13 Av. J. B. Clément 93430 Villetaneuse France.

Réfirance de web :

[1] http://www.asprom.com/technologie/informica.pdf

[3] https://fr.wikipedia.org/wiki/Type_(informatique).

[7] https://fr.qwe.wiki/wiki/Parallel_algorithm.

[10] https//www.irif.fr/_media/users/pierref/notes_algo_distribue.pdf

[15] https://cosy.univ reims.fr/~pdelisle/fichiers/info0802_1314/Cours2_Info0802.pdf

[17] https//tel.archives-ouvertes.fr/tel-00385418/document

[12] https//tel.archives-ouvertes.fr/tel-00680046/document

[31] https://fr.wikipedia.org/wiki/Java_(langage)

[32] https://fr.wikipedia.org/wiki/NetBeans

[33] https://fr.wikipedia.org/wiki/PostgreSQL

Annexes

Annexes

6.1 Classe Algorithme G

```
package algorithme_g;
import java.util.LinkedList;
import java.util.List;
public class Algorithme_G {
       public static void main(String[] args) {
        // TODO code application logic here
        List<Personne> Input = new LinkedList<Personne>();
               // Test values
               Input.add(new Personne(1,"Le Bon","Adam","0666600007","LeBon_Adam@yahoo.fr"));
               Input.add(new Personne(2,"Le Bon ","A.","0666677777","LeBon_A.@yahoo.fr"));
               Input.add(new Personne(3,"Le B."," Adam ","0666677777","LeB_Adam@yahoo.fr"));
               Input.add(new Personne(4,"Grande","Clémence","0666688887","grande@yahoo.fr"));
               Input.add(new Personne(5,"Adam","LeBon","0666677777","AdemLeBon@gmail.com"));
               Input.add(new Personne(5,"Adam","LeBon","0666677777","AdemLeBon@gmail.com"));
               Input.add(new Personne(6,"Unique","Eve","0666622227","unique@gmail.com"));
               Input.add(new Personne(7,"LeBon","Adeline","0666611117","LeBon_Adeline@yahoo.fr"));
               Input.add(new Personne(5,"Adam","LeBon","0666677777","AdemLeBon@gmail.com"));
               Input.add(new Personne(6,"Unique","Eve","0666622227","unique@gmail.com"));
               Input.add(new Personne(7,"LeBon","Adeline","0666611117","LeBon_Adeline@yahoo.fr"));
               List <Personne> All = Algorithme_G(Input);
               for(int n=0;n<All.size();n++)
               System.out.print(  "id: "+All.get(n).id+ ", Nom: "+All.get(n).nom+", Prenom: "+All.get(n).prenom +","
                       + " tlephone: "+All.get(n).tlephone +", mail:"+All.get(n).mail +"\n");
               }
 public static List <Personne> Algorithme_G(List<Personne> Input){ // input (set of i tuples)
        List<Personne> Output = new LinkedList<Personne>(); //set of i_prime tuples
        while (Input.size()!=0) {
            Personne t=Input.get(0);
            Input.remove(0); // remove t from i (from Input)
                if (!Output.isEmpty()) {
                    for(int n=0;n<Output.size();n++) {
                        Personne t_prime=Output.get(n);
                            if (Match(t,t_prime)) { // compare and match them
                                Personne t_2prime=Merge(t,t_prime); // merge fields
                                if (!Input.contains(t_2prime) && !Output.contains(t_2prime) && !Match(t_2prime,t)) {
                                     Input.add(t_2prime);
                                        }
                                        Output.remove(t_prime);
                                                }
                                                    }
                                        }
                    Output.add(t); //add t to i_prime (To the Output)
                                        }
        return Output;
 }
```

```
public static Boolean Match(Personne Tuple1, Personne Tuple2){

    Boolean Result=true;
        //3 => le nombre des attributs dans la classe personne (il faut l'implementer selon la classe)

        //for(int n=0;n<3;n++) {
    Result= Tuple1.id == Tuple2.id && Tuple1.nom == Tuple2.nom && Tuple1.prenom == Tuple2.prenom &&
    Tuple1.tlephone == Tuple2.tlephone&&Tuple1.mail == Tuple2.mail ;
                //}
        return Result;
        }
    }
```

```
public static Personne Merge(Personne Tuple1, Personne Tuple2){
    // Algorithme de Merge içi ...
        int v_id=Tuple1.id;
        String v_nom=Tuple1.nom;//+Tuple2.nom;
        String v_prenom=Tuple1.prenom;//+Tuple2.prenom;
        String v_tlephone=Tuple1.tlephone;//+Tuple2.tlephone;
        String v_mail=Tuple1.mail;//+Tuple2.mail;
        Personne Merged=new Personne(v_id,v_nom,v_prenom,v_tlephone,v_mail);
        return Merged;
    }
```

6.2 La Classe Perssone.java

```
package algorithme_g;

public class Personne {
    /// Class attributes ...
        public String nom,prenom,mail;
        public String tlephone;
        int id;
        public Personne (   int id ,String nom, String prenom, String tlephone,String mail) {
                this.id=id;
            this.nom=nom;

                this.prenom=prenom;
                this.tlephone=tlephone;
                this.mail=mail;
        }

}
```

6.3 La Classe NewJFrame.java

```
package algorithme_g;

import java.util.LinkedList;
import java.util.List;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class NewJFrame extends javax.swing.JFrame {
    PreparedStatement st=null;
    ResultSet rs=null;
    private Object con=null;
       public NewJFrame() {
        initComponents();
           this.setLocationRelativeTo(null);
    }

         Connection connection;
String url = "jdbc:postgresql://localhost:5432/Client";
String user = "postgres";
String password = "0000";
    Generated Code

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
       client etu2 = new client();
        etu2.setVisible(true);
        this.setVisible(false);
        etu2.pack();  // TODO add your handling code here:
    }

    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      try {
          Connection connexion = DriverManager.getConnection(url, user, password);
           java.sql.Statement st = connexion.createStatement();
           String sql=
                   "SELECT id, nom, prenom, teliphone, mail FROM cliente";
           ResultSet result = st.executeQuery(sql);
           DefaultTableModel  dt = (DefaultTableModel )jTable2.getModel();
            dt.setRowCount(0);
            while(result.next())
            {
                Object o[]={result.getInt("id"),result.getString("nom"),result.getString("prenom"),
                    result.getString("teliphone"),result.getString("mail")};
                dt.addRow(o);
              }
            }
        catch(Exception e)
        { JOptionPane.showMessageDialog(this,e);
        } // TODO add your handling code here:
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
   // TODO add your handling code here:
    }
     /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
         Look and feel setting code (optional)
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new NewJFrame().setVisible(true);
            }});
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTable jTable1;
    private javax.swing.JTable jTable2;
    // End of variables declaration
```

6.4 La Classe client.java

```
package algorithme_g;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.MessageFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

public class client extends javax.swing.JFrame {
private DefaultTableModel dt;
    private  int id;
    private String nom,prenom;
    private String teliphontion,mail;
    PreparedStatement st=null;
    ResultSet rs=null;
    private Object con=null;
 public client() {
        initComponents();
        try{
            Connection connexion = DriverManager.getConnection(url, user, password);
            java.sql.Statement st = connexion.createStatement();
            String sql=
                    "SELECT id, nom, prenom, teliphone, mail FROM cliente";
            ResultSet result = st.executeQuery(sql);
            DefaultTableModel  dt = (DefaultTableModel )jTable2.getModel();
             dt.setRowCount(0);
             while(result.next())
             {
                 Object o[]={result.getInt("id"),result.getString("nom"),result.getString("prenom"),
                     result.getString("teliphone"),result.getString("mail")};
                 dt.addRow(o); }}
            catch(Exception e){
             JOptionPane.showMessageDialog(this,e); }
        this.setLocationRelativeTo(null);}
    Connection connection;
String url = "jdbc:postgresql://localhost:5432/Client";
String user = "postgres";
String password = "0000";
    Generated Code

    private void btn1MouseEntered(java.awt.event.MouseEvent evt) {
        btn1.setBackground(new java.awt.Color(102,255,102) );
    }

    private void btn1MouseExited(java.awt.event.MouseEvent evt) {
        btn1.setBackground(new java.awt.Color(255,255,255) );
    }

    private void btn1ActionPerformed(java.awt.event.ActionEvent evt) {
        try{
             Class.forName("org.postgresql.driver");
             Connection connexion = DriverManager.getConnection(url, user, password);
             String sql = "INSERT INTO cliente (id,nom,prenom,teliphone,mail)VALUES (?,?,?,?,?)";
              ResultSet result = st.executeQuery(sql);
             // ps.setString(9, jTextField1.getText());
             st.setString(1, jTextField1.getText());
             st.setString(2, jTextField2.getText());
            st.setString(3,  jTextField3.getText());
             st.setString(4,  jTextField4.getText());
             st.setString(5,  jTextField5.getText());
           st.executeUpdate();
             connexion.close();
             int number = jTable2.getSelectedRow();
             jTextField1.setText(jTable2.getValueAt(number,0).toString());
             jTextField2.setText(jTable2.getValueAt(number,1).toString());
             jTextField3.setText(jTable2.getValueAt(number,2).toString());
             jTextField4.setText(jTable2.getValueAt(number,3).toString());
              jTextField5.setText(jTable2.getValueAt(number,5).toString());
             // recharger la liste des stagiaires
         }catch(Exception e){
             JOptionPane.showMessageDialog(null," le  Client  est ajouté "+e.getMessage()); }
        try{ Connection connexion = DriverManager.getConnection(url, user, password);
           java.sql.Statement st = connexion.createStatement();
           String sql=
                   "SELECT id, nom, prenom, teliphone, mail FROM cliente";
           ResultSet result = st.executeQuery(sql);
            DefaultTableModel  dt = (DefaultTableModel )jTable2.getModel();
            dt.setRowCount(0);
```

```
        try{
            Connection connexion = DriverManager.getConnection(url, user, password);
            java.sql.Statement st = connexion.createStatement();
            String sql=
                    "SELECT id, nom, prenom, teliphone, mail FROM cliente";
            ResultSet result = st.executeQuery(sql);
            DefaultTableModel  dt = (DefaultTableModel )jTable2.getModel();
             dt.setRowCount(0);
             while(result.next())
             {
                 Object o[]={result.getInt("id"),result.getString("nom"),result.getString("prenom"),
                     result.getString("teliphone"),result.getString("mail")};
                 dt.addRow(o); }}
            catch(Exception e){
             JOptionPane.showMessageDialog(this,e); }
         this.setLocationRelativeTo(null);}
    Connection connection;
String url = "jdbc:postgresql://localhost:5432/Client";
String user = "postgres";
String password = "0000";
    Generated Code

    private void btn1MouseEntered(java.awt.event.MouseEvent evt) {
        btn1.setBackground(new java.awt.Color(102,255,102) );
    }

    private void btn1MouseExited(java.awt.event.MouseEvent evt) {
        btn1.setBackground(new java.awt.Color(255,255,255) );
    }

     private void btn1ActionPerformed(java.awt.event.ActionEvent evt) {
        try{
             Class.forName("org.postgresql.driver");
             Connection connexion = DriverManager.getConnection(url, user, password);
             String sql = "INSERT INTO cliente (id,nom,prenom,teliphone,mail)VALUES (?,?,?,?,?)";
              ResultSet result = st.executeQuery(sql);
             // ps.setString(9, jTextField1.getText());
             st.setString(1, jTextField1.getText());
             st.setString(2, jTextField2.getText());
            st.setString(3,  jTextField3.getText());
             st.setString(4,  jTextField4.getText());
             st.setString(5,  jTextField5.getText());
           st.executeUpdate();
              connexion.close();
             int number = jTable2.getSelectedRow();
             jTextField1.setText(jTable2.getValueAt(number,0).toString());
             jTextField2.setText(jTable2.getValueAt(number,1).toString());
             jTextField3.setText(jTable2.getValueAt(number,2).toString());
             jTextField4.setText(jTable2.getValueAt(number,3).toString());
              jTextField5.setText(jTable2.getValueAt(number,5).toString());
             // recharger la liste des stagiaires
         }catch(Exception e){
             JOptionPane.showMessageDialog(null," le  Client  est ajouté "+e.getMessage()); }
        try{ Connection connexion = DriverManager.getConnection(url, user, password);
           java.sql.Statement st = connexion.createStatement();
           String sql=
                   "SELECT id, nom, prenom, teliphone, mail FROM cliente";
           ResultSet result = st.executeQuery(sql);
            DefaultTableModel  dt = (DefaultTableModel )jTable2.getModel();
            dt.setRowCount(0);
           while(result.next())
           { Object o[]={result.getInt("id"),result.getString("nom"),result.getString("prenom"),result.
               dt.addRow(o); }}
        catch(Exception e)
        { JOptionPane.showMessageDialog(this,e);}
        jTextField1.setText("");
        jTextField2.setText("");
        jTextField3.setText("");
        jTextField4.setText("");
        jTextField5.setText("");
        jTextField7.setText("");

    }

    private void btn4MouseEntered(java.awt.event.MouseEvent evt) {
        btn4.setBackground(new java.awt.Color(0,135,102) );  // TODO add your handling code here:
    }

    private void btn4MouseExited(java.awt.event.MouseEvent evt) {
        btn4.setBackground(new java.awt.Color(255,255,255) );  // TODO add your handling code here:
    }

    private void btn4ActionPerformed(java.awt.event.ActionEvent evt) {
        jTextField1.setText("");
        jTextField2.setText("");
        jTextField3.setText("");
        jTextField4.setText("");
        jTextField7.setText("");
        jTextField5.setText("");
        // TODO add your handling code here:
```

```
    }

    private void btn3MouseEntered(java.awt.event.MouseEvent evt) {
        btn3.setBackground(new java.awt.Color(255,0,0) );
    }

    private void btn3MouseExited(java.awt.event.MouseEvent evt) {
        btn3.setBackground(new java.awt.Color(255,255,255) );  // TODO add your handling code here:
    }

    private void btn3ActionPerformed(java.awt.event.ActionEvent evt) {
            try{
             if(JOptionPane.showConfirmDialog(null, "attention est tu es sure que"
                 + " tu veux supprimer ce client",
                     "Supprimer etudiant",JOptionPane.YES_NO_OPTION)
                 == JOptionPane.OK_OPTION)
             Class.forName("org.postgresql.driver");
             Connection con = DriverManager.getConnection(url, user, password);
             String sql="DELETE  FROM cliente where id =? ";
             ResultSet result = st.executeQuery(sql);
             st.setString(1,jTextField1.getText());
             st.executeUpdate();
             con.close();
             JOptionPane.showMessageDialog(null,"Suppression succés");}
         catch(Exception e){  JOptionPane.showMessageDialog(this,e.getMessage());
              int number = jTable2.getSelectedRow();
             jTextField1.setText(jTable2.getValueAt(number,0).toString());
             jTextField2.setText(jTable2.getValueAt(number,1).toString());
             jTextField3.setText(jTable2.getValueAt(number,2).toString());
               jTextField4.setText(jTable2.getValueAt(number,3).toString());
              jTextField5.setText(jTable2.getValueAt(number,4).toString()); }
         try{
                 Connection connexion = DriverManager.getConnection(url, user, password);
             java.sql.Statement st = connexion.createStatement();
             String sql=
                     "SELECT id, nom, prenom, teliphone, mail FROM cliente";
             ResultSet result = st.executeQuery(sql);
             DefaultTableModel  dt = (DefaultTableModel )jTable2.getModel();
              dt.setRowCount(0);
              while(result.next())
              {
                  Object o[]={result.getInt("id"),result.getString("nom"),
                      result.getString("prenom"),result.getString("teliphone"),result.getString("mail")};
                  dt.addRow(o);}}
         catch(Exception e)
         {
             JOptionPane.showMessageDialog(this,e);}
         jTextField1.setText("");
         jTextField2.setText("");
         jTextField3.setText("");
         jTextField4.setText("");
         jTextField5.setText("");
         // TODO add your handling code here:
    }

    private void btn2MouseEntered(java.awt.event.MouseEvent evt) {
        btn2.setBackground(new java.awt.Color(255,255,0) );
    }

    private void btn2MouseExited(java.awt.event.MouseEvent evt) {
        btn2.setBackground(new java.awt.Color(255,255,255)); // TODO add your handling code here:
    }
   private void btn2ActionPerformed(java.awt.event.ActionEvent evt) {
try{ // msg d confirmation !
        if (JOptionPane.showConfirmDialog(null, "confirmer la modification","modification sur un Client"
         JOptionPane.YES_NO_OPTION) == JOptionPane.OK_OPTION)
           Class.forName("org.postgresql.Driver");
           Connection con = DriverManager.getConnection(url, user, password);
           String sql = " update   cliente  set    nom = ?,prenom = ?,teliphone = ? ,mail= ?  where id.
            ResultSet result = st.executeQuery(sql);
           // ps.setString(1, jTextField1.getText());
      st.setString(2, jTextField2.getText());
           st.setString(3, jTextField3.getText());
          st.setString(4, jTextField4.getText());
           st.setString(1, jTextField4.getText());
       }catch(Exception e){
          JOptionPane.showMessageDialog(this,"Erreur de modifiere ce client" +e.getMessage());}
       try{
           Connection connexion = DriverManager.getConnection(url, user, password);
          java.sql.Statement st = connexion.createStatement();
          String sql=
                  "SELECT id, nom, prenom, teliphone, mail FROM cliente";
          ResultSet result = st.executeQuery(sql);
          DefaultTableModel  dt = (DefaultTableModel )jTable2.getModel();
           dt.setRowCount(0);
           while(result.next())
           {
               Object o[]={result.getInt("id"),result.getString("nom"),result.getString("prenom"),
                   result.getString("teliphone"),result.getString("mail")};
               dt.addRow(o);}}
       catch(Exception e)
       {
```

```
            JOptionPane.showMessageDialog(this,e);}
        jTextField1.setText("");
        jTextField2.setText("");
          jTextField3.setText("");
         jTextField4.setText("");
        jTextField5.setText("");
    }

    private void comrechActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextField7ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextField7KeyReleased(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton5MouseEntered(java.awt.event.MouseEvent evt) {
        jButton5.setBackground(new java.awt.Color(51,255,102) );  // TODO add your handling code here:
    }

    private void jButton5MouseExited(java.awt.event.MouseEvent evt) {
        jButton5.setBackground(new java.awt.Color(255,255,255) ); // TODO add your handling code here:
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
try{
       try {
           Class.forName("org.postgresql.Driver");
       } catch (ClassNotFoundException ex) {
           Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
       }
            Connection connexion = DriverManager.getConnection(url, user, password);
             Statement st = connexion.createStatement();
           ResultSet rs = st.executeQuery (" select * from cliente  where  "+combo()
                +"  like '%"+jTextField7.getText()+"%'" );
          //  PreparedStatement pst=con.prepareStatement(sql);
          //  pst.setString(1,(jTextField6.getText()));
          // ResultSet res= pst.executeQuery();
           if(rs.next()){
          jTextField1.setText(rs.getString("id"));
                jTextField2.setText(rs.getString("nom"));
                jTextField3.setText(rs.getString("prenom"));
                 jTextField4.setText(rs.getString("teliphone"));
                jTextField5.setText(rs.getString("mail"));
              Object o[]={rs.getInt("id"),rs.getString("nom"),rs.getString("prenom"),
                rs.getString("teliphone"),rs.getString("mail")};
                 dt.addRow(o);}
           else {
             JOptionPane.showMessageDialog(null," Aucune client");   } }
           catch (SQLException ex) {
          Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);}
       if (dt.getRowCount() == 0){
            JOptionPane.showMessageDialog(null, "Aucun client");
               try{
            Connection connexion = DriverManager.getConnection(url, user, password);
           java.sql.Statement st = connexion.createStatement();
           String sql=
                   "SELECT id, nom, prenom, teliphone, mail FROM cliente";
           ResultSet result = st.executeQuery(sql);
           DefaultTableModel  dt = (DefaultTableModel )jTable2.getModel();
            dt.setRowCount(0);
            while(result.next())
            {
                Object o[]={result.getInt("id"),result.getString("nom"),result.getString("prenom"),
                    result.getString("teliphone"),result.getString("mail")};
                dt.addRow(o);}}
           catch(Exception e)
           {
            JOptionPane.showMessageDialog(this,e);}}

           jTextField5.setText("");
         // TODO add your handling code here:
    }
    String combo(){
switch(comrech.getSelectedIndex()){
    case 0: return "id";
    case 1: return "nom";
    case 2: return "prenom";
      case 3: return "teliphone";}
      return "";
        // TODO add your handling code here:
    }

    private void btn5MouseEntered(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
    }
```

```
    }

    private void btn5MouseExited(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
    }

    private void btn5ActionPerformed(java.awt.event.ActionEvent evt) {
          MessageFormat      header  = new  MessageFormat("liste des Clients:");
         MessageFormat      footer  = new  MessageFormat("page{0,number,integer}");
          try{
             jTable2.print(JTable.PrintMode.NORMAL,header,footer);}
          catch(java.awt.print.PrinterException e){
             System.err.format("Erreur d'inmprision",e.getMessage());}
// TODO add your handling code here:
    }

    private void btn6MouseEntered(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
    }

    private void btn6MouseExited(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
    }

    private void btn6ActionPerformed(java.awt.event.ActionEvent evt) {
       NewJFrame etu2 = new NewJFrame();
        etu2.setVisible(true);
        this.setVisible(false);
        etu2.pack();
      // TODO add your handling code here:
    }

    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        // TODO add your handling code here:
    }

    private void jTable2MouseClicked(java.awt.event.MouseEvent evt) {
        int number = jTable2.getSelectedRow(); // TODO add your handling code here:
        jTextField1.setText(jTable2.getValueAt(number,0).toString());
        jTextField2.setText(jTable2.getValueAt(number,1).toString());
        jTextField3.setText(jTable2.getValueAt(number,2).toString());
        jTextField4.setText(jTable2.getValueAt(number,3).toString());
        jTextField5.setText(jTable2.getValueAt(number,4).toString());
    }
public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new client().setVisible(true); }});
    }

    // Variables declaration - do not modify
    private javax.swing.JButton btn1;
    private javax.swing.JButton btn2;
    private javax.swing.JButton btn3;
    private javax.swing.JButton btn4;
    private javax.swing.JButton btn5;
    private javax.swing.JButton btn6;
    private javax.swing.JComboBox<String> comrech;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton5;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel9;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTable jTable2;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTextField jTextField5;
    private javax.swing.JTextField jTextField7;
    // End of variables declaration
}
```