REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABOU-BEKR BELKAID – TLEMCEN

Laboratoire de Recherche en Informatique de Tlemcen

THÈSE LMD

Présentée à :

FACULTE DES SCIENCES – DEPARTEMENT DE INFORMATIQUE

Pour l'obtention du diplôme de :

DOCTORAT

Spécialité : Informatique

Par :

Mr GUEDDOUDJ EL YAZID

Sur le thème

Spark au service d'ETL pour la gestion des données RDF streaming

---

Soutenue publiquement le 04/11/2023 à Tlemcen devant le jury composé de :

Pr. Benamar Abdelkrim	Professeur	Université de Tlemcen	Président
Dr. Hadjila Fethallah	MCA	Université de Tlemcen	Examinateur
Mr Benomar Amine	MCA	Université de Tmouchent	Examinateur
Pr. Souier Mehdi	Professeur	Ecole supérieure de Management de Tlemcen	Examinateur
Dr. Maliki Fouad		Ecole supérieure sciences appliquées de Tlemcen	Examinateur
Pr. Chikh Azeddine	Professeur	Université de Tlemcen	Directeur de thèse

Laboratoire de Recherche en Informatique LRIT
BP 119, 13000 Tlemcen - Algérie

A ***mon défunt père Gueddoudj Mohamed*** *. . .*

Remerciements

Je tiens à remercier le **professeur. Azeddine Chikh**, directeur du laboratoire de recherche en informatique (LRIT), mon directeur de thèse, pour sa patience, ses conseils et ses précieuses contributions tout au long de la réalisation de cette thèse. La langue ne suffit pas pour exprimer mon appréciation pour son soutien et ses encouragements. Je voudrais exprimer ma gratitude à tous les membres du laboratoire LRIT de l'université de tlemcen et à tous mes enseignants du département informatique durant le cursus master à l'université de Tlemcen.

Un remerciement très chaleureux à Mr. Benamar Abdelkrim, professeur au département d'informatique de l'université de Tlemcen, qui m'a fait l'honneur de présider le jury de cette thèse.

Enfin et surtout, les mots ne peuvent décrire pleinement ma gratitude sincère et la plus profonde envers ma famille - **mon défunt père Gueddoudj Mohamed** qui m'a conduit dans cette voie et m'a toujours encouragé à poursuivre de plus en plus haut, ma mère, mon frère madani, mes soeurs et frères et ma femme, pour leur amour, leur soutien continu et leurs encouragements sans fin, à eux je dédie cet effort.

Tlemcen, le 5 novembre 2023.

مـلـخـص .

يؤدي انفجار البيانات إلى زيادة تكامل تنسيقات البيانات الجديدة مثل قواعد بيانات الرسوم البيانية وبيانات RDF داخل الشركات التي تعيش في عالم شديد التنافسية. بالإضافة إلى ذلك، أظهرت عمليات الاستخراج والتحويل والتحميل (ETL) بعض النضج بالنسبة لمصادر البيانات التقليدية، ولكنها لا تتوسع عند استخدامها للتعامل مع مصادر البيانات المتنوعة والكبيرة جدًا التي تتضمن بيانات RDF . وهي تحتوي على قدر كبير من المعارف والمعلومات القيمة التي ينبغي استغلالها من قبل الشركات المجهزة بتكنولوجيا مستودعات البيانات لزيادة قيمتها في عالم شديد التنافسية. كما يمثل ظهور منصات جديدة مثل polystore فرصة للنشر على أحدث الأجهزة. تتطلب عمليات ETL مرحلتين مهمتين بما في ذلك التقسيم و تخصيص البيانات. بالإضافة إلى ذلك، يتم تحفيز الباحثين لتطوير واختراع عمليات ETL لدعم التحليلات في الوقت الحقيقي. في هذه الأطروحة : أولاً، نقترح النمذجة المفاهيمية لعمليات ETL باستخدام ال BPMN .يتم تحويل هذه العمليات تلقائيًا إلى نصوص برمجية ليتم تنفيذها في إطار عمل Spark . يتم تجميع الحل في بنية ETL موزعة جديدة تدعم المعالجة المجمعة والتدفقية. ولجعل نهجنا الجديد أكثر واقعية وقابلية للتقييم ، يتم النظر في دراسة حالة حقيقية باستخدام معيار LUBM ،والذي يتضمن مصادر بيانات غير متجانسة.

ثانيًا، نقترح بنية جديدة لعمليات ETL تسمى Open-Scala-ETL (Os-ETL) ، ومجهزة بطريقة نشر مستودع البيانات القائمة على المتاجر المتعددة، والتي تتيح التحليل في الوقت الفعلي. يهدف هذا الحل إلى نزع مشكلة نشر مستودع بيانات هيكل الرسم البياني على مخزن متعدد وهي مهمة صعبة، مع مرحلتين هما تقسيم البيانات وتخصيصها. بالإضافة إلى ذلك، يعد Os-ETL حلاً موزعًا يدعم المعالجة المجمعة والتدفقية باستخدام إطار عمل .Spark

يتم تنفيذ نصوص Scala في الأخير لتقسيم الرسوم البيانية لـ RDF وتوزيع الأجزاء المختلفة التي تم الحصول عليها من المصادر المختلفة. يعتمد تطبيق Os-ETL على Apache Spark مع نشر ETL على مخزن Spark

SQL المتعدد. كما يسمح حل Os-ETL للشركات التي لديها تقنية مستودع البيانات بالحصول على الأداء وقابلية التوسع وزمن الوصول بين مستودع البيانات ومصادر البيانات الخاصة به. لقد تم تقييم نهجنا والتحقق من صحته باستخدام بيانات غير متجانسة واسعة النطاق، بما في ذلك معيار LUBM وملفات CSV وقاعدة بيانات Oracle وقاعدة بيانات الرسم البياني Neo4j . تؤكد النتائج التي تم الحصول عليها أداءها المتفوق من حيث قابلية التوسع والتحسين.

Résumé

L'explosion des données suscite de plus en plus l'intégration de nouveaux formats de données tels que les bases de données graphes et les données RDF (Resource Description Framework) au sein des entreprises qui vivent dans un monde hautement concurrentiel. En outre, les processus d'extraction, de transformation et de chargement (ETL) ont démontré une certaine maturité pour les sources de données traditionnelles, mais ils ne s'adaptent pas lorsqu'ils sont utilisés pour gérer des sources de données volumineuses et très variées qui impliquent des données RDF. Ces dernières contiennent une grande quantité de connaissances qui devraient être exploitées par les entreprises, équipées de la technologie d'entrepôt de données, pour augmenter leur valeur dans un monde hautement concurrentiel. L'émergence de nouvelles plateformes telles que le polystore représente également une opportunité pour le déploiement de matériel de pointe. Les processus ETL nécessitent deux phases importantes, notamment le partitionnement et l'allocation des données. De plus, les chercheurs sont motivés à développer et à inventer des processus ETL pour prendre en charge l'analyse en temps réel.

Dans ce manuscrit : Dans un premier temps, nous proposons une modélisation conceptuelle des processus ETL à l'aide de de la notation BPMN (Business Process Modeling Notation). Ces processus sont automatiquement convertis en scripts à implémenter dans le framework Spark. La solution est conditionnée selon une nouvelle architecture ETL distribuée qui prend en charge à la fois le traitement par lots et par flux (stream). Pour rendre notre nouvelle approche plus concrète et évaluable, une étude de cas réel utilisant le benchmark LUBM (The Lehigh University Benchmark), qui implique des sources de données hétérogènes, est envisagée. Deuxièmement, nous proposons une nouvelle architecture pour les processus ETL nommée Open-Scala-ETL (Os-ETL), équipée d'une méthode de déploiement d'entrepôt de données basée sur un polystore, qui permet une analyse en temps réel. La solution Os-ETL vise à résoudre le problème de déploiement d'un entrepôt de données à structure graphe sur un polystore qui est une tâche difficile, avec deux phases à savoir le partitionnement et l'allocation des données. De plus, Os-ETL est une solution distribuée prenant en charge le traitement par lots et en continu (stream) à l'aide du framework Spark. Des scripts Scala sont exécutés dans ce dernier pour partitionner les graphes RDF et répartir les différents fragments obtenus, issus des différents sites. L'implémentation de l'Os-ETL est basée sur Apache Spark avec un déploiement ETL sur le polystore Spark SQL. La solution Os-ETL permet aux entreprises disposant d'une technologie d'entrepôt de données de gagner en performance, en évolutivité et en latence entre un entrepôt de données et ses sources de données. Notre ap-

proche a été évaluée et validée à l'aide de données hétérogènes à grande échelle, notamment le benchmark LUBM, les fichiers CSV, la base de données Oracle et la base de données graphes Neo4j. Les résultats obtenus confirment ses performances supérieures en termes d'évolutivité et d'optimisation.

Mots-clés : ETL, Spark, Big Data, RDF, Partitioning, Data Warehouse, Polystore, Scalability, Design, BPMN.

Abstract

The explosion of data is increasingly prompting the integration of new data formats such as graph databases and RDF (Resource Description Framework) within companies that live in a highly competitive world. Besides the Extract, transform and load (ETL) processes have demonstrated some maturity for traditional data sources, but they do not scale when used to handle large and highly varied data sources which involve RDF data. These later contain a large amount of knowledge that should be exploited by companies, equipped with data warehouse technology, to increase their value in a highly competitive world. The emergence of new platforms such as Polystore represents also an opportunity for the deployment of advanced hardware. ETL processes require two important phases, including partitioning and data allocation. Moreover, researchers are motivated to develop and invent ETL processes to support real-time analytics. In this manuscript: First we, propose a conceptual modeling of these processes using BPMN (Business Process Modeling Notation). These processes are automatically converted to scripts to be implemented within Spark framework. The solution is packaged according a new distributed architecture (Open ETL) that supports both batch and stream processing. To make our new approach more concrete and evaluable, a real case study using the LUBM benchmark, which involves heterogeneous data sources is considered.

Secondly, we propose a novel architecture for ETL processes named Open-Scala-ETL (Os-ETL), equipped with a method of deployment of data warehouse based on a polystore, that permits a real-time analysis. Os-ETL solution aims to solve the problem of deploying a graph structure data warehouse on a polystore which is a difficult task, with two phases namely the partitioning and the allocation of the data. Moreover, Os-ETL is a distributed solution supporting batch and streaming processing using Spark framework. Scala scripts are executed in the later to partition RDF graphs and distribute the different fragments, resulting from the different sites. Os-ETL implementation is based on Apache Spark with an ETL deployment on Spark SQL polystore. Os-ETL solution allows companies with data warehouse technology gain in performance, scalability and latency between a data warehouse and its data sources. Our approach has been evaluated and validated using heterogeneous large-scale data including LUBM benchmark, CSV files, Oracle database, and Neo4j graph database. The achieved results confirm its higher performance in terms of scalability and optimization.

Keywords : ETL, Spark, Big Data, RDF, Partitioning, Data Warehouse, Polystore, Scalability, Design, BPMN.

Liste des contributions scienfifiques

— El Yazid Gueddoudj et Azeddine Chikh (2023), Towards, A Scalable and Efficient ETL. International Journal of Computing and Digital Systems, DOI : https ://dx.doi.org/10.12785/ijcds/140195

— El Yazid Gueddoudj, Azeddine Chikh et Abdelouahab Attia (2023), Os-ETL : A High-Efficiency, Open-Scala Solution for Integrating Heterogeneous Data in Large-Scale Data Warehousing. Ingénierie des Systèmes d'Information, DOI : https ://doi.org/10.18280/isi.280303.

— El Yazid Gueddoudj, Azeddine Chikh et Abdelouahab Attia (2023), AGRETL : A Scalable ETL Process for Decision Support in Agriculture in the Era of Big Data Using Spark, The First International Hybrid Conference On : Artificial Intelligence Applied To The Agriculture Sector, At : University Mohamed El Bachir El Ibrahimi of Bordj Bou ArreridjBordj - ALGERIA.

Table des matières

Liste des figures

Liste des tableaux

Introduction Générale

1

Sommaire

1.1 Contexte et motivations

Aujourd'hui, toutes les entreprises disposent de données, qui peuvent être sous n'importe quelle forme : fichiers csv, fichiers plats, XML, bases de données relationnelles ou graphes, etc. Ces données sont capturées, agrégées, nettoyées et chargées dans un entrepôt de données dimensionnel qui est dénormalisé. Tous les processus liés à la préparation des données pour une analyse future sont appelés ETL (Extract-Transform-Load), qui est largement utilisé dans les projets de la business intelligence.

Étant donné que le volume de données augmente rapidement dans les systèmes d'entreprise, les processus ETL prennent plus de temps. De plus, l'apparition de nouvelles plateformes de stockage comme le polystore, a conduit à un écosystème hétérogène devenu difficile à gérer et à optimiser. En conséquence, de nouveaux systèmes de traitement de données à grande échelle (par exemple, Spark [1] ou MapReduce (MR) [2]) ont vu le jour.

L'une des difficultés de la construction d'une application d'entrepôt de données est la gestion de l'hétérogénéité des sources d'informations et des plates-formes de stockage, en raison de la complexité de la conception des processus ETL. Par conséquent, ce travail de recherche vise à faciliter et optimiser la conception et la mise en œuvre des processus ETL dans un écosystème hétérogène.

Une quantité considérable de littérature a été publiée concernant le domaine de recherche des processus ETL. Les auteurs de [3], proposent une modélisation logique des processus ETL utilisant l'algèbre relationnelle étendue avec des opérations de mise à jour. Les tâches ETL sont exprimées en XML. De plus, le travail [4] sépare quatre types de modèles : ontologie, UML, graphe et BPMN. Diverses approches ont été discutées pour optimiser les processus ETL. Dans [5], les auteurs proposent un modèle conceptuel de diagramme de mappage d'entités (EMD) afin d'implémenter les processus ETL. Les auteurs dans [6], ont développé un framework basé sur ETL avec déploiement sur la plate-forme cloud. Dans [7], l'auteur propose une architecture basée sur le cloud pour un ETL Big Data. Ils utilisent un algorithme général d'optimisation de la taille des clusters. Dans [8], les auteurs proposent un modèle ETL dans lequel l'application de la méthode CDC (Change Data Capture) dans le framework Spark peut réduire la quantité de données dans le processus ETL. Une architecture ETL pour le traitement des flux IoT avec trois composants (le collecteur de données, le moteur de flux ETL et le composant OLAP) a été proposée par [9]. Lorsqu'il s'agit de stocker et d'analyser des données hétérogènes, les entrepôts de données traditionnels échouent car ils reposent sur RDBMS [10]. Peu de travaux ont abordé l'association de l'ETL avec le polystore. Dans [9], les auteurs ont implémenté la

nouvelle architecture Streaming ETL pour une utilisation dans un contexte relationnel, ils ont envisagé un déploiement ETL sur le polystore BigDAWG d'Intel. Une solution de déploiement ETL as a Service (ETLaaS) et Polystore as a Service (PaaS) a été proposée par [11]. Nous soutenons que pour les entrepôts de données de génération moderne, dans lesquels il faut prendre en compte les problèmes d'hétérogénéité des plateformes de stockage, ainsi que la variété et le volume des sources de données, liés au VS du big data, une nouvelle architecture ETL devrait être conçu. Au début de notre thèse nous avons consacré beaucoup de temps à apprendre le fonctionnement, et la programmation parallèle du framework Spark, nous avons acquis une vaste expérience de recherche et pratique avec des algorithmes parallèles de données à grande échelle, notamment le développement de nouveaux algorithmes pour les entrepôts de données dans le contexte du big data.

Dans nos contributions, nous exploitant l'utilisation de système polystore pour fournir des solutions de gestion de données qui ont différents modèles de gestion de données à différents ensembles de données qui ont différents modèles de données et de programmation sous-jacents. Les polystores prennent en charge plusieurs langages de requête et différents SGBD, ils sont conçus pour unifier les requêtes sur plusieurs modèles de données. Ainsi, avec les systèmes polystore, nous pouvons accéder à plusieurs moteurs de stockage via une seule interface.

1.2 Objectifs.

La principale motivation de notre travail est de contribuer à l'amélioration des processus ETL : le besoin d'améliorer l'efficacité et les performances de ces processus qui traitent des données hétérogènes et volumineuses par rapport aux autres techniques d'optimisation existantes. Les auteurs du travail [6], présentent une nouvelle approche basée sur le contrôle de flux dans ETL. Dans [7], les auteurs introduisent une approche basée sur le flux de contrôle pour la modélisation des processus ETL et utilisent la combinaison de la parallélisation et de la mémoire cache partagée pour optimiser les performances des processus ETL de l'entrepôt de données.

Ainsi, dans nos solutions proposées consistent à impliquer diverses sources de données (relationnelles, graphiques et CSV, etc.) et tirer partie de la puissance de calcul et du stockage distribué, offerts par le framework Spark pour optimiser les processus ETL. Les acteurs de l'université et de l'industrie ont adopté Apache Spark comme un framework rapide et évolutif [8]. Donc dans nos contributions, un nouveau framework est concu pour permettre

le développement d'entrepôts de données multidimensionnels capables de gérer de grandes masses de données. Nous assurons le déploiement sur un système polystore.

Par conséquent, notre travail profite des différents systèmes de stockage polystore pour augmenter les performances d'exécution des algorithmes d'analyse. L'objectif principal de notre travail de thèse vise à créer une nouvelle architecture pour un processus ETL basé sur Spark, robuste, flexible et rentable afin d'optimiser et de rendre les processus ETL plus efficaces en les mettant à l'échelle.

1.3 Contributions

Dans le cadre de cette thèse, nous traitons les deux pricipales difficultés rencontrées lors la mise en oeuvre d'un processus ETL efficace et robuste : La première difficulté est la diversité syntaxique et sémantique des sources de données, qui découle de la variété des structures et des formats utilisés pour stocker les informations, l'hétérogénéité sémantique découle également des multiples interprétations des éléments du monde réel. La deuxième difficulté est l'émergence des nouveaux systèmes de stockages de données, tels que les systèmes polystores. Alors, le souci principal de notre travail est d'améliorer la performance et l'efficacité des processus ETL appliqués aux entrepôts de données afin d'augmenter la productivité pour les systèmes d'informations d'aide à la décision dans l'écosystème big data en adaptant les nouveaux systèmes de traitement de données massives et les nouvelles méthodes de modélisation de systèmes logiciels. Ainsi, nos contributions de cette thèse de doctorat peuvent être subdivisées en trois contributions décrites comme suit :

1.3.1 Contribution 1

Il s'agit de proposer une nouvelle méthode qui comprend à la fois des méthodes de modélisation et d'optimisation du processus ETL [12], en respectant l'importance de la question et les défis dans ce domaine. Nous avons présenté une nouvelle approche pour optimiser les processus ETL grâce à une nouvelle architecture distribuée (Open ETL) qui prend en charge à la fois le traitement par lots et par flux (stream). Nous avons présenté une nouvelle approche pour concevoir des processus ETL en utilisant un ensemble de notations en BPMN pour les opérateurs ETL capables de modéliser des activités ETL ainsi que des transformations de modèles en scripts Scala qui sont automatiquement convertis en scripts à implémenter dans le framework Spark. Une solution de déploiement polystore hybride basée sur Spark SQL est donnée. Nous avons validé les étapes concrètes de l'utilisation de l'approche Open ETL propo-

sée pour charger les données dans un schéma d'entrepôt de données. Les résultats importants sont liés au temps de performance du système proposé par rapport aux autres alternatives ETL. De plus, nous avons également présenté l'évolutivité de la méthode proposée sur de grands ensembles de données.

1.3.2 Contribution 2

Il s'agit de proposer proposé une nouvelle solution appelée Os-ETL qui consiste en six étapes : (i) extraire et transférer des données de flux ; (ii) aligner toutes les sources de données sur le modèle de données RDF ; (iii) partitionner les sources de données obtenues (iv) transformer les données RDF en données GraphX ; (v) transformations appropriées sur les données GraphX ; et (vi) allocation et distribution des fragments RDF obtenus sur le polystore Spark SQL. Tout d'abord, nous avons décrit les composants utilisés par Os-ETL qui extrait des sources de données hétérogènes impliquant des données Web en tant que données externes. Deux techniques, la méthode CDC (Change Data Capture) et la stratégie de partitionnement ont été utilisées et qui ont grandement amélioré le temps de réponse d'ETL. En outre, nous avons proposé l'utilisation de systèmes polystore comme solution matérielle pour déployer l'entrepôt de données cible et les processus ETL. Le système proposé Os-ETL améliore considérablement les processus ETL, il permet aux développeurs de se concentrer sur la logique métier, plutôt que de se soucier du processus complexe d'extraction, de transformation et de chargement des données dans un environnement très varié. Les résultats les plus importants obtenus, sont liés au temps de performance des processus ETL par rapport aux travaux précédents, grâce aux stratégies de partitionnement, en mémoire et en pipeline. Les résultats obtenus confirment les meilleures performances du système proposé Os-ETL en termes de scalabilité (évolutivité) et optimisation.

1.3.3 Contribution 3

Il s'agit de proposer un grand entrepôt de données sur la distribution du stockage et de la vente au détail des agriculteurs sur le marché. En effet la croissance démographique, qui devrait atteindre environ 57 625 000 habitants d'ici 2040 d'aprés l'office National des Statistiques (ONS), l'augmentation du pouvoir d'achat et l'évolution des habitudes alimentaires lancent un défi différent au système de production alimentaire algérien. Le problème est maintenant de relever le défi d'augmenter la production alimentaire de manière durable avec une utilisation judicieuse des intrants et un minimum de dommages à l'environnement. Cependant, le

pays a besoin d'outils et de techniques de big data pour résoudre les problèmes complexes de l'agriculture et de la fabrication et des services agro-industriels. Dans ce contexte s'articule l'objectif de ce travail. La conception d'un grand entrepôt de données sur la distribution du stockage et de la vente au détail de l'agriculteur au marché. Le grand entrepôt de données proposé peut être étendu au système d'aide à la décision (DSS) combiné avec des techniques d'exploration de données.

1.4 Organisation de la thèse

Cette thèse est divisée en quatre chapitres. Le **chapitre 1** fournit le contexte de la recherche, y compris un aperçu des processus d'extraction, transformation et de chargement des données, l'objectif de la recherche sur la motivation, les questions de recherche, l'approche méthodologique et les grandes lignes.

Le chapitre 2 présente les fondements théoriques et les concepts fondamentaux sur lesquels repose notre thèse. Ces fondements présentent les principales notions relatives au domaine de recherche abordé dans la thèse.

Le chapitre 3, aborde la revue de la littérature sur les processus ETL, les entrepôts de données et les outils d'intégration introduit et utilisé dans cette thèse.

Le dernier **chapitre 4**, est consacré à nos contributions dans le domaine de l'ETL avec des études de cas. Pour chaque proposition nous décrivons la méthode proposée, où nous présentons une formalisation des entrées et des sorties de la méthode avec une description de l'ensemble des étapes à suivre, ainsi que les expérimentations effectuées pour la validation de notre de chaque solution proposée et les résultats obtenus.

Enfin, cette thèse se termine par une conclusion générale sur les travaux de recherche présentés et les perspectives de travail de recherche.

2 Généralités et concepts fondamentaux sur les processus ETL

Sommaire

2.1 Introduction

Ce chapitre présente les concepts fondamentaux et explique les terminologies techniques et les technologies fondamentales liées au processus ETL, y compris l'entrepôt de données, le système de source de données, les données structurées et non structurées, les techniques de traitement des données traditionnels et non traditionnels qui se sont appliquées dans cette thèse et comment elles sont combinées pour les mettre en œuvre.

2.2 Les Systèmes d'Information Décisionnels (SID)

2.2.1 Définition

Un système dinformation décisionnel consiste en un ensemble de méthodologies, de processus, d'architectures et de technologies qui transforment les données brutes en informations utiles à la prise de décision. Il apporte une aide à la gestion des différents niveaux organisationnels de la hiérarchie pour l'analyse des informations stratégiques.

L'objectif principal est de simplifier des données complexes sous une forme abrégée. Le processus de conversion de la grande quantité de données en une forme organisée et simplifiée comprend un ensemble d'activités, à savoir l'extraction des données à partir d'une source spécifiée, la transformation des données, l'intégration des données, le filtrage ou le nettoyage et le stockage des données dans un référentiel particulier. Ce référentiel est appelé Entrepôt de données [13].

Les systèmes de Business Intelligence combinent des données opérationnelles avec des outils analytiques pour présenter des informations complexes et compétitives aux planificateurs et aux décideurs [14]. Les processus décisionnels ont un impact énorme sur le succès d'une organisation. Les décideurs, à savoir les cadres supérieurs, utilisaient généralement les connaissances acquises avec l'expérience pour décider. Aujourd'hui, la Business Intelligence (BI) est apparue comme un sujet important des Systèmes d'Information.

2.2.2 Les avantages des SID

1. Le SID est crucial pour la gestion efficace et l'emploi des données. Il est apparu dans la dernière partie du 20ième siècle et est devenue une partie intégrante du processus de prise pour les entreprises prudentes qui cherchent à faire usage d'une multitude de

données relatives au service à la clientèle, les stocks, les prix, et bien plus encore la décision.

2. L'intégration des données est une caractéristique essentielle de toute solution de BI et peut tenir compte des facteurs qui affectent effectivement les préoccupations spécifiques du marché, tels que les données démographiques des clients, des conditions économiques et des environnements de marché. Ces analyses prédictives permettent aux entreprises d'affiner les processus d'affaires en préparation pour l'avenir. On peut soutenir que l'adaptation individuelle des processus d'affaires est une distinction essentielle entre les entreprises dans le même secteur. Ils utilisent souvent des technologies et des produits similaires, mais les personnaliser façons très différentes.
3. Le SID facilite la structure d'entreprise appropriée en fournissant un assortiment de données qui détaille ses besoins. Il permet d'identifier les zones qui ont un manque ou un surplus d'attention, d'accélérer la rationalisation des ressources de l'entreprise qui peuvent aider à la productivité. Solutions de BI compétitifs permettent la mise à jour immédiate, accordant aux utilisateurs les informations les plus récentes sur les processus de l'entreprise et leurs effets.

2.3 Les Entrepôts de données

2.3.1 Définition

Un entrepôt de données est une base de données relationnelle ou multidimensionnelle conçue pour les requêtes et les analyses. Il consolide généralement les données historiques et transactionnelles issues de plusieurs ressources. Bill Inmon dans [15] : "Un entrepôt de données est un processus de prise de décision de gestion de données orienté sujet, intégré, variable dans le temps et non volatile", figure 2.1. Ralph Kimball dans [16] a fourni une autre définition d'un entrepôt de données, figure 2.2 : "Un entrepôt de données est une copie d'une structure de données de transaction pour la requête et l'analyse". Bill Inmon et Ralph Kimball sont des pionniers des entrepôts de données, ils avaient des approches différentes de leur conception, le premier considérait l'entrepôt comme le référentiel centralisé de toutes les données de l'entreprise, ralph considérait l'entrepôt comme une combinaison de différents magasins de données. Ainsi, un entrepôt de données est un système utilisé pour intégrer, stocker et traiter des données provenant de sources de données souvent hétérogènes afin de fournir aux décideurs une perspective multidimensionnelle.

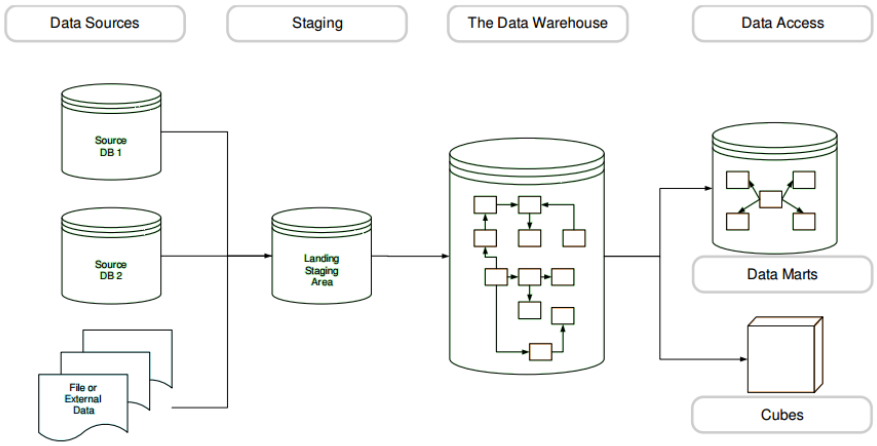

FIGURE 2.1 – *Inmonn model.*

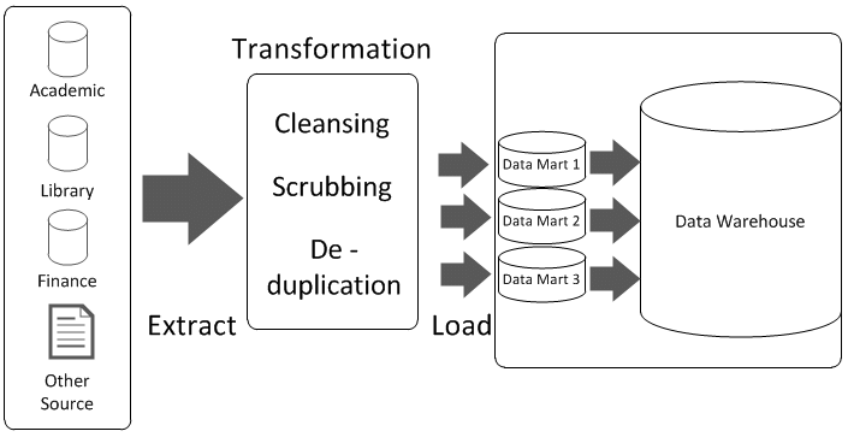

FIGURE 2.2 – *Ralph model.*

Avec l'émergence du Big Data, plusieurs changements se sont produits au niveau de l'entrepôt de données. Le Big Data a permis à de plus en plus d'organisations de se concentrer sur le stockage de la grande quantité de données disponibles pour obtenir un avantage commercial. Dans le volet technique, il est garanti que les données de l'entreprise sont assemblées à partir de ses systèmes sources et qu'elles sont stockées, organisées et nettoyées malgré leur origine. Le composant métier garantit que les données structurées permettent de créer les rapports et les chiffres clés souhaités.

Les principales caractéristiques d'un Entrepôt de Données sont :

1. Intégré : les données stockées dans l'entrepôt de données doivent être intégrées dans une structure cohérente, de sorte que les incohérences entre les différents systèmes opérationnels doivent être éliminées. Les informations sont également généralement structurées en différents niveaux de détail pour répondre aux différents besoins des utilisateurs.
2. Thématique : seules les données nécessaires au processus de génération de connaissance métier sont intégrées depuis l'environnement opérationnel. Les données sont organisées

par sujet pour créer un accès facile et compréhensible pour les utilisateurs. Par exemple, toutes les données des clients peuvent être consolidées dans une seule table de l'entrepôt de données. Ainsi, les demandes d'informations des clients seront plus faciles à répondre car toutes les informations résident au même endroit.

3. Historique : le temps est une partie implicite des informations contenues dans un entrepôt de données. Les systèmes opérationnels affichent toujours l'état réel de l'activité commerciale. Au contraire, les informations stockées dans l'entrepôt de données sont utilisées, par exemple, pour effectuer des analyses de tendance. Par conséquent, l'entrepôt de données est chargé avec différentes valeurs d'une variable dans le temps pour permettre des comparaisons.
4. Non volatile : le magasin d'informations d'un entrepôt de données existe pour être lue mais pas modifiée. L'information est permanente, ce qui signifie que la mise à jour de l'entrepôt de données intègre les dernières valeurs des variables qu'il contient sans aucune action sur ce qui existait déjà.

La gestion, l'exploitation et l'optimisation d'un jeu de données d'un volume important issu de sources hétérogènes à des fins d'analyse est un problème qui est apparu au sein des entreprises il y a de nombreuses années. Leur réponse à ce type de problème s'est traduite par la mise en place d'un ensemble de nouvelles méthodes et technologies regroupées sous le terme d'informatique décisionnelle (business intelligence en anglais). Plus récemment, ces technologies ont été adaptées pour répondre à des problématiques biologiques, dans les domaines de la foresterie (Miquel et al., 2002a), de la gestion de la pollution de l'eau (Bimonte, 2007) ou encore de la biologie moléculaire (Shah et al., 2005).

Le marché de l'entrepôt de données en tant que service était évalué à 1,44 milliard USD en 2020 et devrait atteindre 4,3 milliards USD d'ici 2026, avec un TCAC de 20 % sur la période de prévision 2021-2026. L'intérêt croissant des entreprises à comprendre les ressources disponibles les informations concernant les processus commerciaux, les produits, les clients et les services pour saisir de nouvelles opportunités commerciales ont un impact positif sur le marché.

En 2020, la valeur du marché du stockage de données en tant que service s'élevait à 1,44 milliard de dollars. On estime qu'il atteindra 4,3 milliards de dollars d'ici 2026, avec un taux de croissance annuel composé (TCAC) de 20 % sur la période de prévision de 2021 à 2026. L'intérêt croissant des entreprises pour comprendre les ressources disponibles, telles que les informations sur les processus commerciaux, les produits, les clients et les services, afin de saisir de nouvelles opportunités commerciales, a un impact positif sur le marché [17].

2.3.2 Avantages des Entrepôts de données

— Le premier avantage réside dans les structures dans lesquelles les informations sont stockées (modèles de tables en étoile, flocon de neige, cubes relationnels...). Ce type de persistance de l'information est cohérent et fiable et permet une consultation et un traitement hiérarchisé (toujours dans un environnement différent des systèmes opérationnels).

— Un entrepôt de données nous donne un outil de prise de décision dans n'importe quel domaine fonctionnel, basé sur des informations commerciales intégrées et globales. Il facilite également l'application de techniques statistiques et d'analyses de modélisation pour trouver des relations cachées entre différentes données. Il offre également la possibilité d'apprendre des données passées et de prédire les situations futures dans divers scénarios.

2.3.3 Datamart

Un Datamart est une base de données départementale spécialisée sur le stock de données d'un domaine d'activité déterminé. Il se caractérise par une structure de données optimale pour analyser les informations en détail sous tous les angles qui affectent les processus du département. Un datamart peut être alimenté à partir des données d'un entrepôt de données, ou intégrer lui-même une compilation de différentes sources d'informations.

2.3.4 Lac de données (Data Lake)

Les auteurs de [18], définissent un lac de données (figure 2.3) comme une collection massive d'ensembles de données qui : (1) peuvent être hébergés dans différents systèmes de stockage ; (2) peuvent varier dans leurs formats ; (3) peuvent ne pas être accompagnés de métadonnées utiles ou peuvent utiliser différents formats pour décrire leurs métadonnées ; et (4) peuvent changer de manière autonome au fil du temps.

Contrairement à un entrepôt de données hiérarchique avec stockage de données de fichiers ou de dossiers, le lac de données utilise une architecture plate, où chaque élément de données a un identifiant unique et un ensemble de balises de métadonnées étendues. Le lac de données ne nécessite pas un schéma rigide ou une manipulation des données de toutes formes et tailles, mais il nécessite de maintenir l'ordre d'arrivée des données.

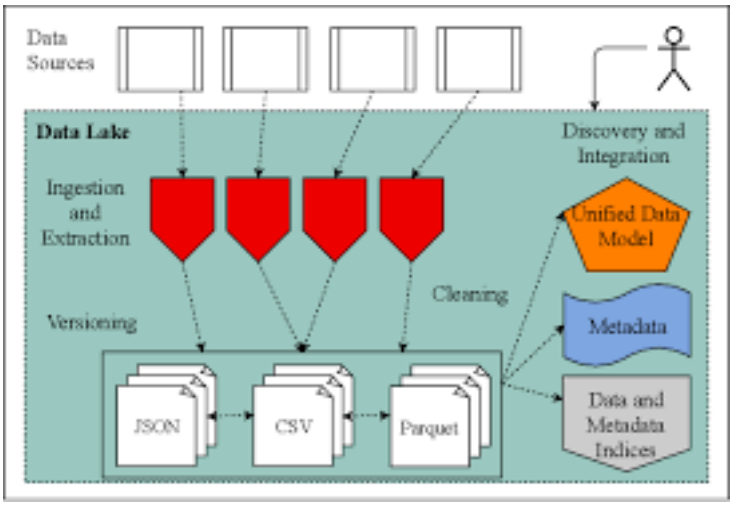

FIGURE 2.3 – *Système de gestion des lacs de données.*

2.4 Les Processus d'Extraction, de Transformation et de Chargement (ETL)

2.4.1 Definition

Dans une architecture d'entrepôt de données, les processus ETL sont responsables des tâches qui se déroulent en arrière-plan [19]. Après l'apparition des entrepôts de données, l'ETL était là, pourtant encore caché entre les lignes. Ce n'est que dans les années 2000 que l'ETL accède à une existence à part en devenant une phase importante d'intégration des sources, compte tenu de sa consommation en termes d'argent, de temps et de programmeurs. Le processus ETL extrait les données des sources, les transforme en un modèle commun et les charge dans l'entrepôt de données cible. La figure 2.4 montre les différentes phases impliquées dans le processus ETL, à savoir Extract, Transform et Load :

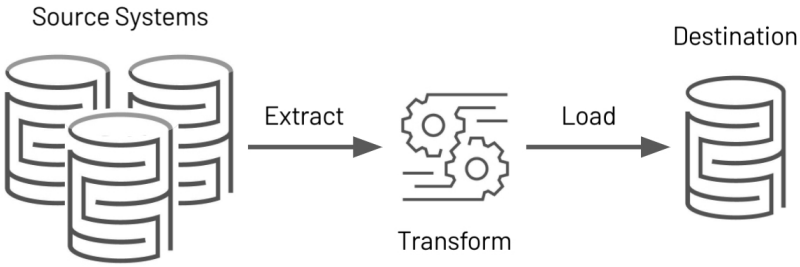

FIGURE 2.4 – *Les Processus ETL.*

Extract

La phase d'extraction des données s'effectue en deux phases. L'extraction complète est effectuée lorsque toutes les données sont extraites pour la première fois. L'extraction incré-

mentielle se produit lorsque des données nouvelles ou modifiées sont extraites des sources. L'extraction incrémentielle utilise des stratégies telles que des techniques basées sur les journaux, les déclencheurs ou les timestamp-based techniques (horodatages) pour détecter les données nouvellement ajoutées ou modifiées. Dans la technique basée sur les journaux, les fichiers journaux du SGBD sont utilisés pour rechercher les données nouvellement ajoutées ou modifiées dans les bases de données source.

Transform

Après l'extraction des données, la transformation apporte de la clarté et des données propres basées sur un ensemble de règles allant de la transformation de base à la transformation complexe. La transformation des données sélectionnées en formes appropriées est également considérée comme la partie la plus compliquée de l'intégration des données. Les données extraites des sources sont validées à la fois syntaxiquement et sémantiquement pour s'assurer qu'elles sont correctes en fonction des contraintes de la source. Les approches de validation de la qualité des données et d'audit des données peuvent être utilisées à cette étape pour détecter les problèmes dans les données.

Selon le format des données, il peut y avoir beaucoup de conflits entre les sources de données. Certains de ces types de conflits sont les suivants :

— représentations de valeurs différentes (par exemple pour l'état civil),
— interprétation différente des valeurs (par exemple, unités de mesure Dollars contre Euros),
— différents niveaux d'agrégation (par exemple, ventes par produit contre ventes par groupe de produits),
— référence à différents moments dans le temps (par exemple, les ventes actuelles d'hier pour une certaine source par rapport à la semaine dernière pour une autre source)

Cette liste peut être enrichie par des problèmes techniques de bas niveau, tels que des conflits sont les suivants :

— conversions de types de données,
— appliquer des masques de format,
— affecter des champs à un numéro d'ordre,
— substitution de constantes,
— définir des valeurs sur NULL ou DEFAULT en fonction d'une condition,

— en utilisant des opérateurs SQL simples (par exemple, UPPER , TRUNC , SUBSTR , etc.)

Ainsi, les tâches de transformation, qui opéreront sur les données extraites, devraient résoudre les problèmes supérieurs. En conséquence, un ensemble d'activités doit être effectué, telles que :

— reformatage,
— recalculer,
— modifier les structures de clés,
— ajouter un élément de temps,
— identification des valeurs par défaut,
— fournir une logique pour choisir entre plusieurs sources,
— résumer,
— fusionner des données provenant de plusieurs sources, etc.

Load

Le chargement des données dans l'entrepôt de données relève de la responsabilité de l'étape finale, appelée chargement [20]. Le chargement des enregistrements source dans la table appropriée marque l'achèvement du processus ETL pour ces enregistrements. Le processus de chargement varie considérablement en fonction des exigences organisationnelles. Certains entrepôts de données peuvent remplacer les données existantes par de nouvelles données sur une base quotidienne, hebdomadaire ou mensuelle, tandis que d'autres entrepôts de données peuvent conserver l'historique des données en ajoutant de nouvelles données à intervalles réguliers. La phase de chargement est souvent mise en œuvre à l'aide de tâches de chargement qui transforment entièrement ou progressivement les données de DSA vers l'entrepôt de données. Le chargement complet transforme l'intégralité des données du DSA, tandis que le chargement incrémentiel met à jour les données nouvellement ajoutées ou modifiées dans l'entrepôt de données en fonction des journaux, des déclencheurs ou des horodatages définis dans le DSA. Les phases ETL, à savoir l'extraction, la transformation et le chargement, ne sont pas des tâches indépendantes et doivent être exécutées dans le bon ordre pour toutes les données données. Cependant, la parallélisation peut être obtenue si différents composants s'exécutent sur des blocs de données distincts. Par exemple, en mode incrémental les différentes phases peuvent être exécutées simultanément ; les données nouvellement ajoutées peuvent être extraites des sources pendant que le bloc de données précédemment extrait est transformé et chargé dans l'entrepôt de données cible.

Dans un environnement d'entrepôt de données traditionnel, le processus ETL actualise périodiquement l'entrepôt de données pendant les périodes d'inactivité ou de faible charge de son fonctionnement (par exemple, chaque nuit) et dispose d'une fenêtre de temps spécifique pour se terminer. Actuellement, les besoins et les demandes des entreprises nécessitent un rafraîchissement de l'entrepôt de données en temps quasi réel et une attention particulière est portée à ce type d'avancée technologique.

2.4.2 Domaine d'application

Il est essentiel de bien formater et préparer les données afin de les charger dans le système de stockage de données de votre choix. La triple combinaison d'ETL fournit des fonctions cruciales qui sont plusieurs fois combinées en une seule application ou suite d'outils qui aident dans les domaines suivants :

- Offre un contexte historique profond pour les entreprises.
- Améliore les solutions de la business intelligence pour la prise de décision.
- Permet l'agrégation de contextes et de données afin que l'entreprise puisse générer des revenus plus élevés et/ou économiser de l'argent.
- Permet un référentiel de données commun.
- Permet de vérifier les règles de transformation, d'agrégation et de calcul des données.
- Permet une comparaison d'échantillons de données entre le système source et le système cible.
- Aide à améliorer la productivité car il codifie et réutilise sans compétences techniques supplémentaires.

D'aprés [21], il existe six (06) domaines dans lesquels les solutions ETL sont appliquées. Il s'agit notamment de l'entrepôt de données (rapporté dans 63 articles) ; la business intelligence (rapportée dans 12 articles) ; le big data (rapporté dans 4 articles) ; la santé (rapporté dans 4 articles) ; la télécommunications (rapportées dans 2 articles) et le finance (rapportées dans 2 articles).

2.4.3 Conception et modélisation des processus ETL

Les processus ETL étant complexes et coûteux, il est important de réduire leurs coûts de développement et de maintenance. La modélisation de ces processus à un niveau conceptuel contribuerait à atteindre cet objectif.

Parmi les travaux s'intéressant à la conception ETL [22], décrivent l'environnement du pro-

cessus ETL (Figure 2.5 [22]) en trois couches. Ils ont développé l'une des premières approches de modélisation basée sur un formalisme graphique non standard, implémenté sous la forme du prototype ARKTOS II. Trujillo et Luján-Mora (2003), quant à eux, ont adopté une approche plus globale pour la modélisation de l'ETL en utilisant des choix standardisés, notamment en s'appuyant sur une extension de la notation UML (Unified Modeling Language). El Akkaoui et Zemányi (2009) ont choisi d'utiliser la notation standard BPMN (Business Process Model and Notation), dédiée à la modélisation des processus métier. Ce travail a été poursuivi par El Akkaoui et al. (2011), qui ont proposé un framework de modélisation reposant sur un métamodèle dans le cadre d'une architecture de développement dirigé par les modèles (MDD, Model Driven Development). Dans [23], les auteurs catégorisent les travaux de recherche en six (06) grandes classes, selon le formalisme de modélisation sur lequel ils s'appuient :

1. La modélisation des processus ETL avec UML ;
2. Modélisation des processus ETL basée sur l'ontologie ;
3. La modélisation des processus ETL par l'architecture dirigée par les modèles (MDA) [24] ;
4. Modélisation de processus ETL basée sur le flux graphique (BPMN [25], CPN [26], YAWL (YAWL (Yet Another Workflow Language) [27]) et le flux de visualisation de données) ;
5. La modélisation des processus ETL basée sur des formalismes ad hoc, qui incluent des constructions conceptuelles, CommonCube et EMD ;
6. La modélisation des processus ETL pour le Big data.

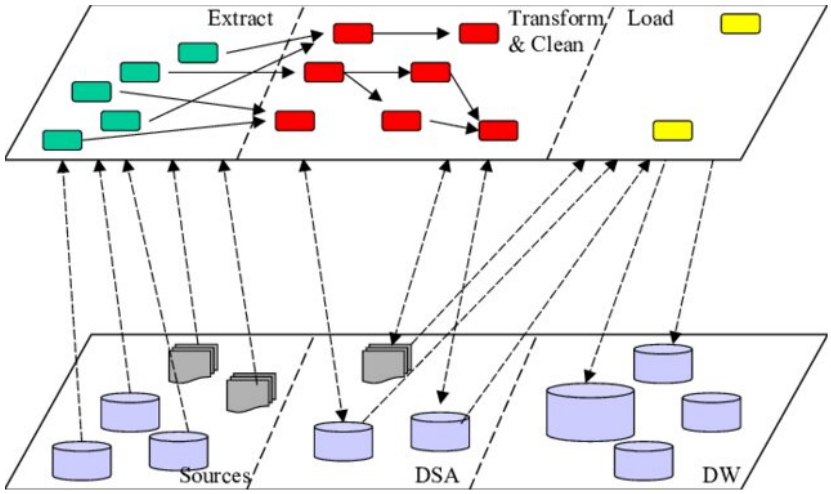

FIGURE 2.5 – *Environnement du processus Extract-Transform-Load (ETL).*

Modélisation des processus ELT basé sur UML

UML, ou Unified Modeling Language (Langage de Modélisation Unifié), est un langage standard de modélisation utilisé dans le génie logiciel. Il offre une notation graphique pour décrire, spécifier et visualiser les aspects d'un système logiciel, facilitant ainsi la communication et la conception de logiciels de manière structurée et standardisée.

Par conséquent, UML est utilisé comme langage de modélisation standard pour définir les activités ETL les plus courantes (notamment : l'intégration de données, la transformation d'attributs entre les magasins de données source et cible, ainsi que la génération de clés de substitution). Les activités ETL sont représentées par des packages UML pour modéliser un grand flux de travail ETL sous forme de plusieurs packages, simplifiant ainsi la complexité d'un flux de travail ETL pour le développeur ETL. La figure 2.6 représente le flux inférieur de l'exemple en cours d'exécution en tant que modèle conceptuel basé sur UML d'un flux de travail ETL utilisant les icônes de stéréotype définies.

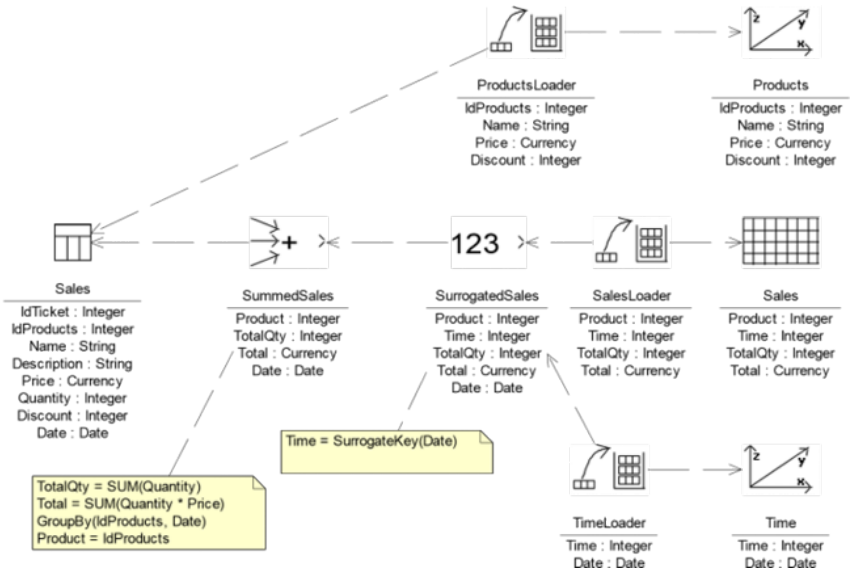

FIGURE 2.6 – *Processus ETL basée sur UML.*

Modélisation du processus ETL basée sur BPMN

La modélisation des processus métiers peut varier de simples schémas dessinés à la main à des diagrammes plus complexes avec des éléments extensibles pour une mise en œuvre détaillée. Dans sa forme avancée, la BPMN est réalisée par des analystes certifiés par l'Object Management Group (OMG). L'OMG propose cinq certifications OCEB 2 en BPMN 2.0, dénommées OMG-Certified Expert in BPM 2.0, couvrant les aspects business et techniques. L'objectif de l'OMG est de normaliser la modélisation des processus s de la même manière que

l'Unified Modeling Language (UML) a standardisé la modélisation des logiciels. Un résumé de la plupart des éléments BPMN est présenté à la figure 2.7.

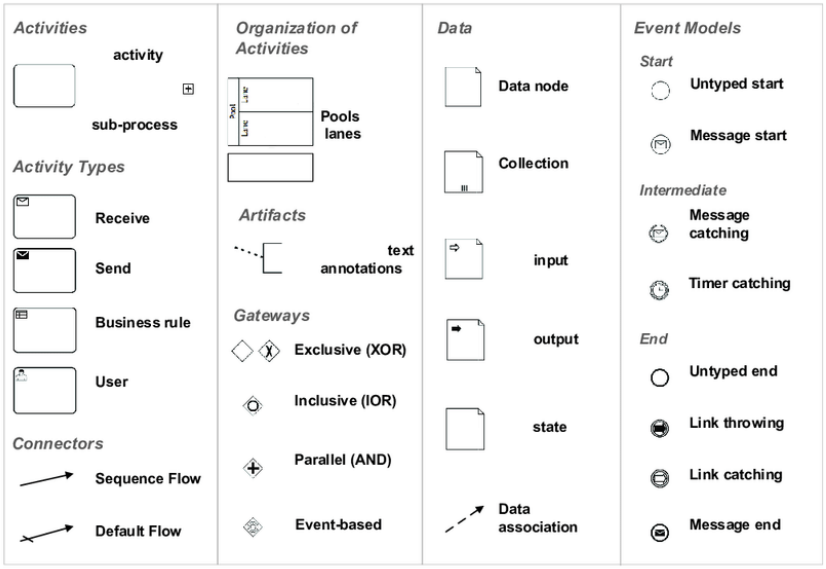

FIGURE 2.7 – *Eléments BPMN.*

Pour offrir plus d'abstraction, la modélisation conceptuelle des processus ETL utilisant la notation BPMN est suggérée par les auteurs. En utilisant Business Process Modeling Notation, les auteurs de [28], proposent une modélisation conceptuelle des processus ETL. La question de la mise à jour des dimensions à évolution lente (SCD) avec des dépendances, ou la situation dans laquelle la mise à jour d'une table SCD a un effet sur les tables SCD connectées, est abordée dans ce travail. Dans le cadre de l'entreposage de données multi-modèles et pour améliorer les performances de l'ETL, diverses options se présentent pour représenter logiquement les dimensions et les faits. Le document traite de plusieurs opérateurs BPMN pour représenter diverses activités ETL. Par exemple, les passerelles BPMN représentent la séquence d'activités dans un flux de travail ETL, basée sur des conditions et des contraintes ; Les événements BPMN représentent les événements de début, de fin et de gestion des erreurs, Les objets de connexion BPMN représentent le flux d'activités, Les artefacts BPMN décrivent la sémantique d'une tâche ETL.

Modélisation dirigée par les modèles pour la modélisation des processus ETL (MDD)

Selon [29], MDD est une approche du développement logiciel dans laquelle des modèles étendus sont créés avant l'écriture du code source. Comme le montre la figure 2.8 [30], l'approche MDD définit quatre couches principales (voir Meta-Object Facilities (MOF)) : la couche Model Instance (Mo), la couche Model (M1), la couche Meta-Model (M2), et la couche Méta-

Méta-Modèle (M3). La couche d'instance de modèle (M0) est une représentation du système du monde réel où la conception et la mise en œuvre du processus ETL sont censées s'exécuter. Cela peut être représenté, respectivement, par une interface graphique indépendante du fournisseur et par un moteur ETL spécifique au fournisseur. Au niveau de la couche Modèle (M1), le modèle de processus ETL est conçu et le code de processus ETL est dérivé en appliquant un ensemble de transformations, passant ainsi de la conception à la mise en œuvre. La couche méta-modèle (M2) se compose du métamodèle BPMN4ETL qui définit les modèles ETL lors de la phase de conception et d'une grammaire 4GL lors de la phase de mise en œuvre.

Enfin, le niveau Méta-Méta-Modèle (M3), correspond au méta-métamodèle MOF en phase de conception, alors qu'il correspond au Backus Naur Form (BNF) en phase d'implémentation.

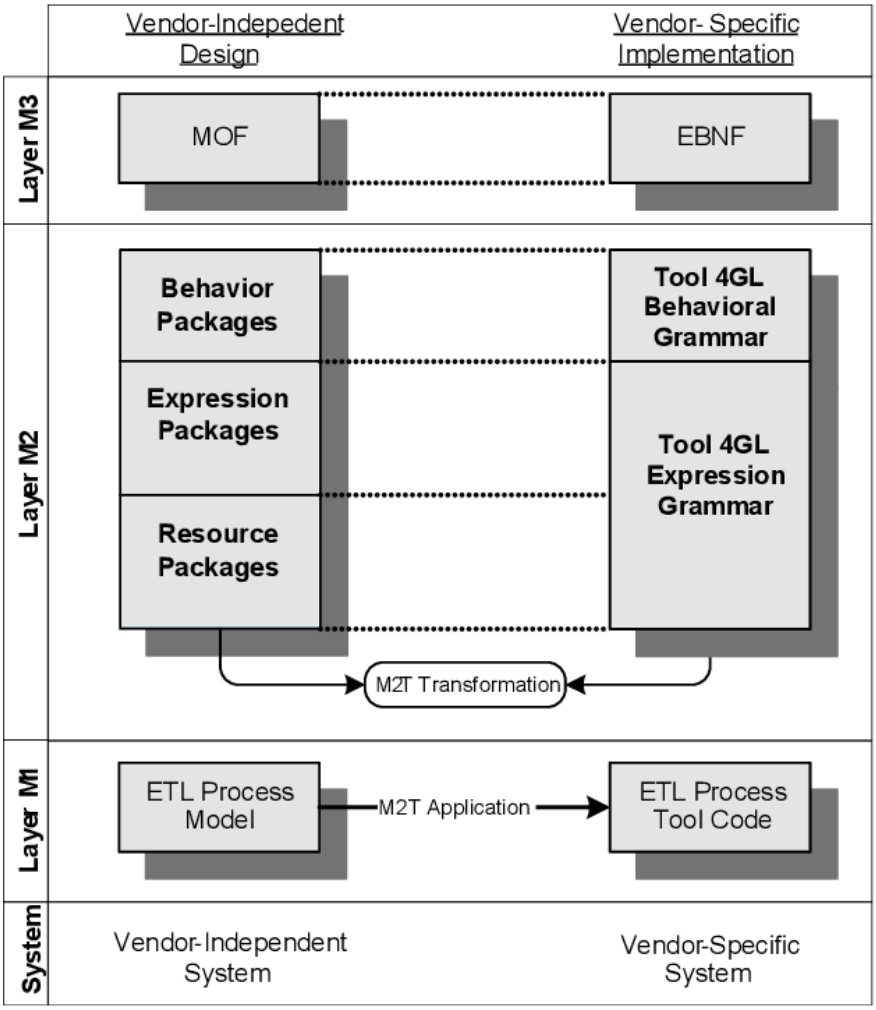

FIGURE 2.8 – *Couches MDD*

Modélisation des processus ELT pour le Big Data

Les approches traditionnelles de modélisation des processus ETL utilisées dans les systèmes d'information opérationnels classiques sont inadaptées à l'évolution du "Big Data".

Par conséquent, de nouvelles méthodes doivent être mises en place pour gérer les immenses volumes de données sources, les données semi-structurées et non structurées, le traitement complexe des données, la prise de décision rapide et la garantie de qualité des données.

Malgré plusieurs études abordant le Big Data, la plupart se sont principalement concentrées sur le déploiement des technologies, négligeant souvent l'importance de la modélisation conceptuelle. Dans la suite, nous apporterons quelques contributions récentes dans ce domaine.

walha2021design, ont abordé l'analyse du contenu généré par l'utilisateur (UGC) pour l'analyse de la décision. Dans ce contexte, ils ont proposé ETL4Social, une approche de modélisation des processus ETL basé sur le contenu des réseaux sociaux. Dans [31], les auteurs ont introduit une approche pour extraire, transformer et charger des sources de données NoSQL à la demande. Ils ont décrit l'architecture générale de la démarche qui, dans un premier temps, extrait le schéma global de chaque collection. Ils ont assuré une correspondance entre les schémas globaux de ces collections et leurs attributs multidimensionnels lors de la seconde étape. Dans cette approche, il n'y a ni implémentation ni une étude de cas pour son efficacité.

2.4.4 Outils ETL

En raison de la grande difficulté à concevoir et à gérer ces processus ETL, il y a eu récemment une prolifération du nombre d'outils ETL disponibles qui tentent de simplifier cette tâche. Sur le marché, il existe une variété d'outils ETL (Figure 2.9) qui proviennent soit d'un outil open source, soit d'un outil propriétaire. Nous avons considéré quelques outils réputés pour notre étude afin d'effectuer un benchmark de ces outils et des comparaisons dans nos contributions.

Pentaho ETL, également connu sous le nom de Pentaho Kettle, est un outil développé par Pentaho Corporation basé aux États-Unis. Il s'agit d'une plateforme open source qui offre des fonctionnalités de Business Intelligence et d'intégration de données. Les transformations effectuées dans Pentaho sont stockées en XML. Elles sont exécutées en Java.

Oracle Data Integrator (ODI) Oracle Data Integrator (ODI) est une application ETL basée sur un logiciel qui est utilisée pour la transformation, la fusion et l'intégration de données à partir de charges de travail volumineuses et exigeantes en termes de performances. Il est conçu pour gérer des événements et des processus de services de données compatibles SOA en utilisant le parallélisme pour améliorer les performances. Un élément clé de l'architecture d'ODI est son référentiel, qui rassemble toutes les métadonnées et peut être accédé en mode

client-serveur ou en mode client léger. En plus de sa fonction principale de mise en scène et de transformation de données, Oracle Data Integrator est également compatible avec d'autres logiciels Oracle, offrant ainsi une intégration complète dans l'écosystème Oracle.

Javlin Inc a développé CloverETL en 2002. Cet outil d'intégration de données, qui repose sur Java, permet de transformer et de distribuer des données, ainsi que de gérer le stockage des données. CloverETL peut être utilisé en tant qu'application autonome ou intégré à une application serveur. Il est compatible avec plusieurs plateformes.

Jaspersoft extrait et transforme les données de plusieurs systèmes et les charge dans un magasin de données à des fins de création de rapports et d'analyse. Il fonctionne comme un concepteur de tâches ETL ayant des capacités d'intégration de données.

Talend a fait son apparition en 2006. Cet outil open source basé sur Java est largement utilisé pour l'intégration de données et les processus d'analyse de données.

FIGURE 2.9 – *Outils ETL.*

Exemple ETL et Entrepôt avec Talend

L'exemple que nous illustrons dans cette section, est basé sur les besoins fonctionnels exprimés par le service scolarité de l'institut spécialisé de formation professionnelle (INSFP BBA01-Bordj Bou Arreridj- Algerie), il consiste à avoir des informations décisionnelles sur les étudiants de l'institut, à savoir leurs résultats par date, module et moyenne. Nous avons déployé les bases de données (sous ensemble de données de test délivrées par l'administration de l'institut) créées dans le système de base de données opérationnel du service scolarité. La source S1 a été créée avec MYSQL et la source S2 avec le tableur Ms-Excel. Les sources de données : S1 contient les tables : Etudiant, Module et Date. S2 est un fichier Excel qui contient les résultats des étudiants dans les examens, les travaux dirigés et les travaux pratiques. L'extraction, la transformation et le chargement dans un entrepôt de données, dont le modèle multidimensionnel est illustré sur la figure 2.11 sont réalisés via l'outil Talend (Talend open

studio est un projet open source d'intégration de données basé sur eclipse RCP qui prend principalement en charge les implémentations orientées ETL).

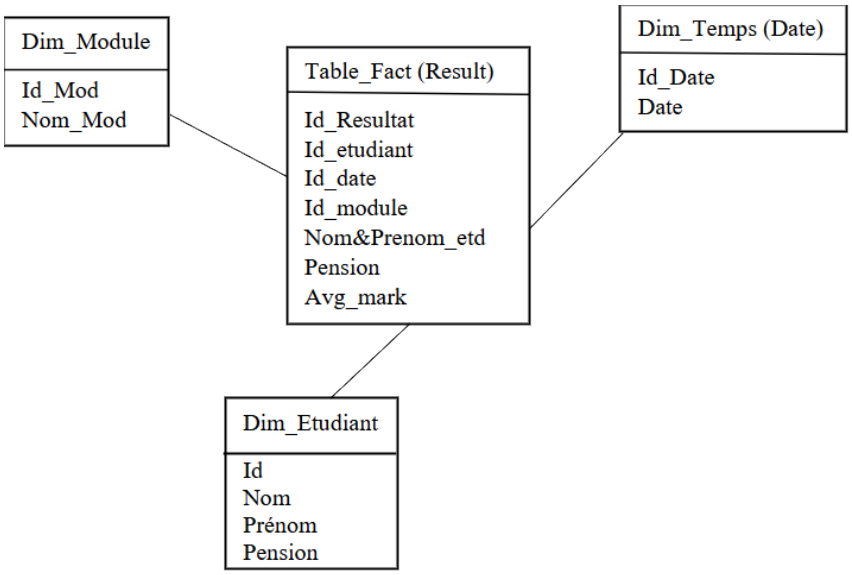

FIGURE 2.10 – *ETL & Entrepôt avec Talend (le Modèle multidimensionnel).*

On passe alors d'une base de données opérationnelle avec un fichier Excel vers une base de données intégrée, dans laquelle on peut décider et faire des analyses et des rapports, pour savoir l'échec ou le succès de l'étudiant, ainsi que ses tendances et son niveau scientifique.

Phase d'Extraction : on charge les sources de données à travers le job Extraction 01 (figure **??** : Screenshot from TOD) dans une base de données MYSQL (Stagging Area) Institut_bd.

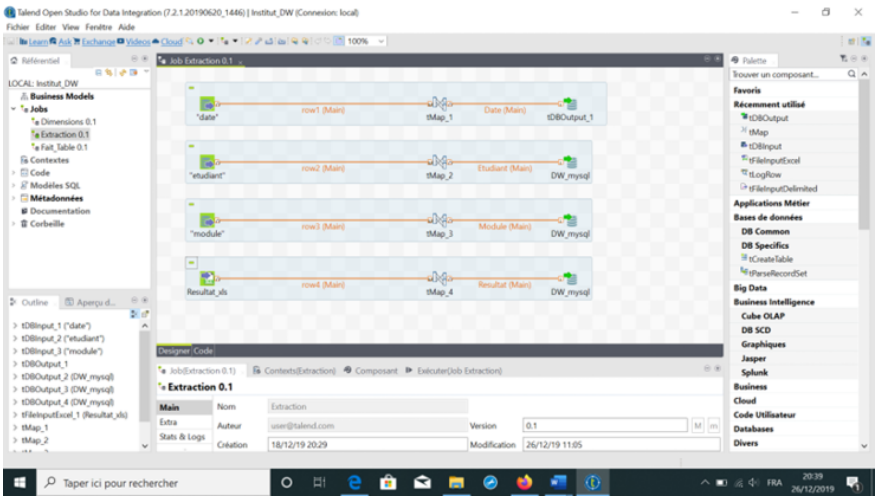

FIGURE 2.11 – *Phase extraction.*

Phase de Transformation : elle consiste à réaliser des transformations et créer les dimensions (2.12 2.13). Ici nous créons les trois dimensions de notre modèle dimensionnel, à savoir : «Etudiant» ; «Date» ; et «Module». Les transformations subies sont : (1) pour la dimension «Etudiant», fusionner le nom et le prénom dans un seul champ ; (2) pour la dimension « Module », sélectionner uniquement les champs Id_Module et le Nom_Module ; (3) pour la di-

mension «Date», éclater le champ «date» en trois champs : «Year», « Month » et «Day» (figure 2.14).

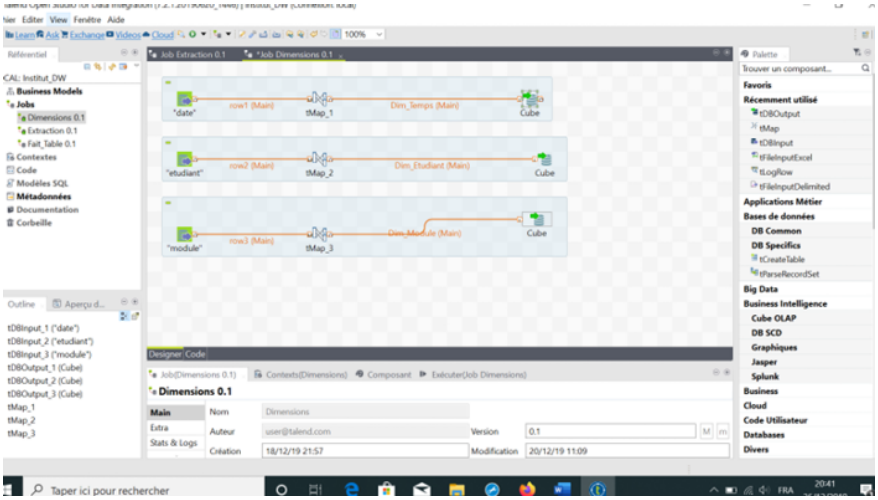

FIGURE 2.12 – *Phase transformation : Job dimension .*

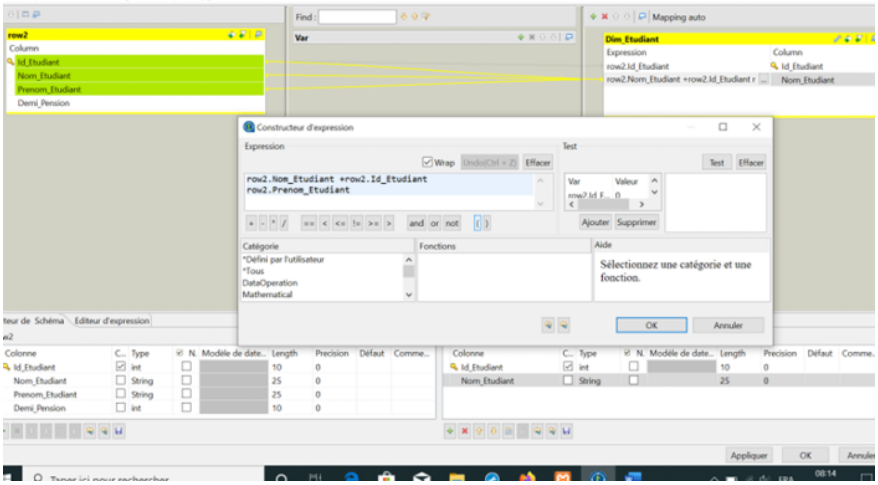

FIGURE 2.13 – *Phase transformation .*

Les dimensions sont chargées dans une base de données que nous créons dans MYSQL : cube_olap. **Phase de chargement :** Dans la dernière étape, la création du job « Fait_table » pour générer la table de faits comme illustré dans la figure 2.15. On doit récupérer le schéma des dimensions à partir de la base cube_olap, et du Stagging Area. Une seule mesure qu'on a prise c'est la moyenne (figure 2.16).

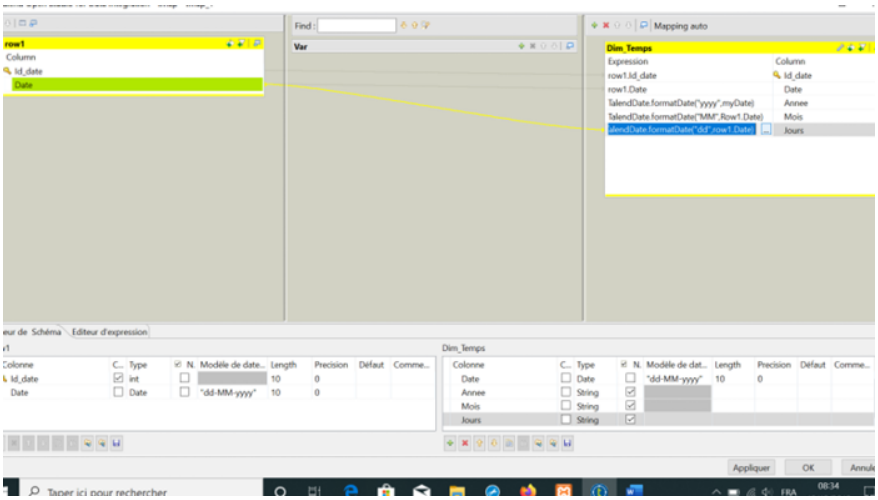

FIGURE 2.14 – *Phase transformation : Hiérarchie du champ «date».*

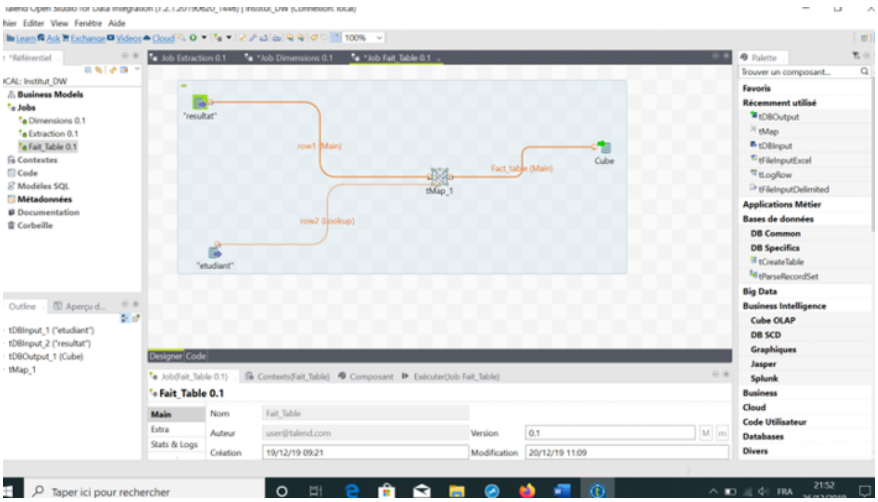

FIGURE 2.15 – *Phase chargement : Job Table_Fait.*

Enfin les résultat dans la table de fait sont illustrés à la figure 2.17 :

Après avoir chargé notre base de données (entrepôt), nous pouvons maintenant créer un cube OLAP sur la base de cet entrepôt de données. Nous utilisons pour cela l'outil Pentaho [32].

Enfin le cube est prêt pour réaliser des rapports. Nous avons utilisé l'outil Jasperserver [33].

Configurations matérielle et logicielle utilisée : Un ordinateur portable, Intel i5 8th Gen, 8G de Ram et 256 SSD de disque, Windows 10 ; le Sgbd Mysql ; le serveur local apache ; Talend Open Studio 7 ; Pentaho 3.14 et Jasperserver 7.

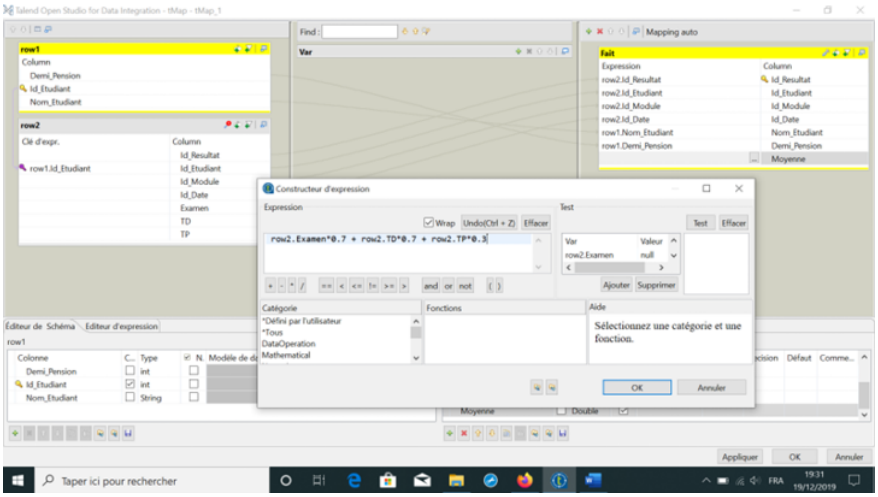

FIGURE 2.16 – *Phase chargement : Screenshot Mesure moyenne.*

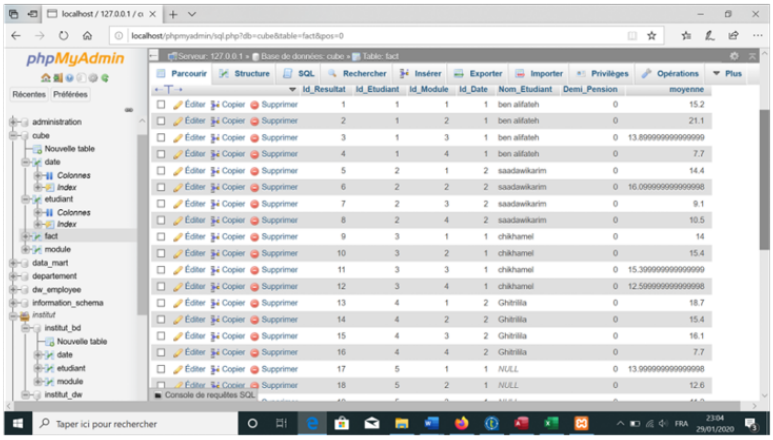

FIGURE 2.17 – *Résultats : Table de fait.*

2.4.5 Système de source de données

Les données sont conservées dans des formats structurés et non structurés. L'intégration de données structurées et non structurées implique de lier des informations communes entre elles, qui sont probablement représentées sous forme de données de base ou de clés dans des données structurées dans des bases de données et sous forme de balises de métadonnées ou de contenu intégré dans des données non structurées [34]. Une source de données, dans le contexte de l'informatique et des applications informatiques, est l'endroit d'où proviennent les données utilisées. La source de données peut être des fichiers plats tels que Excel, des fichiers XML ou des sites Web, des applications logicielles, etc. Les données structurées sont essentiellement stockées dans les systèmes d'entreprise (ES), y compris la planification des ressources d'entreprise (ERP) et la gestion de la relation client (CRM) pour enregistrer toutes les transactions qui ont été entreprises. Les entreprises stockent de grandes quantités de don-

nées sur différentes plates-formes, grâce à ETL, l'organisation peut obtenir une vue globale qui lui permet de faire de meilleures affaires stratégiques.

Les fichiers plats sont des fichiers sans hiérarchie interne, pouvant être constitués de données de table unique comprenant des lignes et des colonnes. Ce sont de simples fichiers de données au format texte ou binaire. Les fichiers plats peuvent être des transactions, des données de séries chronologiques au format Excel, CSV, TSV, XML, etc.

Les sources en ligne telles que le contenu de la page Web, les sites de médias sociaux et les publications en ligne constituent une autre source de données essentielle pour les entreprises. L'extraction de données à partir de ces sources en ligne sera très bénéfique pour les propriétaires d'entreprise.

Les données multimédias sont des données à extraire des médias sociaux en ligne, par ex. tweets, messages, légendes. Ils sont généralement considérés comme des données complexes avec une grande dimensionnalité et une structure incohérente. Par conséquent, ils sont tenus de faire progresser les techniques de traitement et d'exploration de données vers des modèles non cachés dans leurs données et d'extraire des connaissances.

Les données traitées et préparées pour l'intégration via les processus ETL dans un entrepôt de données proviennent notamment :

— du Web : les textes, les images, les vidéos, et tous ce qui peut figurer sur les pages Web,
— de l'internet et des objets communicants : réseaux de capteurs, journaux des appels en téléphonie,
— des sciences : génomique, astronomie, physique subatomique, etc.,
— données commerciales,
— données personnelles (ex : dossiers médicaux),
— données publiques (open data).

2.5 L'Hétérogeneité

La tâche la plus difficile pour les processus ETL consiste à gérer des données hétérogènes dans différents formats pour une analyse en ligne [35].

L'hétérogénéité est l'un des principaux facteurs qui augmentent la complexité pour assurer l'intégration entre différentes sources de données. Dans le contexte de l'écosystème big data, les processus ETL doivent prendre en compte trois enjeux principaux comme le montre la figure 2.18 : (i) la variété des sources de données, y compris les bases de données tradition-

nelles, XML, la sémantique, les graphes, etc. ; (ii) la variété des plates-formes de stockage, où le système hôte peut contenir de nombreux magasins (appelés polystores), chacun hébergeant un type spécifique de données ; et (iii) l'évolutivité ou la scalabilité (la capacité des systèmes de traitement de données à augmenter leurs capacités de traitement à mesure que les données augmentent).

FIGURE 2.18 – *Hétérogénéité des sources de données et des plateformes de stockage.*

L'intégration de données hétérogènes a conduit à de nombreux problèmes qui peuvent être classés en deux catégories : l'intégration de données hétérogènes et l'évaluation des requêtes [36].

2.6 Données massives (big data)

2.6.1 Définition

De nos jours, les organisations génèrent de plus en plus de grandes quantités de données dans une grande variété de formats à haut débit (Figure 2.19). Nous sommes à l'ère du big data [37]. Les chercheurs définissent le big data par les quatre V suivants : Volume, Variété, Vélocité et Véracité. Ces quatre dimensions caractérisent et distinguent le big data des données ordinaires. Aussi, la croissance rapide d'une base de données de graphes (données RDF) est une excellente opportunité d'enrichir les entrepôts de données traditionnels avec une nouvelle dimension en V du big data : la valeur. Ainsi, permettre aux entreprises d'exploiter ces données riches pour accroître leur valeur ajoutée dans un monde hautement concurrentiel.

2.6.2 Scalabilité : Passage à l'échelle

La plupart des entreprises considèrent les projets liés à l'intégration des données et au big data comme une priorité. Les décisions futures seront désormais basées sur les données ou

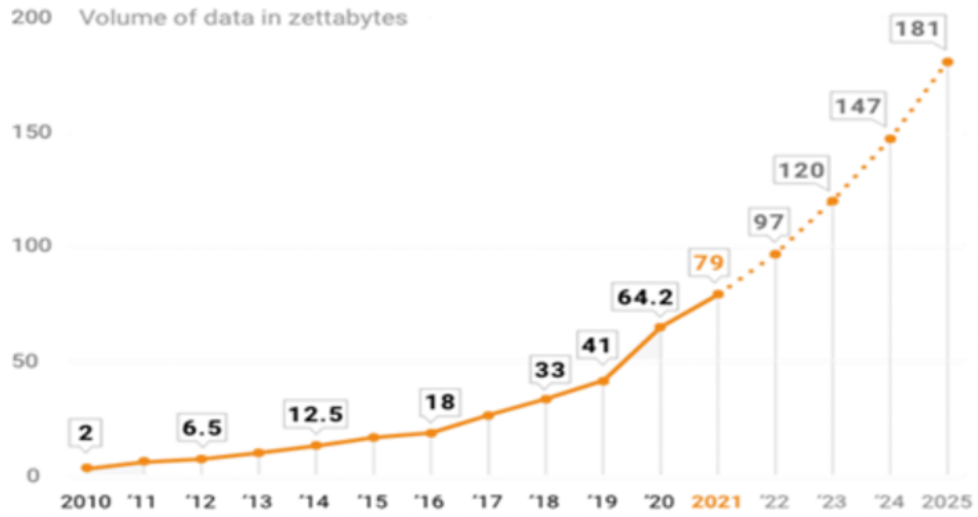

FIGURE 2.19 – *Explosion des données (Source : www.statista.com).*

elles ne seront pas prises du tout. Avec une augmentation constante du volume de données, il est essentiel de penser à la scalabilité dès le début d'un projet de traitement des données.

On distingue deux types de scalabilité d'un système :

— **L'évolutivité verticale**, également appelée mise à l'échelle verticale, implique d'améliorer les performances du matériel en ajoutant progressivement des ressources supplémentaires telles que des processeurs, de la mémoire ou de l'espace de stockage lorsque le système atteint ses limites. Le concept sous-jacent repose sur l'ajout de composants plus performants. Cependant, cette approche atteint rapidement ses limites lorsqu'un volume important de données est généré, ce qui entraîne des coûts élevés. De nos jours, cette approche est moins favorisée, du moins pas sous sa forme initiale,

— **L'évolutivité horizontale**, également appelée mise à l'échelle horizontale, consiste à augmenter la capacité de traitement en ajoutant davantage de machines. Cela se fait en augmentant le nombre de serveurs pour répartir plus efficacement la charge de travail entre eux.

2.6.3 Caractéristiques des données massives (big data)

Plusieurs règles de base sont utilisées pour définir les mégadonnées et leurs caractéristiques en termes de nombre de propriétés appelés V. Le nombre de V a évolué de 3 V en 2001 à 17 V en 2017 [38]. Chaque règle est définie par un ensemble de propriétés caractérisant le Big Data, les principales caractéristiques sont les suivantes :

Le volume

Le volume se rapporte à l'immense quantité de données a produit de chaque seconde. Pensez juste à tous les emails, messages de Twitter, clips de photo, vidéo et données de sonde que nous produisons et partageons chaque seconde. Nous ne parlons pas des Terabyte, mais des zettabytes ou des brontobytes des données. [20]

La vitesse

La vitesse se rapporte à la vitesse à laquelle de nouvelles données sont produites et à la vitesse autour derrière la à laquelle les données se déplacent. Pensez juste aux messages sociaux de media allant viraux en quelques minutes, la vitesse à laquelle des transactions de carte de crédit sont examinées pour assurer les activités frauduleuses ou les millisecondes où elle prend des systèmes de commerce pour analyser les réseaux sociaux de media pour prendre des signaux que des décisions de déclencheur pour acheter ou vendre des actions. La grande technologie de données nous permet maintenant d'analyser les données tandis qu'elle est produite sans la mettre jamais dans des bases de données.

La variété

La variété se rapporte aux différents types de données que nous pouvons maintenant employer. Dans le passé nous nous sommes concentrés sur les données structurées qui s'insèrent d'une manière ordonnée dans des tables ou des bases de données relationnelles telles que des données financières (par exemple, sous-produit ou région de ventes). En fait, 80 pour cent des données du monde sont maintenant non structurés et ne peuvent pas donc facilement être mis dans des tables ou apparenté base de données-pensez aux photos, aux ordres visuels ou aux mises à jour sociales de media. Avec la grande technologie de données nous pouvons maintenant armer les types différés de données comprenant des messages, des conversations sociales de media, des photos, des données de sonde, la vidéo ou des enregistrements de voix et les apporter ainsi que des données plus traditionnelles et plus structurées.

La véracité

La véracité se réfère au désordre de la fiabilité des données. Avec de nombreuses formes de données importantes, la qualité et la précision sont moins contrôlables, pour les postes exemple Twitter avec hashtags, les abréviations, les fautes de frappe et le langage familier. La

technologie big data et d'analyse nous permet maintenant de travailler avec ces types de données. Les volumes font souvent le manque de qualité ou l'exactitude. Mais tous les volumes de données rapide mouvement de la variété et de la véracité différente doivent être transformés en valeur ! Ceci est la raison pour laquelle la valeur est celle de V données volumineuses qui importe le plus.

La valeur

Si nous produisons tellement des données, alors quelle quantité de lui est réellement utile ? La compréhension de la valeur de ces données est cruciale de sorte que le processus puisse être optimisé. L'investissement dans de grandes solutions de données est cher ainsi il est très important de s'assurer que les données étant extraites sont précieuses pour la société.

2.7 Outils du Big data

2.7.1 Hadoop

Le système de fichiers distribué Hadoop (HDFS) est un système de fichiers distribué conçu pour fonctionner sur du matériel standard. Il présente de nombreuses similitudes avec les systèmes de fichiers distribués existants. Cependant, les différences avec les autres systèmes de fichiers distribués sont importantes. HDFS est hautement tolérant aux pannes et est conçu pour être déployé sur du matériel à faible coût. HDFS fournit un accès à haut débit aux données d'application et convient aux applications qui ont de grands ensembles de données [39].

Hadoop est un framework open source d'Apache et est utilisé pour stocker, traiter et analyser des données très volumineuses. Hadoop est écrit en Java et n'est pas OLAP (traitement analytique en ligne). Il est utilisé pour le traitement par lots/hors ligne. Il est utilisé par Facebook, Yahoo, Google, Twitter, LinkedIn et bien d'autres. De plus, il peut être mis à l'échelle simplement en ajoutant des nœuds dans le cluster.

Les Modules de Hadoop

— HDFS : système de fichiers distribué Hadoop. Google a publié son article GFS (google file system) [40] et sur la base de celui-ci HDFS a été développé. Il indique que les fichiers seront divisés en blocs et stockés dans des nœuds sur l'architecture distribuée.

— Yarn : Un autre négociateur de ressources est utilisé pour la planification des travaux et la gestion du cluster.
— Map Reduce : Il s'agit d'un cadre qui aide les programmes Java à effectuer le calcul parallèle sur les données à l'aide d'une paire clé-valeur. La tâche Map prend les données d'entrée et les convertit en un ensemble de données qui peut être calculé dans la paire de valeurs clés. La sortie de la tâche Map est consommée par la tâche de réduction, puis la sortie du réducteur donne le résultat souhaité.
— Hadoop Common : Ces bibliothèques Java sont utilisées pour démarrer Hadoop et sont utilisées par d'autres modules Hadoop.

Différences entre FileSystem standard et HDFS

Système de fichiers régulier : dans le système de fichiers standard, les données sont conservées dans un seul système. Si la machine tombe en panne, la récupération des données est difficile en raison de la faible tolérance aux pannes. Le temps de recherche est plus long et il faut donc plus de temps pour traiter les données.

HDFS : Les données sont distribuées et conservées sur plusieurs systèmes. Si un DataNode tombe en panne, les données peuvent toujours être récupérées à partir d'autres nœuds du cluster. Le temps nécessaire pour lire les données est comparativement plus long, car il y a des données locales lues sur le disque et une coordination des données provenant de plusieurs systèmes.

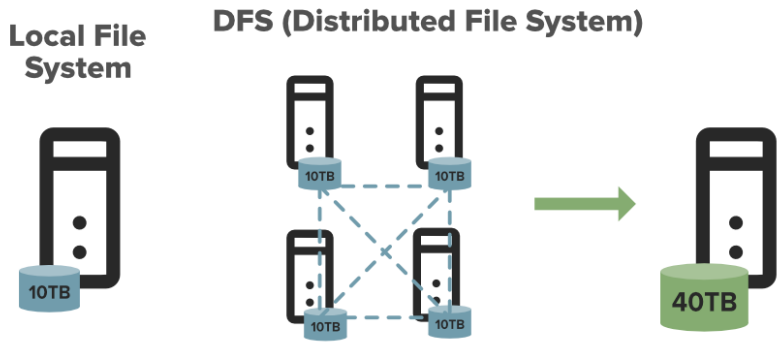

FIGURE 2.20 – *HDFS vs FS.*

2.7.2 Apache Spark

Apache Spark est un puissant framework de traitement de données distribué et évolutif, conçu pour traiter des volumes massifs de données de manière rapide et efficace. Il offre une interface conviviale et des fonctionnalités avancées pour la manipulation, l'analyse et le traitement de données en temps réel, ainsi que pour les tâches de calcul par lots. Spark a un modèle de programmation similaire à MapReduce mais l'étend avec une abstraction de partage de données appelée "jeux de données distribués résilients" ou RDD [41]. Spark exploite le concept de RDD, qui permettent la manipulation et la transformation de données de manière distribuée et tolérante aux pannes. Il prend en charge plusieurs langages de programmation, tels que Scala, Python et Java, et offre des bibliothèques étendues pour des tâches telles que le traitement de flux, le traitement de graphes, l'apprentissage automatique et l'analyse de données. Grâce à son architecture distribuée et à sa capacité à optimiser les calculs, Apache Spark offre des performances élevées et une extensibilité pour répondre aux besoins des applications de traitement de données les plus exigeantes.

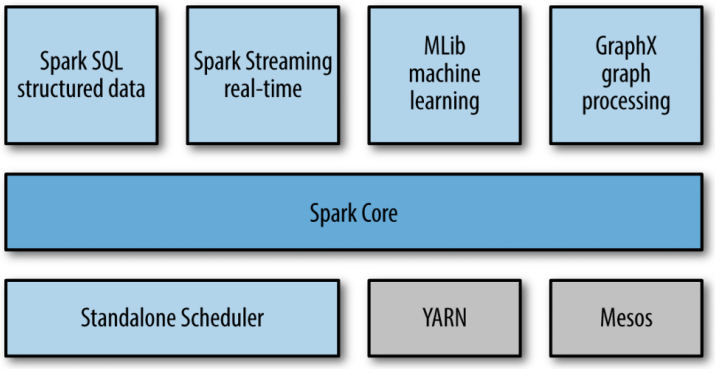

FIGURE 2.21 – *La pile Spark.*

Bibliothques de Spark

Lorsque vous disposez de petites données, il est possible de les analyser avec un seul ordinateur dans un délai raisonnable. Lorsque vous disposez de gros volumes de données, utiliser un seul ordinateur pour analyser et traiter ces données (et les stocker) peut s'avérer extrêmement lent, voire impossible. C'est pourquoi nous souhaitons utiliser Spark. Spark dispose d'une bibliothèque principale et d'un ensemble de bibliothèques intégrées (SQL, GraphX, Streaming, MLlib), comme le montre la figure 2.22. Comme vous pouvez le constater, grâce à son API DataSource, Spark peut interagir avec de nombreuses sources de données, telles que Hadoop, HBase, Amazon S3, Elasticsearch et MySQL, pour n'en citer que quelques-unes. Cette figure montre la véritable puissance de Spark : vous pouvez utiliser plusieurs langages

différents pour écrire vos applications Spark, puis utiliser de riches bibliothèques pour résoudre divers problèmes de big data. Pendant ce temps, on peut lire/écrire des données à partir de diverses sources de données.

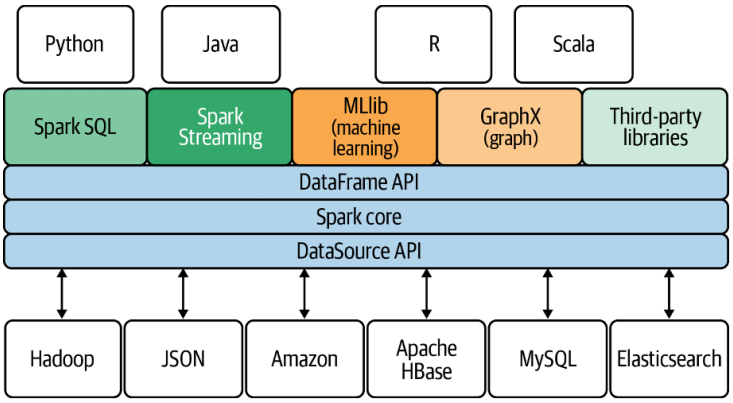

FIGURE 2.22 – *Les bibliothèques de Spark.*

Pourquoi Spark ?

Spark est un puissant moteur d'analyse qui peut être utilisé pour le traitement de données à grande échelle [42]. Les raisons les plus importantes d'utiliser Spark sont :

1. Spark est simple, puissant et rapide.
2. Spark est gratuit et open source.
3. Spark fonctionne partout (Hadoop, Mesos, Kubernetes, autonome ou dans le cloud).
4. Spark peut lire/écrire des données depuis/vers n'importe quelle source de données (Amazon S3, Hadoop HDFS, bases de données relationnelles, etc.).
5. Spark peut être intégré à presque toutes les applications de données.
6. Spark peut lire/écrire des données dans des formats basés sur des lignes (comme Avro) et des colonnes (comme Parquet et ORC).
7. Spark dispose d'un ensemble d'API riche mais simple pour tous les types de processus ETL.

La pile Spark

illustrés à la figure 2.21 [43], comprend plusieurs composants clés qui fonctionnent ensemble pour permettre le traitement et l'analyse de données à grande échelle. Au cœur de la pile se trouve le moteur de traitement Spark, qui gère la planification, l'exécution et l'optimisation des tâches de traitement. Il s'appuie sur les RDDs (Resilient Distributed Datasets) pour

la manipulation distribuée des données. Au-dessus du moteur de traitement, Spark offre un large éventail de bibliothèques et de modules, tels que Spark SQL pour le traitement des données structurées, Spark Streaming pour le traitement des flux de données en temps réel, MLlib pour l'apprentissage automatique, et GraphX pour le traitement et l'analyse de graphes. Enfin, Spark s'intègre également à d'autres outils et technologies, tels que Hadoop, pour tirer parti de leur écosystème et de leur capacité de stockage distribué. Grâce à cette architecture modulaire et à la richesse de ses composants, Spark offre une solution complète et polyvalente pour le traitement et l'analyse de données massives.

Modèle de programmation Spark

L'abstraction de programmation clé dans Spark est les RDD, qui sont des collections d'objets tolérantes aux pannes partitionnées sur un cluster qui peuvent être manipulées en parallèle. Les utilisateurs créent des RDD en appliquant des opérations appelées « transformations » (telles que map, filter et groupBy) à leurs données.

Spark expose les RDD via une API de programmation fonctionnelle dans Scala, Java, Python et R, où les utilisateurs peuvent simplement transmettre des fonctions locales à exécuter sur le cluster.

Par exemple, le code Scala suivant crée un RDD représentant le messages d'erreur dans un fichier journal, en recherchant les lignes commençant par ERROR, puis en imprime le nombre total d'erreurs :

1. lignes = spark.textFile(“hdfs ://...”)
2. erreurs = lignes.filter(s => s.startsWith("ERROR"))
3. println("Erreurs totales : " + erreurs.nombre())

La première ligne définit un RDD soutenu par un fichier dans le système de fichiers distribués Hadoop (HDFS) comme une collection de lignes de texte. La deuxième ligne appelle la transformation de filtre pour dériver un nouveau RDD à partir des lignes. Son argument est un littéral ou une fermeture de fonction Scala. Enfin, les appels de la dernière ligne comptent, un autre type d'opération RDD appelée "action" qui renvoie un résultat au programme (ici, le nombre d'éléments dans le RDD) au lieu de définir un nouveau RDD.

Spark exploite l'évaluation paresseuse des RDD, permettant une planification de calcul efficace. Les transformations créent de nouveaux RDD sans calcul immédiat, tandis que les actions déclenchent Spark pour optimiser le plan d'exécution. Les RDD prennent également

en charge le partage de données et peuvent être conservés en mémoire pour une réutilisation plus rapide. erreurs.persist() Après cela, l'utilisateur peut exécuter diverses requêtes sur les données en mémoire :

1. Compte les erreurs mentionnant les erreurs MySQL.filter(s => s.contains("MySQL")).count(),
2. Récupère les champs d'heure des erreurs qui mentionnez PHP, en supposant que l'heure est le champ 3 : errors.filter(s => s.contains("PHP")).map(line =>line.split('\t')(3)).*collecter*().

La figure 2.23 [44] montre les RDD dans notre requête précédente, où nous obtenons les champs temporels des erreurs mentionnant PHP en appliquant deux filtres et une carte.

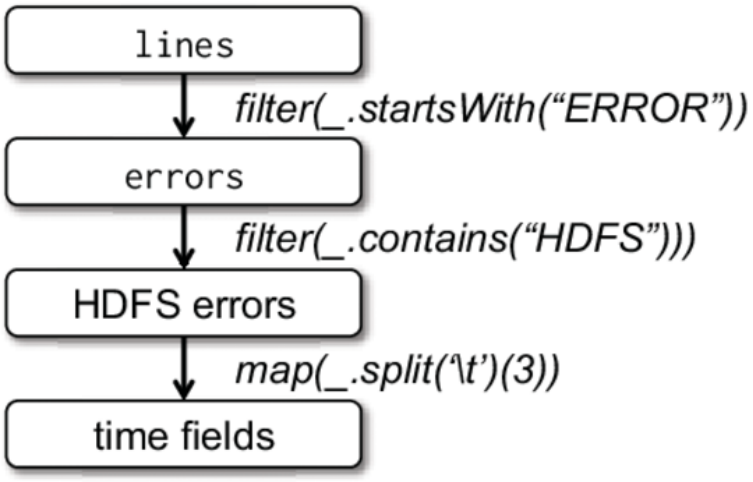

FIGURE 2.23 – *Exemple de graphique de lignée.*

Stockage pour Spark

Spark peut créer des ensembles de données distribués à partir de n'importe quel fichier stocké dans le système de fichiers distribué Hadoop (HDFS) ou d'autres systèmes de stockage pris en charge par les API Hadoop (y compris votre système de fichiers local, Amazon S3, Cassandra, Hive, HBase, etc.) [41]. Il est important de se rappeler que Spark ne nécessite pas Hadoop, il prend simplement en charge les systèmes de stockage implémentant les API Hadoop. Spark prend en charge les fichiers texte, SequenceFiles, Avro, Parquet et tout autre format d'entrée Hadoop.

2.7.3 Algorithmes parallèles

Parmi les principaux objectifs d'Apache Spark, était de permettre à un large éventail d'utilisateurs de mettre en œuvre plus facilement des algorithmes parallèles. Dans Spark, de nouveaux algorithmes agissant sur des données à grande échelle ont un impact profond dans

tous les domaines de l'informatique, nous avons mis en œuvre de tels algorithmes dans nos contributions, sans avoir à créer un système distribué à partir de zéro.

Algorithmes distribués : Exemple ETL avec les DataFrames sous scala

Scala est un compilateur basé sur un langage de programmation multiparadigmes qui est compact, rapide et efficace. Le principal avantage de Scala est la JVM (Java Virtual Machine). Le code Scala est d'abord compilé par un compilateur Scala, le byte code correspondant est généré, qui sera ensuite transféré à la machine virtuelle Java pour générer la sortie. Ainsi, Scala est devenu la clé du succès pour gérer les données massives (le big data) [45].

Dans cet exemple (expérience), la direction de l'institut INSFP Bordj bou arreridj (BBA01) est confrontée à un problème lors de la prise de décision en raison de la non disponibilité de données intégrées concernant les employées. Elle ne peut pas étudier les données conformément à ses besoins. Il est donc demandé de concevoir un système qui puisse les aider rapidement dans la prise de décision. - Les données sources sont téléchargées de la plateforme Kaggle [46]. L'exemple suivant consiste à intégrer des données issues de base de données MYSQL et CSV : Source1 est une table employée du service personnel ; Source2 est une table employée du service de la paie ; et la source 3 est un fichier CSV des employées de la direction de la formation de la wilaya. L'ETL workflow du processus ETL est illustré dans la figure 2.24.

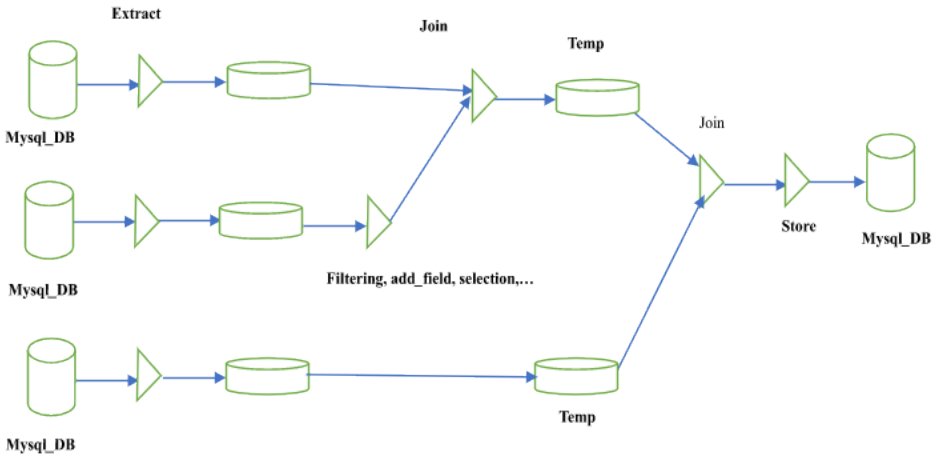

FIGURE 2.24 – *ETL workflow pour l'exemple en cours.*

Nous avons programmé et testé des jobs ETL distribués Spark en Python et en Scala en mode cluster et en local.

Algorithme 1 : La phase Extraction

```
//extraction tablemysql employees, et de la table mysql departement_employees et
 du fichier emp.csv
1. val table = "employees"
2. val properties = new Properties();
3. properties.put("user", "root");
4. properties.put("password", "")
5. Class.forName("com.mysql.jdbc.Driver")
   // departement_employees
6. val url1 = "jdbc :mysql ://localhost :3306
   //departement";
7. val table1 = "departement_employees"
8. val emp_salaires_depDF = spark.read.jdbc(url1, table1, properties)
   // emp.csv
9. val csvdata_file = spark.read.option("header", "true").option("sep",
   ",").option("infercSchema", "true").csv("C :/dataset/emp.csv").cache();
```

Algorithme 2 : Effectuer des transformations, ex. ajout d'une colonne, jointure, filter et selection... etc

```
1. val csvnouveaushemaDF = StructType(StructField("id", IntegerType, false) : :
   StructField("id_emp", IntegerType, false) : : StructField("nom", StringType, true) : : Nil )
   //appliquer le nouveau schema
2. val csv_datafile1 = spark.read.option("header", "true").option("sep", ",")
   .schema(csvnouveaushemaDF)
   .csv("C :/Users/DELL/Downloads/sf-salaries/Salaries.csv")
   csv_datafile1.printSchema()
3. val join1DF = employees_pDF.join(emp_salaires_depDF, "id_emp").select("id_emp",
   "Fonction", "salaire") //val join1DF = employees_pDF.join(emp_salaires_depDF,
   "id_emp").select("*")
4. println("————————————la jointure entre departement.employees et
   personnels.employees————————————-"); val query3 = join1DF.select("*");
   query3.show()
```

Algorithme 3 : Le chargement

```
//————-sauvegarde finale dans DW_EMP
1. val url3="jdbc :mysql ://localhost :3306/data_mart";
2. val tablemysql3 = "datamart_emp" DW_DF.write.mode(SaveMode.Overwrite).jdbc(url3,
   tablemysql3,properties)
```

L'entrepôt de données est finalement chargé et notre dataset est prêt pour effectuer des analyses et créer des rapports.

2.8 Graphes RDF

Nous rappelons ici les concepts de base et les notations liées aux graphes RDF. À première vue, ceux-ci peuvent être considérés comme des cas particuliers de graphes étiquetés et orientés, et en effet les techniques classiques de résumé de graphes ont été directement adaptées à RDF ;

2.8.1 Graphes orientés étiquetés

Les graphes orientés étiquetés sont les concepts de base Les graphes orientés étiquetés sont le concept de base permettant de modéliser les ensembles de données RDF. De plus, la plupart des propositions (pas toutes) pour résumer un graphe RDF modélisent également le résumé comme un graphe orienté. Ainsi, sans perte de généralité, nous baserons notre discussion sur ce modèle. Notez qu'il peut être facilement généralisé à des graphes plus complexes, par exemple, ceux avec des nœuds multi-étiquetés.

Étant donné un ensemble A d'étiquettes, on note G = (V , E) un graphe orienté étiqueté à arêtes A dont les sommets sont V, et dont les arêtes sont $E \subseteq VxAxV$.

La figure 1 affiche deux de ces graphiques ; Des étiquettes de bord sont attachées aux bords.

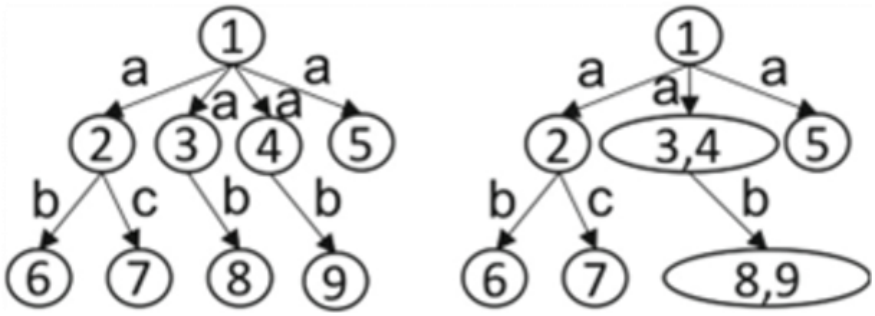

FIGURE 2.25 – *Graphes orientés étiquetés par les arêtes).*

2.8.2 Le framework de description des ressources (RDF)

RDF est le modèle de données standard promu par le W3C pour les applications Web sémantique.

2.8.3 Graphe RDF

Un graphe RDF

En bref un graphe, est un ensemble de triplets de la forme (s, p, o). Un triplet indique qu'un sujet s a la propriété p et que la valeur de cette propriété est l'objet o. Nous ne considérons que des triplets bien formés, conformément à la spécification RDF [47]. Un triplet indique que son sujet s a la propriété p et que la valeur de cette propriété est l'objet o. Ils sont utilisés pour décrire les ressources et les assertions de propriété. Le standard RDF fournit un ensemble de classes et de propriétés intégrées, avec des espaces de noms prédéfinis. Par exemple rdf :type spécifie les classes auxquelles appartient une ressource. Un graphe RDF encode une structure de graphe dans laquelle chaque triplet décrit une arête dirigée étiquetée par p du nœud étiqueté par s au nœud étiqueté par o.

Exemple (graphe RDF) extrait de du benchmark LUBM [48], le graphe RDF G suivant décrit un livre, identifié par doi1 , son auteur (un nœud vide - :b1 dont le nom est connu), son titre et sa date de publication :

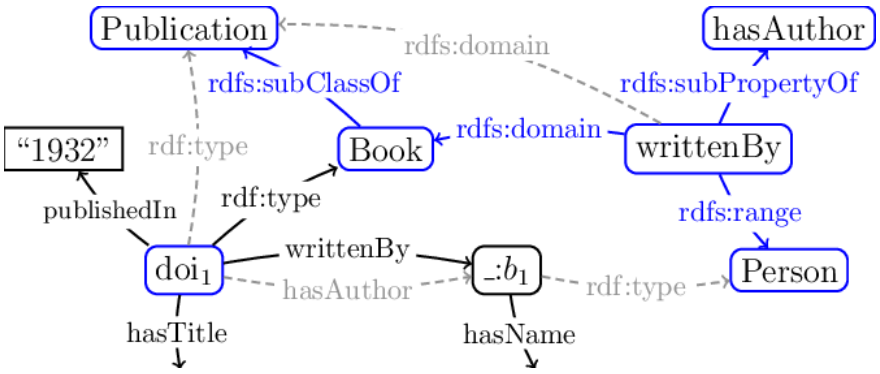

FIGURE 2.26 – *Graphes orientés étiquetés par les arêtes.*

RDF Schema (RDFS)

RDFS [49] enrichit les descriptions dans les graphes RDF, en déclarant des contraintes sémantiques entre les classes et les propriétés utilisées dans le graphe. De telles contraintes peuvent définir des relations de sous-classe, de sous-propriété, un domaine et une plage de propriétés. Les contraintes RDFS sont interprétées dans [50] sous l'hypothèse du monde ouvert (OWA).

2.9 Formalisation des bases de données

Base de données relationnelle

On défini une base de données relationnelle par un ensemble de tables, d'attributs, d'instances et de contraintes. Par conséquent, chaque source Si est définie par un ensemble d'entités E = e1, e2, . . . , em, un ensemble d'attributs liés aux entités A = a1, a2, . . . , am et un ensemble d'ensembles d'enregistrements R=r1,r2, . . . , rn.

Base de données graphe

Généralement utilisé pour représenter des bases de connaissances à travers un graphe G dont les nœuds (V), les arêtes (E) et les étiquettes (Lv, Le) représentent respectivement des classes, des instances et des propriétés de données, des propriétés d'objet et des constructeurs DL. [51] est un exemple de système de stockage de bases de données de graphes.

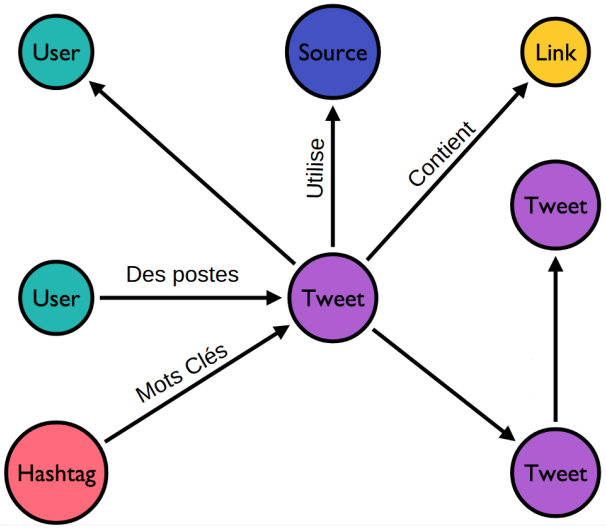

FIGURE 2.27 – *Apercu de base de données graphe Neo4j.*

2.9.1 Modélisation du graphe ETL

Habituellement, le processus ETL est représenté comme un graphe acyclique dirigé (DAG) comprenant des nœuds (symbolisant des attributs ou des instances de schéma) et des arêtes (illustrant le flux de données entre les nœuds via la taxonomie RDFS). Le flux de données est un ensemble d'opérations nécessaires à la transformation des données d'entrée. Les opérations sont effectuées sur les nœuds, produisant de nouveaux nœuds qui constituent un graphe ETL.

Définition

Pour un graph ETL, nous définissons formellement un processus ETL comme un quadruplet : **[Os-Entrée, Os-Sortie, Os-Operateur, Os-Graphe_ETL]** , où Entrée représente un ensemble fini de nœuds d'entrée (concepts et instances) correspondant au schéma de l'entrepôt et aux sources de données, Sortie est un ensemble fini de nœuds intermédiaires ou cibles (concepts et instances), Operateur est un ensemble d'opérateurs couramment rencontrés lors du processus ETL, et Graphe_ETL est un ensemble de nœuds de sortie (concepts et instances) ajoutés au schéma d'entrepôt final et liés par des bords (rôles). Dans [12], les auteurs ont établi dix (10) opérateurs ETL conceptuels universels cités dans le tableau 2.1. Nous les avons adaptés pour tenir compte des spécificités d'un graphe G représentant une base de données graphe RDF, où les nœuds (N), les arêtes (E) et les étiquettes (L) symbolisent respectivement les classes, les instances, les propriétés de données, les propriétés d'objet et les constructeurs DL.

Opérateurs	Fonction
Select	Filtre les valeurs de la propriété P, en excluant les valeurs qui n'appartiennent pas à l'ensemble spécifié par R.
Convert	Convertit les valeurs de la propriété P du format de représentation R1 au format de représentation R2.
Add	Ajouter l'attribut a au schéma actuel.
detect duplicates	Détecte et supprime de manière appropriée les enregistrements ayant la même valeur pour la propriété P.
concatenate	Concatène les valeurs des attributs a1,. . ., an pour définir la valeur de la propriété P.
Split	Divise la valeur de l'attribut a pour définir les valeurs des propriétés P1,. . ., Pn.
Aggregate	Agrége les valeurs de la ou des propriété(s) P.
Union	Comme l'opérateur « union » dans SQL.
Join	Comme l'opérateur « join » dans SQL.
Distribute	L'opération inverse de jointure : si les enregistrements de données contiennent des attributs de plusieurs relations, résultat d'une opération de jointure précédente, alors cette opération distribue les données aux relations impliquées en conséquence.

TABLE 2.1 – *Ensemble d'opérateurs ETL*

Nous avons ajouter certains opérateurs dans le tableau 2.2 pour gérer les opérations ETL nécessaires à la construction du graphe ETL.

Opérateurs	Fonction
Extract(g, n)	Extrait, de G, le nœud n.
Filter(g, n)	Est appliqué sur les nœuds n.
Retrieve(g, n, A)	Récupère un nœud n ayant une arête étiquetée par Lj de g.
Aggregate(g , ni)	Agrège les instances représentées par les nœuds ni.
Store(G , n, I)	Associe les instances Ij aux nœuds $n \in G$ ajoutés au graphe cible G' .
Split	Divise la valeur de l'attribut a pour définir les valeurs des propriétés P1,. . ., Pn.
AddNode(GT , nj,Ej)	Ajoute une arête Ej et un nœud nj à GT.

TABLE 2.2 – *Ensemble d'opérateurs pour le graphe ETL*

Processus ETL logique

La phase de conception logique nécessite la traduction du schéma de l'entrepôt de données en modèle logique (exemple de modèle relationnel) et la traduction du processus ETL en flux logique. La traduction du processus ETL en workflow logique consiste à traduire les opérateurs ETL conceptuels en SPARQL requêtes. Chaque résultat de requête est représenté via un graphe RDF. Le processus ETL est défini par le lien de tous ces graphes. L'exemple suivant traduit l'opérateur Filter en un graphe RDF :

Select ?instance ?P where { GRAPH :?G {?Instance rdf :type name-space :Class.name-space :P ?P . FILTER (?P op value_condition)}}

2.10 Systèmes de gestion de données Multimodèles

Il existe une approche différente qui prend en charge de nombreux modèles de données utilisant plusieurs moteurs de base de données [52], tels que (i) le système fédéré [53], (ii) le système Polyglot, et (iii) le système Multistore ou polystore. Récemment, les systèmes polystore ont été proposés pour fournir un accès intégré à plusieurs magasins de données hétérogènes via un seul moteur de requête. En particulier, une grande attention est accordée à l'intégration de données non structurées, généralement stockées sur le système de fichier distribué (HDFS), avec des données relationnelles.

Les auteurs de [54], ont divisé les systèmes multimagasins en fonction du niveau de cou-

plage avec les magasins de données sous-jacents : faiblement couplé, étroitement couplé et hybride.

Systèmes multimagasins à couplage lâche

Les systèmes multimagasins à couplage lâche sont similaires aux systèmes multibases de données car ils sont capables de gérer des magasins de données indépendants. Ces magasins de données peuvent être accessibles à la fois via l'interface commune du système multimagasin et individuellement via leur propre API locale. Ces systèmes adoptent l'architecture médiateur-wrapper, qui implique l'utilisation de plusieurs magasins de données (tels que NoSQL et RDBMS), comme illustré dans la figure. 2.28 [54].

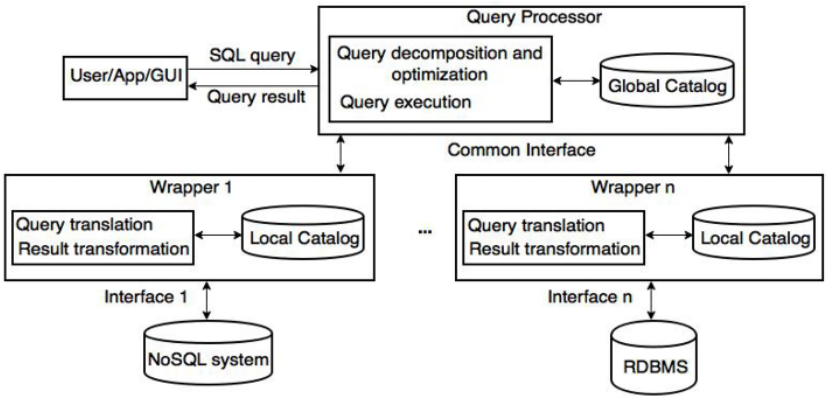

FIGURE 2.28 – *Systèmes multimagasins à couplage lâche.*

Systèmes multimagasins étroitement couplés

Les systèmes multimagasins étroitement couplés figure 2.29 [54], sont conçus dans le but d'interroger de manière efficace les données structurées et non structurées afin de les analyser à grande échelle. Ils peuvent également avoir des objectifs spécifiques tels que l'auto-ajustement ou l'intégration des données HDFS et RDBMS. Cependant, ces systèmes sacrifient leur autonomie en échange de performances accrues, généralement en utilisant un cluster dédié, ce qui signifie que les magasins de données ne sont accessibles que par le biais du système multimagasin via leur API locale. Comme les systèmes faiblement couplés, ils offrent un langage unique pour interroger à la fois les données structurées et non structurées.

Quelques systèmes polystore

Dans ce qui suit, nous passons brièvement en revue certains des systèmes polystore représentatifs existant dans la littérature les systèmes polystore fournissent un accès intégré

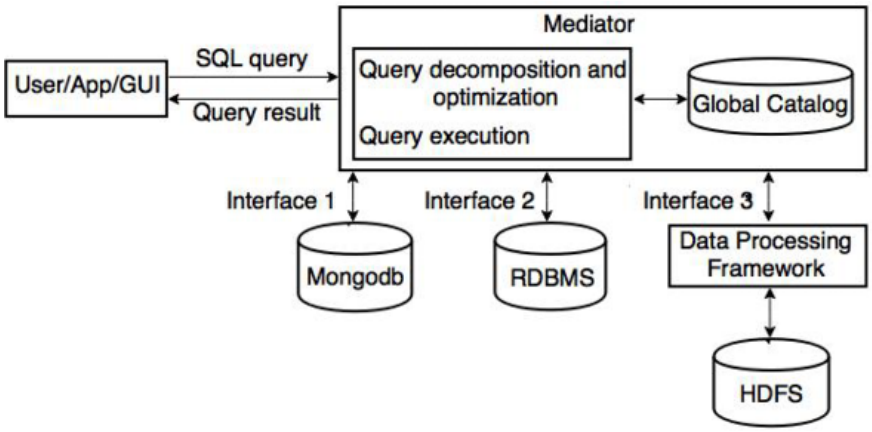

FIGURE 2.29 – *Systèmes multimagasins étroitement couplés.*

aux magasins de données tels que RDBMS, NoSQL ou HDFS et leur lien avec l'opération de chargement d'ETL :

Spark SQL [52] est un polystore hybride avec un couplage étroit de Spark/HDFS et un couplage lâche de sources de données externes. Spark SQL, qui s'exécute comme une bibliothèque au-dessus de Spark. Le processeur de requêtes accède directement au moteur Spark via l'interface Spark Scala, tandis qu'il accède à des sources de données externes (par exemple, un SGBD relationnel ou un magasin Neo4j) via l'interface Spark SQL commune prise en charge par les wrappers (JDBC, connecteur Neo4j).

Un autre système multi-magasins pour entrepôts de données appelé QoX [55], il intègre des données issues de bases de données relationnelles, MapReduce ou des outils ETL. Il utilise des wrappers appropriés pour traduire xLM en un format XML spécifique à l'outil et vice versa, QoX Optimizer peut se connecter à des moteurs ETL externes et importer ou exporter des flux de données.

Un système polystore nommé BigDAWG [56] qui vise à normaliser l'interrogation dans de nombreux formats de données et langages. Un îlot d'informations, qui est un groupe de référentiels de données accessibles à l'aide d'un seul langage de requête, est une abstraction utilisateur cruciale dans BigDAWG. Il existe de nombreux types différents, tels que les systèmes relationnels (RDBMS), les tableaux, les systèmes sans SQL et les systèmes de gestion de flux de données (DSMS).

La pile BigDAWG récemment publiée inclut Myria [57]. Myria masque les variations de modèles de données entre les backends fédérés, contrairement à BigDAWG. Il est accessible en tant que service cloud et prend en charge l'analyse fédérée.

Nous résumons dans le tableau 3.1, 3.5, une comparaison des systèmes multi-magasins cités ci-dessus.

Référcence	Muti-store system	Modèle de données	Magasins de données
[52]	Spark SQL	modèle imbriqué	HDFS, RDBMS
[55]	QoX	Graphe	RDBMS/ETL Tools/MapReduce
[56]	BigDAWG	pas de modèle typique	RDBMS/Array, NoSQL/DSMSs
[57]	Myria	pas de modèle typique	HDFS, RDBMS

TABLE 2.3 – *Comparaison des systèmes multi-magasins*

Référcence	Objectif du travail	Langages de requête utilisés
[52]	Interroger	SQL-like
[55]	performance	XML
[56]	Unificateur	opérateurs CAST et SCOPE
[57]	Interroger	MyriaL,relational,Python

TABLE 2.4 – *Comparaison des systèmes multi-magasins*

Tous les travaux mentionnés ci-dessus visent à proposer un système capable de gérer différents systèmes de gestion de données tels que RDBMS, NoSQL, XML, HDFS, etc. Cela permet de traiter des sources de données variées et de résoudre le problème d'accès à des données hétérogènes. Cependant, les systèmes polystore existants offrent un accès intégré à plusieurs magasins de données via un ou plusieurs langages de requête, avec un modèle de données commun généralement basé sur la logique relationnelle. Néanmoins, plusieurs systèmes multi-magasins sont en cours de développement pour des cas d'utilisation spécifiques, principalement en raison du niveau d'abstraction requis par la conception du schéma global, qu'il soit logique ou physique. De plus, ces systèmes sont définis avec des objectifs différents tels que l'unification des schémas, l'interrogation, les performances, etc., ainsi que des approches de traitement de requêtes, des architectures, etc. Cette diversité rend difficile le choix du système le plus adapté pour le déploiement dans un environnement spécifique. Il n'existe toujours pas de solution générique permettant de gérer de manière unifiée les différents systèmes de stockage de données, en harmonisant les langages et la variété des sources de données.

Conclusion du chapitre

Dans cette section, nous avons fournis les principaux concepts et fondamentaux qui sont nécessaires pour comprendre et construire l'état actuel des systèmes ETL et utilisés dans nos contributions et études. Aussi nous avons présenté et discuté les travaux cités dans la littérature pour tous les concepts.

Etat de l'art

3

Sommaire

3.1 Introduction

Les entreprises comme les organisations investissent dans des projets d'entrepôts de données afin d'améliorer leur activité et de mesurer leur performance. Ils visent à améliorer le processus de prise de décision en fournissant un accès unique à plusieurs sources de données. Notons que les sources de données sont autonomes ou semi autonomes. C'est la couche d'intégration dans l'environnement d'entrepôt de données [58] qui est responsable de préparer les données pour le processus de prise de décision. Les outils ETL extraient des données de plusieurs sources (tables de bases de données, fichiers plats, ERP, Internet, etc.), leur appliquent des transformations complexes suivant le schémas cible puis effectuent le chargement vers un entrepôt de données. ETL est un composant essentiel dans l'environnement d'entrepôt de données. En effet, il est largement reconnu que la construction d'un processus ETL, lors d'un projet d'entrepôt de données, est coûteuse en temps et en argent. Les couches sources de l'entrepôt de données englobent tous les types de sources de données, Ils sont fournisseurs de données. En raison de son rôle essentiel dans l'intégration des données, le processus ETL fait l'objet d'une attention croissante dans la recherche et la pratique, mais moins d'études ont synthétisé les études existantes sur l'ETL pour définir une méthode d'évaluation ou une orientation de recherche future qui soit solide et durable.

Dans ce chapitre, nous proposons d'étudier les différents travaux de recherches proposés sur l'ETL de l'entrepôt de données. Nous présenterons une synthèse des travaux retenus, leurs principales contributions avec une classification de ces travaux selon un ensemble de critères retenus. Ces critères sont choisis pour leur pertinence par rapport à nos contributions. Nous ressortons les principales difficultés et limites liées à ces travaux, et nous mettons en avant les problèmes sur lesquels nous allons tenter d'apporter des solutions. Les travaux que nous avons retenus dans l'état de l'art sont les travaux les plus référencés dans la littérature, et aussi les travaux récents proposés ces dernières années. Ces travaux sont présentés par ordre d'apparition. D'autres travaux moins référencés sont uniquement cités dans la classification que nous fournissons pour analyser les propositions de l'état de l'art.

3.2 État de l'Art sur les processus Extract Tranform Load

Nous présentons et nous passons en revue dans cette section les principaux travaux proposant des méthodes de conception, de développement ou d'optimisation des processus ETL. Ainsi, plusieurs études ont été proposées pour améliorer l'efficacité, op-

timiser et mettre à l'échelle les processus ETL. Les articles sélectionnés ont été collectés auprès des sources suivantes : (i) IEEEXplore (https ://ieeexplore.ieee.org/) ; (ii) ACM Digital Library (https ://dl.acm.org/) ; (iii) Scopus, Web of Science, SpringerLink (https ://www.springerlink.com) ; (iv) ScienceDirect (https ://www.sciencedirect.com/) ; (v) Google Scholar (https ://scholar.google.com/) et (vi) DBLP(https ://dblp.uni-trier.de).

Dans [12], les auteurs ont proposé un modèle conceptuel pour le processus ETL utilisant les technologies du Web sémantique. La méthode définit la sémantique de la source de données à l'aide d'une ontologie. La méthode proposée est semi-automatique et ne prend pas en compte de scénarios réels. Les auteurs de [59], introduisent des règles de transformation de graphe personnalisables et extensibles pour construire le processus ETL de manière incrémentielle.

Dans le travail de [60], les auteurs suggèrent un processu ETL qui utilise un thésaurus pour simplifier le remplissage de l'entrepôt de données. Le thésaurus qui détaille les structures des sources est établi et utilisé pour créer des correspondances entre les schémas des sources et celui de l'entrepôt. Ces correspondances (mappings) reposent sur des critères de ressemblance sémantique. Une fonction transformationnelle est mise en place pour résoudre les divergences entre les sources et nourrir le schéma de l'entrepôt. Ce travail se déroule en trois phases voir la figure 3.1 : (i) extraction de la description du schéma des sources à l'aide d'un wrapper pour les différents types de sources de données (RDF, XML, etc.) ; (ii) annotation utilisant WordNet [61] qui est le processus d'association du concept le plus approprié d'une ontologie de référence ; et (iii) création d'un thésaurus des relations entre les éléments du schéma de l'entrepôt de données.

Dans tous les travaux précédents, le processus d'extraction commence par relier des parties du schéma des sources de données aux concepts et propriétés du modèle de données.

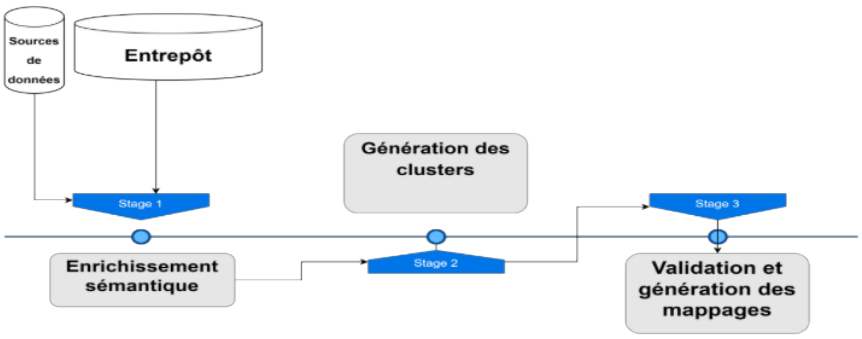

FIGURE 3.1 – *Architecture fonctionnelle de l'outil d'extraction sémantique.*

Dans le travail de [62], les auteurs ont traité un problème important qui se pose dans

les premières étapes d'un projet d'entrepôt de données : le problème de l'identification des transformations appropriées et de la spécification des mappings inter schémas qui doivent conduire le flux de données des sources de données vers l'entrepôt de données cible. Dans cette approche, les technologies sémantiques sont utilisées pour améliorer d'avantage les définitions des activités ETL impliquées dans le processus plutôt que les données elles-mêmes. L'études est axées sur les étapes de transformation et de chargement, l'utilisation d'ontologies et du langage OWL. L'hétérogénéité est résolue à l'aide des technologies du web sémantique. Les magasins de données sont sémantiquement annotés par une ontologie appropriée décrite dans OWL-DL.

Dans le travail de [63], les auteurs ont considéré l'optimisation logique des processus ETL, en les modélisant comme un problème de recherche dans l'espace d'états. Chaque ETL workflow est vu comme un état, donc l'espace d'état est construit à travers un ensemble de transitions d'état correctes. Des algorithmes implémentés en C++ pour minimiser le coût d'exécution des processus ETL sont présentés. Les résultats expérimentaux sur ces algorithmes suggèrent que les bénéfices de la méthode proposée peuvent être significatifs.

L'objectif principale du travail de [64] est de fournir aux concepteurs un moyen semi-automatique de produire des représentations multi-dimensionnelles (MD) conceptuelles ainsi qu'une représentation conceptuelle des processus d'extraction-transformation-chargement (ETL). Les auteurs ont proposé le framework GEM (Requirement-Driven Generation of ETL and Multidimensional Conceptual Designs). Un système qui facilite la génération semi-automatique des concepts ETL et MD, à partir d'un ensemble d'exigences métier et de sources de données. Ils avaient décrit comment les exigences peuvent être validées et enrichies, afin de produire une ontologie annotée contenant des informations correctes à la fois pour les sources et les exigences, ensuite, ils ont montré comment utiliser cette ontologie pour produire les concepts MD et ETL. La méthode est validée en termes du benchmark TPC-DS [65]. Dans ce papier, les auteurs démontrent comment les données sémantiques peuvent être analysées dynamiquement à l'aide d'agrégations, de navigation et de rapports de type OLAP. La diversité et l'hétérogénéité des sources d'information rendent le processus de construction de l'entrepôt de données complexe et long, ce processus la plupart du temps effectué manuellement. Les auteurs de [66], proposent un framework sémantique d'extraction-transformation-charge (ETL) qui utilise des technologies sémantiques pour intégrer et publier des données provenant de plusieurs sources en tant que données liées ouvertes (LOD). Dans cette approache, les processus ETL sont définis au niveau ontologique pour masquer les détails de l'implémentation, les auteurs proposent la modélisation conceptuelle des processus ETL avec la notation

BPMN, ils considèrent un fragment de BPMN comme élément central pour décrire ces processus. Le fragment de méta-modèle BPMN décrit par les auteurs est composé des méta-classes suivantes : BaseElement, Flow objects, Activity, Task, Event, DataInput, DataOutput, InputSet, OutputSet, Gateway, Inclusive, Exclusif, Parallèle, Complexe. En plus les auteurs ont étendu ce fragment BPMN avec les opérations ETL de base : Extract, Retrieve, Merge, Union, Join, Filter, Conversion, Aggregation et Store. Pour valider leurs travail les auteurs ont considéré le scénario suivant : Construire et déployer un entrepôt de données sémantique à partir de six bases de données sémantiques (SDB). Les six Oracle SDBs alimentées à l'aide de l'ontologie Lehigh University Benchmark (LUBM). Les instances N-triples sont chargées dans six sources SDB Oracle et l'intégration dans l'entrepôt est réalisée au moyen d'un processus ETL. Le processus général d'intégration est mis en œuvre sous forme de services, ETL as a Service (ETLaaS) et Physical storage as a Service (PDaaS). Les auteurs considèrent deux critères pour mesurer la faisabilité de leur approche : la complexité de l'algorithme ETL proposé et l'évolutivité en termes d'instances des sources de données.

Pour intégrer des données provenant de sources hétérogènes et publier ces données sous forme de LOD, dans le travail de [67] un framework ETL sémantique est présenté au niveau conceptuel. L'approche utilise la technologie sémantique pour faciliter le processus d'intégration et discute de l'utilisation de différents outils pour accomplir les différentes étapes du processus ETL. Seule la phase transformation du processus ETL est considérée pour créer un modèle de données sémantiques et générer des données liées sémantiques (triples RDF) à stocker dans un entrepôt de données. Les phases d'extraction et de chargement du processus ETL resteraient les mêmes. La phase de transformation continuera toujours à effectuer d'autres activités telles que la normalisation et le nettoyage des données. Le modèle de données sémantique est créé lors de la phase de transformation qui génère un fichier RDF qui va alimenter l'entrepôt de données. Sur la base des résultats, le schéma des sources de données devra être mapper sur une ontologie existante spécifique au domaine ou une ontologie devra être créée à partir de zéro. Si les sources de données appartiennent à des domaines disparates, plusieurs ontologies seront nécessaires et des règles d'alignement devront être spécifiées pour tout champ de données commun ou associé. Pour implémenter l'ETL sémantique, les auteurs utilisent deux datasets (National Household travel survey data et EPA's Fuel Economy data). Un modèle de données est créé composer des classes primaires Personne, Ménage, Véhicule au plus haut niveau. Tous les champs des datasets ont été modélisés en tant que classes, sous-classes ou propriétés et connectés aux classes primaires via des relations appropriées. L'ontologie avait à la fois des propriétés d'objet et des propriétés de données en fonction du

type de champ de données représenté. La figure 3.2 montre un aperçu général des activités dans l'ETL sémantique..

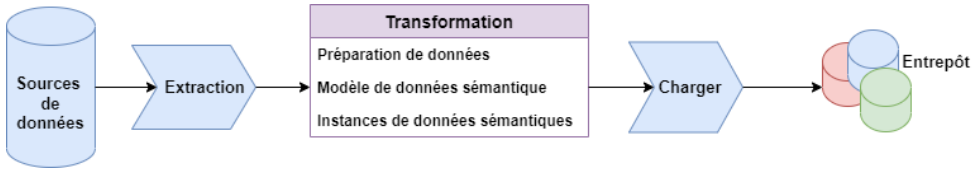

FIGURE 3.2 – *Vue d'ensemble des activités en ETL sémantique.*

Les données intégrées ont été testées à l'aide du langage de requête RDF SPARQL. Une application a été créée qui charge les ontologies et les données, vérifie la conformité des données avec le vocabulaire, puis exécute des requêtes SPARQL sur les données. L'un des défis de ce projet est l'ingénierie des ontologies qui nécessite une assez bonne compréhension des données provenant de différentes sources. Un expert humain doit effectuer cette étape et c'est souvent un processus qui prend du temps.

Les auteurs de [68], ont pris en compte la variété des sources de données et la variété des plates-formes de stockage des entrepôts de données. Ils ont proposé l'unification de l'environnement ETL pour gérer la variété, pour cela, ils ont utilisé les techniques d'ingénierie dirigée par les modèles, un méta-modèle pour les sources de données, un méta-modèle pour les opérateurs ETL, et par conséquent un méta-modèle pour le processus ETL. Pour le méta-modèle des sources de données, l'instanciation de trois types de bases de données (relationnel, sémantique et graphe) est proposée, et chaque opérateur ETL générique est implémenté en tant que service web en utilisant l'implémentation du polymorphisme de surcharge du langage Java. À partir de leurs expériences, ils démontrent que la surcharge des opérateurs ETL peut gérer l'inférence en moins de temps.

Le travail de [28], aborde le problème de la mise à jour des dimensions à évolution lente (SCD : slowly changing dimension) avec dépendances, c'est-à-dire le cas lorsque la mise à jour d'une table SCD a un impact sur les tables SCD associées. Une approche de développement des processus ETL est proposée, qui commence par un modèle conceptuel BPMN pour l'ETL traduit en RA (algèbre relationnel) étendu avec des opérations de mise à jour au niveau logique. Trois implémentations sont discutées : Le benchmark TPC-DI [69] est utilisé comme exemple d'exécution :

— Spécifications BPMN d'un processus ETL -> Vers Pentaho data integration tool [32].
— Spécifications BPMN d'un processus ETL ->Vers Talend data integration tool [70].
— Spécification RA (relational algebra) étendu par des opérations de mise à jour -> Vers SQL exécuté sur SGBD.

Les résultats suggèrent que l'approche RA étendue aboutit à des processus ETL plus efficaces que ceux produits en implémentant la spécification BPMN4ETL sur les outils ETL open sources mentionnés.

Intégrer des ensembles de données RDF hétérogènes et indépendants, mais qui contiennent des caractéristiques communes qui permettent leur intégration. L'objectif du projet dans le travail [71] est de faciliter à l'utilisateur l'interrogation d'ensembles de données RDF similaires d'une manière intégrée et multidimensionnelle (appelé Jeu de données (IMD)). IMD est annoté comme un cube avec le vocabulaire QB4OLAP [72], qui active les fonctionnalités OLAP et résout les problèmes de compatibilité tels que les différents noms de ressources ou les détails de granularité. QB4OLAP est un vocabulaire pour la business intelligence sur les données liées. C'est une extension du vocabulaire DataCube qui permet de représenter les cubes OLAP dans RDF, et d'implémenter les opérateurs OLAP (tels que Roll-up, Slice et Dice) comme requêtes SPARQL directement sur la représentation RDF. Dans ce papier, les auteurs proposent une nouvelle méthode pour combiner semi-automatiquement des ensembles de données RDF dans le même espace multidimensionnel. Une intégration basée sur une interprétation multidimensionnelle d'ensembles de données arbitraires.

Dans le papier [9], les auteurs proposent une architecture générique qui satisfait aux exigences d'un système de d'ingestion de données en streaming. Les principales exigences fonctionnelles auxquelles une telle approche ETL streaming devrait répondre tels que le stockage local intégré et les garanties transactionnelles ACID (Atomicité, Cohérence, Isolation et Durabilité). Le streaming ETL est le traitement et le déplacement de données en temps réel d'un endroit à un autre. Pour répondre aux besoins des applications big data dans lesquelles la latence est importante, de nouvelles exigences sont imposées au processus ETL.

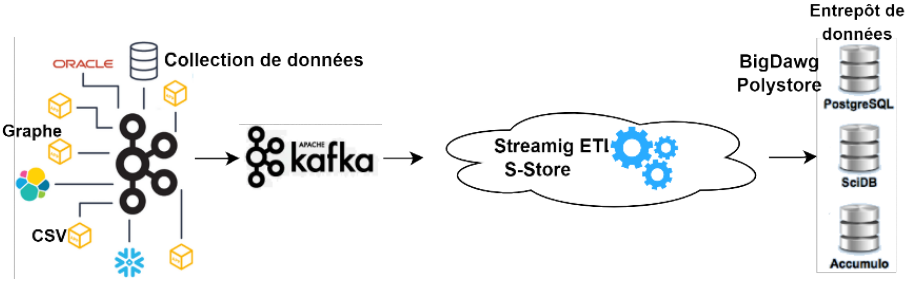

FIGURE 3.3 – *Architecture ETL en streaming.*

Les auteurs implémentent la nouvelle architecture de flux ETL en streaming (figure 3.3 pour une utilisation dans un contexte ETL relationnel, ils ont considéré trois technologies pour implémenter l'ETL streaming : Apache Kafka [73], S-Store [74], et Intel's BigDAWG polystore

[56]. L'une des questions clés auxquelles est confronté un système ETL en streaming est la fréquence à laquelle les données doivent être migrées vers l'entrepôt de données. Il existe deux méthodes pour déterminer le moment où cette migration se produit : soit le moteur d'ingestion envoie (PUSH) périodiquement les données à l'entrepôt, soit l'entrepôt extrait (PULL) les données du moteur d'ingestion lorsque cela est nécessaire.

Les auteurs dans [75], ont présenté une enquête sur les processus ETL en temps réel utilisés dans les systèmes d'entreposage de données. Contrairement à d'autres enquêtes, les auteurs de ce papier ont discuté d'une liste de travaux connexes qui fournissent une recherche générale.

De plus les auteurs de [37], ont proposé un framework basé sur le cloud pour l'ETL dans le contexte du big data. Trois scénarios d'agrégation de données pendant l'ETL ont été envisagés, tous avec un seul type de source de données CSV ou JSON et l'outil Infobright DB [76] comme entrepôt de données. Les données utilisées ont été recueillies à partir de divers dispositifs (c.-à-d. téléphones intelligents, sites Web et applications de bureau).

Alors que les auteurs de [4], ont présenté l'état de l'art dans l'ensemble du processus de développement et d'optimisation d'un workflow ETL. Les recherches récentes ont eu tendance à se concentrer moins sur la modélisation conceptuelle du processus ETL et plus sur leurs mise en œuvre des technologies de big data dans l'écosystème Hadoop, telles que Spark, Hive, Pig et Storm, entre autres.

Les entreprises utilisent des pools d'informations plus larges (par exemple, les médias sociaux, les données de capteurs, les documents...etc.) pour permettre une meilleure prise de décision, les technologies traditionnelles d'entrepôts de données et OLAP doivent être adaptées et étendues de manière appropriée.

Dans le travail [77], l'objectif était d'étudier comment les technologies du Web sémantique peuvent aider à la découverte de données, à l'acquisition, à l'intégration et à l'interrogation analytique de données externes, et ainsi servir de base au traitement analytique en ligne exploratoire (OLAP). Les auteurs ont analysé les étapes à suivre pour créer des entrepôts de données et répondre aux requêtes OLAP. Cette étude révèle que les technologies du web sémantique et OLAP peuvent et doivent être considérées ensemble.

Dans les cadre d'intégrer de grandes quantités de données RDF provenant de plusieurs sources, [78], proposent RDFINT, un benchmark pour comparer les deux approches d'intégration de données RDF (Traditionnelle et Virtuelle).

Une approche d'entrepôt de données traditionnel où les données RDF pertinentes sont extraites de différentes sources puis intégrées dans un entrepôt de données.

Une approche d'intégration virtuelle où les données RDF réside toujours à chaque source et l'intégration des données se produit lorsque les données sont interrogées, via un médiateur qui se coordonne avec les wrappers à chaque source.

Des mesures sont considérées pour quantifier les différences entre les différentes approches d'intégration des données RDF. Les auteurs ont réalisé des expériences sur trois ensembles de données générés : petit (environ 100 000 à 150 000 triples par source de données), moyen (environ 1,5 million de triples par source de données) et grand (environ 20 millions de triples par source de données). Pour chaque expérience, varier l'un des facteurs et maintenir les autres aux valeurs par défaut. Les résultats expérimentaux de l'évaluation de l'entrepôt de données et des approches d'intégration virtuelle sont rapportés : Les résultats ont montré que l'intégration virtuelle paye une certaine surcharge au moment de l'exécution, mais présente l'avantage de l'exhaustivité des résultats. La plupart des opérations d'intégration de données évoluent également de manière linéaire avec la taille des données et le nombre de sources.

Afin de réduire le coût de développement et le temps d'exécution des processus ETL, des solutions logiciels sont à proposés. Dans leur travail [79], les auteurs ont introduit le framework pygrametl pour le développement de packages ETL en langage python, les développeurs pourraient traiter efficacement les données en utilisant des fonctionnalités intégrées et utilisant l'accès aux données pour les tables de faits et de dimensions. Cette nouvelle approche a été introduite dans le but de réduire à la fois le temps d'exécution et de développement.

Recement, les auteurs de [35], ont proposé une approche formelle pour l'intégration en temps réel des données IoT en streaming. Les auteurs abordent la question des conflits de synchronisation et d'alignement entre les données en continu provenant de plusieurs sources.

Dans le papier [23], les auteurs fournissent un aperçu des études pertinentes concernant la modélisation des méthodes d'entreposage de données, englobant les processus ETL traditionnels ainsi que les approches de conception ELT contemporaines. Ce travail identifie les principaux obstacles et préoccupations associés au développement de systèmes d'entreposage de big data. Certains travaux ont considéré les ontologies comme des ressources destinées à faciliter et automatiser la conception du processus ETL.

Les auteurs de[80], proposent une architecture ETL en temps réel pour le big data non structuré pendant la phase de transformation, en se concentrant sur l'accélération des jointures de flux sur disque et l'atténuation de la surcharge du disque et de la perte de données.

Enfin afin d'améliorer les performances du processus ETL à l'ère du big data et en prétraitant les données en amont. Cependant, le auteurs de [81], ont proposé un framework ETL pour l'entreposage de données utilisant et tirant parti des technologies de bases de données NoSQL. Les auteurs de [10], ont proposé des lignes directrices pour la conception multidimensionnelle multimodèles dans des entrepôts de données basés sur multi-model DBMS (MMDBMS), validées par des comparaisons intra-modèle et intermodales, et illustrées par une étude de cas.

La plupart des organisations préparent actuellement de passer à la mise en œuvre au cloud computing de leurs applications essentielles. Dans le papier de [6], une solution ETL basée sur le cloud est présentée et un nouveau framework ETL basé sur Spark a été développé. Ce framework gère le flux de données en temps réel sur la plate-forme cloud. Le travail couvre les caractéristiques générales demandés par les outils ETL actuels :

— La possibilité d'extraire et de collecter des données de tous types de sources de bases de données (RDBMS, NoSQL), de fichiers plats, de technologies big data (Hadoop, Spark), et de données générées par des capteurs... etc.,
— Interface graphique simple pour faciliter l'utilisation,
— Plus de capacités de transformation de base,
— Intégration de métadonnées de gestion des fonctionnalités et de documentation pour les différentes règles de transformations et d'analyse,
— Ordonnancement efficace des tâches et contrôle de la gestion des erreurs, des alertes et de la journalisation,
— La caractéristique la plus recherchée est d'intégrer des données à la fois sur le site et sur le cloud.

La figure 3.4 [6], présente l'architecture de base du framework proposé.

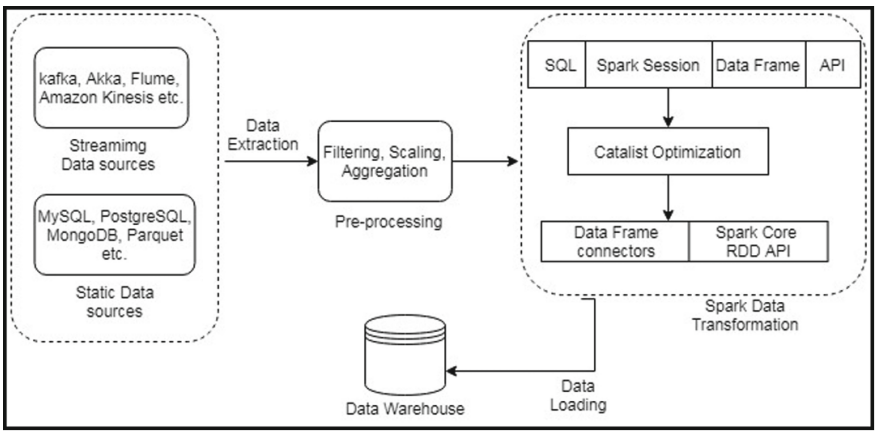

FIGURE 3.4 – *Le framework Streaming ETL basé sur Spark.*

Les entreprises produisent une énorme quantité de données en temps réel grâce aux nou-

velles applications. Ces données doivent être collectés, transformés et analysés en temps réel pour prendre des décisions critiques. Dans le travail de [82], les auteurs ont présenté les principales fonctionnalités du moteur ETL de Striim. Striim est une nouvelle plate-forme ETL de streaming distribuée de bout en bout qui permet un développement et un déploiement rapides d'applications de streaming. C'est un moteur ETL en temps réel conçu pour permettre de créer et de déployer des applications de streaming.

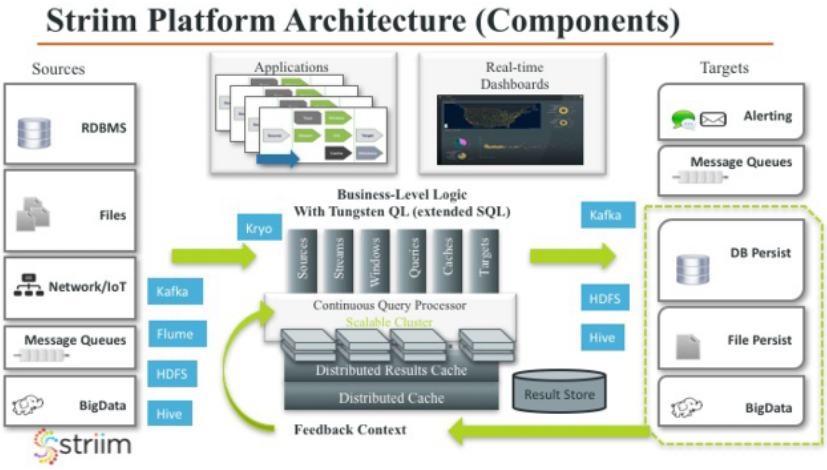

FIGURE 3.5 – *Les composants du framework Streaming ETL basé sur Spark.*

Le schéma fonctionnel de la plate-forme Striim est illustré sur la figure 3.5 [82]. Le moteur ETL s'exécute dans un cluster de noeuds de calcul évolutif et tolérant aux pannes, extrait les données en temps réel des sources, les transforme, produit des événements de résultat qui sont chargés/diffusés dans les cibles. Les auteurs ont démontré l'efficacité de Striim par rapport à un moteur ETL de streaming open source populaire KSQL [83] construit sur Apache Kafka.

L'intégration des données massives (big data) se heurte à de nombreux problèmes, car il est difficile de traiter ces informations à l'aide de bases de données et de méthodes logicielles traditionnelles. Les auteurs de [84], présentent une étude comparative des méthodes d'intégration des big data avec le Web sémantique, et met en évidence certains des défis rencontrés lors du processus d'intégration. Les auteurs discutent du rôle du Web sémantique pour résoudre ces problèmes en utilisant l'intégration de données sémantiques.

La grande quantité de données rend ETL extrêmement coûteux. L'utilisation de technologies de parallélisation est la clé pour atteindre une meilleure évolutivité et performance ETL. Dans leur travail [85], les auteurs ont présenté le framework ETLMR qui utilise MapReduce pour atteindre l'évolutivité. L'utilisateur peut implémenter des programmes ETL parallèles (seules quelques lignes de code déclarant les tables cibles et les fonctions de transformation)

en utilisant des constructions sur les tables de faits et les dimensions y compris les SCD proposé par le framework sans connaître les détails de l'exécution parallèle des processus ETL. Sur la figure 3.6, est décrit le framework ETLMR. Le flux ETL se compose de deux phases séquentielles : le traitement des dimensions et le traitement des faits. Les données sont lues à partir des sources, des fichiers sur un système de fichiers distribué hdfs, transformées et traitées en valeurs de dimensions et en faits par des instances ETLMR parallèles qui consolident les données dans l'entrepôt de données.

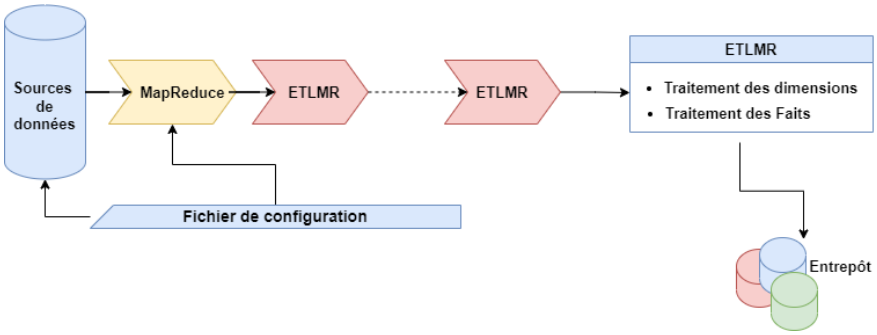

FIGURE 3.6 – *Le Framework ETL basé sur MapReduce.*

Le processus ETL peut être configuré avec seulement quelques lignes de code. Avec l'utilisation de MapReduce les données seront stockées dans Hdfs sur Hadoop, ce qui n'est pas aussi bon avec l'utililisation de Spark qui stocke des données en mémoire.

Dans le cadre des travaux sur le parallélisme des flux ETL, les auteurs [86], ont proposé un framework (Figure 3.7) d'optimisation utilisant le partitionnement et la parallélisation des flus ETL dans le but de minimiser le temps et les ressources nécessaires aux flux de données ETL. Le framework proposé se base sur les caractéristiques des composants ETL pour faire le partitionnement et les optimisations. Les auteurs ont proposé l'utilisation de caches partagés pour le transfert de données, qui peuvent réduire l'utilisation des E/S.

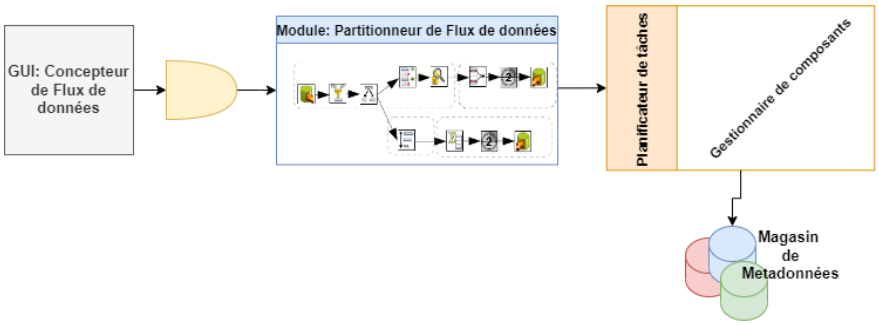

FIGURE 3.7 – *Framework de partitionnement de flux de données ETL.*

Pour résoudre ces problèmes les auteurs proposent un partitionnement vertical sur les flux de données, puis en utilisant un seul cache partagé pour transférer les données entre

les composants au sein d'une même partition. Ensuite, le pipeline et le multi-threading sont appliqués aux partitions. Le framework est implémenté sur l'outil open source ETL, Talend Open Studio pour l'intégration de données [70]. De plus, XML est utilisé comme référentiel de métadonnées (le SGBDR peut également être configuré pour être utilisé) et le gestionnaire de référentiel est étendu pour prendre en charge la gestion des informations de partitionnement d'un flux de données. D'après les auteurs le framework proposé est 4,7 fois plus rapide que les processus ETL sans utiliser les techniques d'optimisation proposées.

Dans le travail [63], l'objectif est la réduction du coût d'exécution soit en diminuant le nombre total d'activités, soit en modifiant l'ordre des activités dans un workflow ETL. Un problème de recherche dans l'espace d'états est défini, où chaque état dans un espace de recherche est un graphe acyclique orienté (DAG). Dans [86], les auteurs ont proposé le partitionnement et la parallélisation des workflows ETL comme méthode d'optimisation. Cette classe est illustrée à la figure 3.12

Dans [87], Stonebraker souligne également que les systèmes de style MapReduce excellent dans les analyses complexes et les tâches ETL. La grande quantité de données rend l'ETL très coûteux. Suite à l'émergence du big data, certaines études se sont penchées sur des questions intéressantes. Par exemple, les auteurs dans [85], ont introduit une approche centrée sur les performances des processus ETL traitant du Big Data selon le paradigme MapReduce. Cette approche a été implémentée dans un prototype appelé ETLMR, décrit à des fins de démonstration.

Dans le travail de [6], l'objectif principal était de surmonter les limites du framework MapReduce, un framework de calcul parallèle en mémoire appelé Spark utilise le calcul multi-passes pour développer ETL, c'est-à-dire le calcul de composants plusieurs fois, en utilisant le modèle Direct Acyclic Graph (DAG) . Afin d'accélérer le traitement en temps réel des données, différentes méthodologies ont été employées au fil du temps. La création de vues matérialisées est l'une des méthodes établies dans les applications centrées sur les données pour accélérer la vitesse de traitement des requêtes. Les entreprises produisent d'énormes quantités de données en temps réel grâce à de nouvelles applications. Une nouvelle architecture ETL distribuée nommée Striim est développée dans [82] pour prendre en charge les transformations de données en temps réel sur le flux de données.

3.3 Analyse et comparaison des travaux sélectionnées

En conclusion, dans cette enquête que nous avons mené dans le domaine de l'ETL a mis en évidence plusieurs facteurs importants qui affectent le succès des processus ETL. Il s'agit notamment des défis d'intégration de données hétérogènes, des problèmes de performances, d'optimisation et d'évolutivité. Pour surmonter ces défis, les organisations doivent adopter les meilleures pratiques dans la conception et la mise en œuvre des processus ETL. Dans l'ensemble, les résultats de l'enquête suggèrent que les processus ETL continuent d'être essentiels pour les organisations dans la gestion et l'exploitation efficaces des données, il est nécessaire d'innover et d'améliorer continuellement dans ce domaine pour suivre le rythme de l'évolution du paysage des données.

Nos recherches ont révélé que plusieurs approches sont proposées pour résoudre le problème des conflits sémantiques et structurels. De plus, de nombreuses études se sont concentrées sur la modélisation et l'optimisation du processus ETL. Le monde universitaire a développé plusieurs outils ETL open source populaires basés sur du code, tels que Petl et Pygrametl [79]. Cependant, compte tenu de l'expansion rapide des données d'entreprise, une quantité considérable de recherches se concentre actuellement sur l'intégration du big data dans les processus ETL, où les tâches d'extraction et de transformation sont effectuées par Spark ou MapReduce.

D'autres travaux exploitent les avantages du cloud computing. Dans cette enquête, nous avons proposé une nouvelle méthode de classification de la littérature récente dans le domaine des processus ETL selon un ensemble de critères. Nous classons ces travaux en sept classes. Sur la base de la revue de la littérature et des forces et faiblesses identifiées, on peut conclure qu'il existe de nombreuses opportunités de recherche dans les différentes classes identifiées au sein des environnements d'entreposage de données. De nombreux travaux ont présenté des idées initiales de solutions, mais peu ont entièrement résolu le problème. Nous développons actuellement une série d'approches dans cette lignée.

Les sept critères suivants ont été utilisés pour classer, évaluer et examiner les processus ETL :

— Sources de données (S.D) : ce critère indique le type des sources de données prises en charge et impliquées dans le processus ETL,
— Modélisation : Ce critère désigne le modèle utilisé dans le processus ETL. Cependant, si le processus ETL est modélisé : conceptuel : ontologie de conception (sources et entre-

pôt) : une représentation basée sur un graphe, appelé graphe de magasin de données, BPMN ou UML,

— Automaticité : Ce critère indique si l'ETL est réalisé (i) de manière manuelle nécessitant la présence d'un expert ou concepteur ; (ii) approche semi-automatique ou (iii) automatique,

— Implémentation : le critère d'implémentation détermine si les processus ETL sont implémentés avec un SGBD, via un framework, un outil, ou codés manuellement,

— Processus ETL : Ce critère indique quel effort de développement de processus ETL est utilisé,

— Déploiement : le type de déploiement ETL, le type de magasin de déploiement définit le stockage physique de l'entrepôt de données, s'il est traditionnel (un magasin) ou multi-magasin (polystore),

— Optimization : Indique s'il ya un objectif d'optmiser les processus ETL ou non dans l'approche.

Les tableaux 3.1, 3.5, 3.3 et 3.4 décrivent les comparaisons entre la majorité des travaux sélectionnées. La comparaison est donnée sur la base des critères cités ci-dessus.

Référcence	Modélisation	Automaticité
[12]	/	Semi-Auto
[60]	/	Semi-Auto
[62]	Conceptuel (user)	Semi-Auto
[64]	Non	Semi-Auto
[88]	Conceptuel (user)	Semi-Auto
[66]	Conceptuel (BPMN)	Auto
[67]	Conceptuel	Auto
[68]	Conceptuel (multidimensionnel)	Semi-Auto
[89]	Conceptuel(BPMN)	Semi-Auto
[11]	Myria	Semi-Auto
[28]	Myria	Auto
[71]	Myria	Semi-Auto
[10]	Myria	Auto
[9]	Myria	Auto
[78]	Myria	Semi-Auto
[79]	Myria	Auto
[35]	Myria	Auto

TABLE 3.1 – *Comparaisons entre la différents travaux sélectionnées.*

Référcence	Implémentation
[12]	Algo.phase transformation/pas de scénario réel
[60]	Algo.phase transformation/pas de scénario réel
[62]	Algo.phase de transformation/pas de scénario réel
[64]	Framework (outil GEM)
[88]	Algo.phase extraction/avec scénario réel
[66]	Framework
[67]	Framework
[68]	Algo ETL
[89]	Algo ETL
[11]	Algo ETL
[28]	Outil
[71]	Framework
[10]	Framework
[9]	Streaming ETL
[78]	Benchmark
[79]	Framework
[35]	Outil

TABLE 3.2 – *Comparaisons entre la différents travaux sélectionnées.*

Référcence	Processus ETL
[12]	Semantique
[60]	Semantique
[62]	Semantique
[64]	Semantique
[88]	Semantique
[66]	Semantique
[67]	Semantique
[68]	Semantique
[89]	Semantique
[11]	Semantique
[28]	Outil ETL
[71]	Outil ETL
[10]	Semantique
[9]	Streaming
[78]	Codé à la main
[79]	Codé à la main
[35]	Codé à la main

TABLE 3.3 – *Comparaisons entre la différents travaux sélectionnées.*

3.4 Classification

Sur la base de la dernière comparaison et en tenant compte des critères proposés dans cette enquête, nous proposons une nouvelle classification des processus ETL en cinq classes :

3.4.1 Classe 1 :

Avec des sources de données dynamiques et de structure hétérogène, l'intégration du big data implique de relier d'énormes volumes de données hétérogènes provenant de diverses sources. Les principaux problèmes rencontrés sont le mappage de schéma, la liaison d'enregistrements et la fusion de données. La plupart des travaux sur l'ETL ont impliqué des données hétérogènes provenant de sources de données opérationnelles d'entreprise. De plus, et avec le développement rapide des nouvelles technologies de traitement de données, des travaux ont pu utiliser des données non traditionnelles comme l'IoT [78], les bases de données de structure graphes et les bases de connaissances [89], [68]. Basé sur le critère sources de données, on distingue une première classe qui est illustrée à la figure 3.8 qui résume les différents types et formats de données utilisée dans les travaux sélectionnés, le constat le plus important qui ressort de l'analyse de ces travaux est que toutes les expérimentations ont montré que les tests sur les différentes sources hétérogènes peuvent encore être améliorés.

Référcence	Dépoiement	S.D
[12]	1 magasin	Hétérogène
[60]	1 magasin	Hétérogène
[62]	1 magasin	Hétérogène
[64]	1 magasin	Hétérogène
[88]	1 magasin	Hétérogène
[66]	1 magasin	Hétérogène
[67]	1 magasin (ontologie)	Hétérogène
[68]	Polystore	Knowledge base
[89]	1 magasin (graph RDF)	Knowledge base
[11]	Polystore	Graph RDF
[28]	1 magasin	Hétérogène
[71]	1 magasin (graph RDF)	Graph RDF
[10]	1 magasin (graphe RDF)	Hétérogène
[9]	Polystore	Iot
[78]	1 magasin (graphe RDF)	Graph RDF
[79]	1 magasin	CSV
[35]	1 magasin	Données Iot

TABLE 3.4 – *Comparaisons entre la différents travaux sélectionnées.*

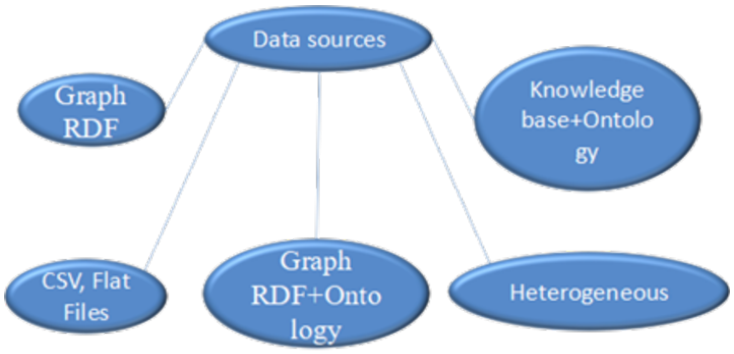

FIGURE 3.8 – *Classe 1.*

3.4.2 Class 2 :

Dans cette classe illustrée à la figure 3.9, on distingue différents types des processus ETL, le constat est le suivant : l'ETL sémantique impliquant des données sémantiques peut offrir une solution au problème d'hétérogénéité et garantir l'interopérabilité entre les applications exploitant l'environnement du big data. De plus, les solutions ETL avec une interface utilisateur graphique (GUI) offrent un moyen convivial de mapper rapidement et facilement des éléments de données entre le système source et le système cible [90]. Ces outils sont souvent comparés aux ETL basés sur du code largement utilisés dans le milieu universitaire et la recherche lorsqu'ils tentent de trouver des solutions à des problèmes d'intégration plus spécifiques. D'autre part des solutions ETL en streaming ont vu le jour dans l'environnement numérique d'aujourd'hui dans lequel les entreprises doivent accéder, stocker et analyser en

Référcence	Optimisation
[12]	Hétérogène
[60]	Hétérogène
[62]	Hétérogène
[64]	Hétérogène
[88]	Hétérogène
[66]	Hétérogène
[67]	Hétérogène
[68]	Knowledge base
[89]	Knowledge base
[11]	Graph RDF
[28]	Hétérogène
[71]	Graph RDF
[10]	Hétérogène
[9]	Iot
[78]	Graph RDF
[79]	CSV
[35]	Données Iot

TABLE 3.5 – *Comparaisons entre la différents travaux sélectionnées.*

temps réel de grandes quantités de données provenant de sources de données de structure graphe en streaming.

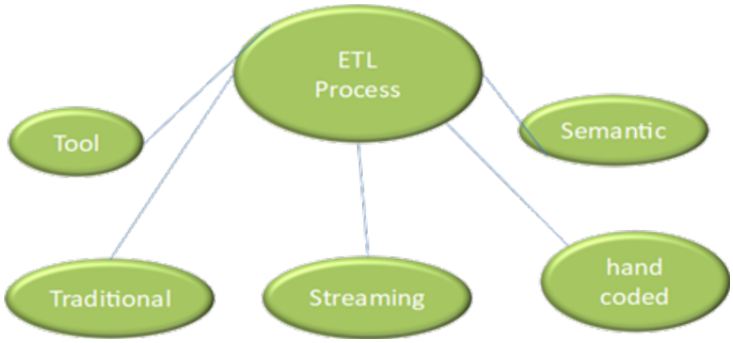

FIGURE 3.9 – *Classe 2.*

3.4.3 Classe 3 :

Dans cette classe (figure 3.10), la modélisation ETL conceptuelle vise à relier le schéma de données source au schéma d'entrepôt de données cible [91]. La modélisation donne une abstraction de haut niveau du processus ETL dans le but de masquer les détails techniques de mise en œuvre physique. Une littérature considérable a été réalisée pour la modélisation des processus ETL en utilisant des méthodes basées sur UML, MDE (Model Driven Engineering), ou BPMN pour masquer les implémentations physiques et fournir plus d'abstraction aux concepteurs. Il est évident que la modélisation conceptuelle est largement utilisée pour concevoir et mettre en œuvre des pipelines ETL.

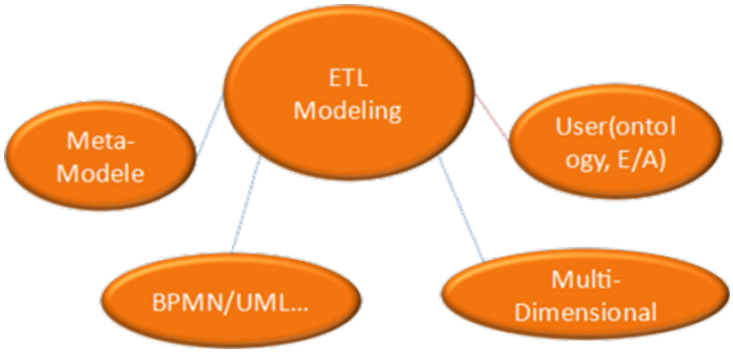

FIGURE 3.10 – *Classe 3.*

3.4.4 Classe 4 :

Cette classe illustrée sur la figure 3.11 avec le constat : la plupart des travaux de recherche déploient leur entrepôt de données sur un magasin, via des SGBD relationnels, tandis que d'autres utilisent un déploiement sémantique et permettent le stockage d'ontologies en OWL ou RDF. D'autres études connexes et avec l'menrgence des polystores ont envisagé un déploiement sur des multi-magasins, des technologies conçues pour interroger et stocker des données provenant de diverses sources de données. D'aprés une analyse profonde, les recherches existantes dans ce domaine ne sont pas encore mûres, une grande partie de nos prochaines études seront consacrées au déploiement sur le système polystore.

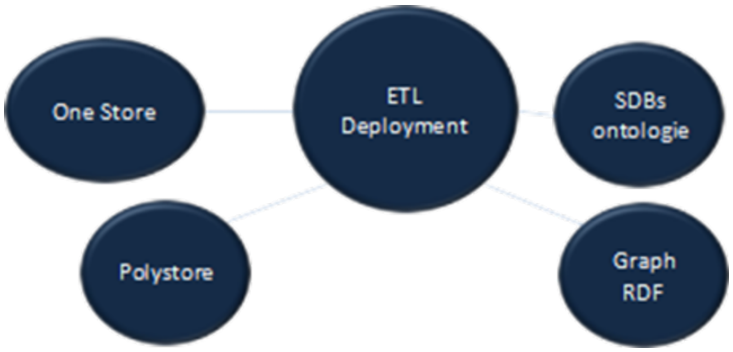

FIGURE 3.11 – *Classe 4.*

3.4.5 Classe 5 :

Avec l'émergence des nouvelles plates-formes de déploiement de matériel avancé telles que les systèmes polystore, des études récentes soulignent leurs importance avec une valeur ajoutée pour les applications d'entrepôts de données modernes. Nous distinguons dans cette classe les différents type de déploiement d'un point de vue matériel et logiciel (graphe ou sémantique). Les modèles de déploiement de stockage peuvent suivre différentes représen-

tations en fonction d'exigences spécifiques. Un entrepôt de données peut être déployé selon plusieurs schémas de stockage : modèles horizontaux, verticaux, hybrides, NoSQL, etc.

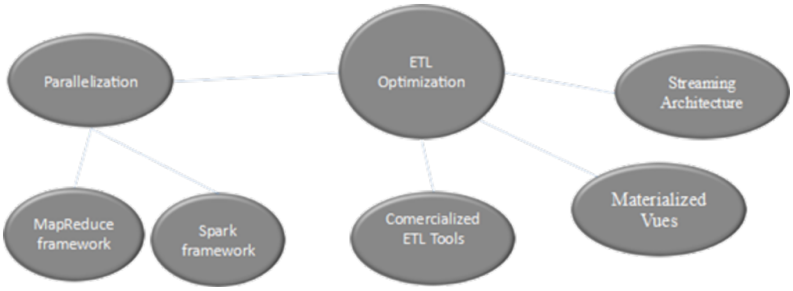

FIGURE 3.12 – *Classe 5.*

3.5 Le processus d'évaluation d'un système ETL

Dans cette section, nous présentons les résultats de notre enquête sous forme d'une méthode d'évaluation qui répond à la question : **Quels sont les attributs de qualité à prendre en compte lors de la conception de solutions ETL ?** et sur la base des critères surmentionnés nous déffinissons un modèle de qualité suivant lequel on peut aprécier la qualité des approches actuellement utilisées pour mettre en œuvre des solutions ETL.

3.5.1 Modèle d'évaluation de la qualité d'un processus ETL

Nous définissons un modèle qui définit la qualité de l'ETL à travers des attributs de la qualité attendue, à savoir : (i) la performance ; (ii) efficacité ; et (ii) l'évolutivité du processus du service rendu. Ce modèle peut être représenté sous forme d'arborescence (figure 3.13.

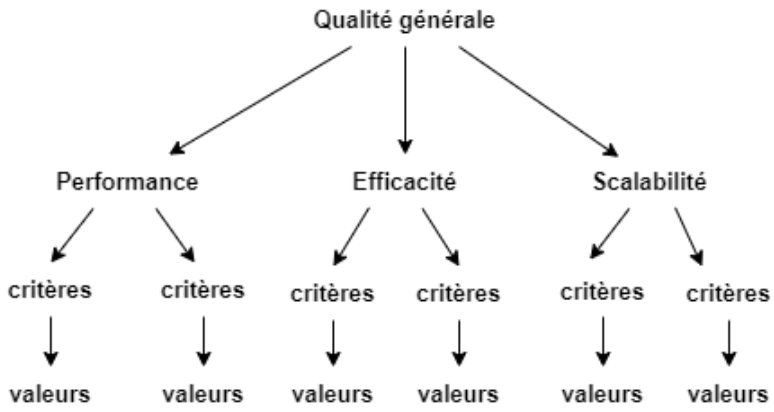

FIGURE 3.13 – *Le modèle d'évaluation proposé.*

- Une valeur est la mesure d'une propriété d'un critère. (Par exemple, un magasin pour le critère "Déploiement et Stockage").

- Les principales qualités d'un ETL sont le respect des différentes valeurs des critères déjà cités. Au vu de notre modèle, un excellent processus ETL doit donc présenter tous ces critères.

la modélisation, les sources de données impliquées dans l'intégration, le déploiement du processus ETL, le type de développement, et enfin la technique d'optimisation de ces processus.

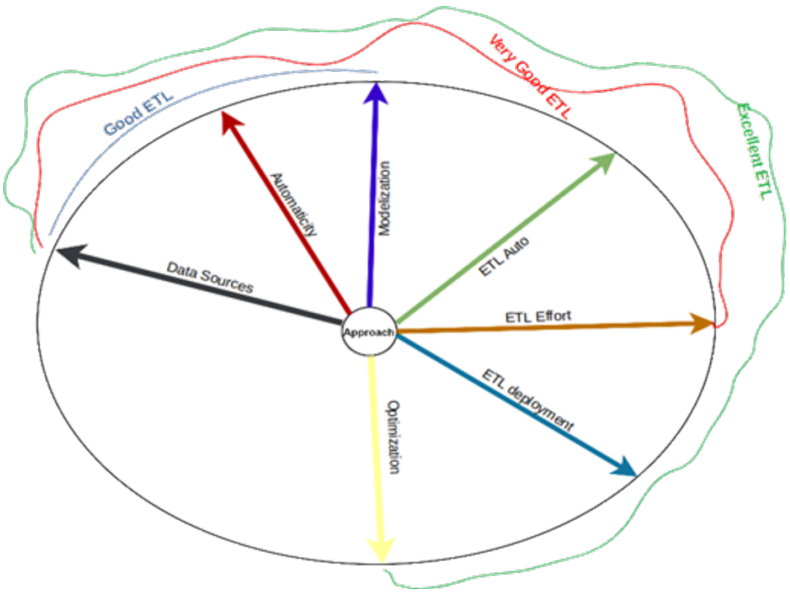

FIGURE 3.14 – *Qualité du processus ETL.*

Conclusion du chapitre

En conclusion, cette enquête menée dans le domaine de l'ETL a mis en évidence plusieurs facteurs importants qui affectent le succès des processus ETL. Il s'agit notamment des défis d'intégration de données hétérogènes, des problèmes de performances, d'optimisation et d'évolutivité. Pour surmonter ces défis, les organisations doivent adopter les meilleures pratiques dans la conception et la mise en œuvre des processus ETL.

Dans l'ensemble, les résultats de l'enquête suggèrent que les processus ETL continuent d'être essentiels pour les organisations dans la gestion et l'exploitation efficaces des données, et il est nécessaire d'innover et d'améliorer continuellement dans ce domaine pour suivre le rythme de l'évolution du paysage des données.

Nos recherches ont révélé que plusieurs approches sont proposées pour résoudre le problème des conflits sémantiques et structurels. De plus, de nombreuses études se sont concentrées sur la modélisation et l'optimisation du processus ETL.

Le monde universitaire a développé plusieurs outils ETL open source populaires basés sur du code, tels que Petl et Pygrametl. Cependant, compte tenu de l'expansion rapide des données d'entreprise, une quantité considérable de recherches se concentre actuellement sur l'intégration du Big Data dans les processus ETL, où les tâches d'extraction et de transformation sont effectuées par Spark ou MapReduce.

D'autres travaux exploitent les avantages du cloud computing. Dans nos travaux, nous avons proposé une nouvelle méthode de classification de la littérature récente sur le domaine des processus ETL selon un ensemble de critères. Nous classons ces processus d'intégration ETL en sept classes.

Sur la base de la revue de la littérature et des forces et faiblesses identifiées, on peut conclure qu'il existe de nombreuses opportunités de recherche dans les différentes classes identifiées au sein des environnements d'entreposage de données. De nombreux articles ont présenté des idées initiales de solutions, mais peu ont entièrement résolu le problème. Nous développons actuellement une série d'approches dans cette lignée.

Après analyse, nous avons identifié cinq classes : (i) sources de données ; (ii) la mise en œuvre de l'ETL ; (iii) Modélisation ETL ; (iv) effort ETL ; (v) Déploiement ETL et, (vi) Optimisation ETL. Une comparaison des travaux mentionnés dans cette enquête nous a conduit à suggérer quelques questions de recherche ouvertes.

Contributions

4

Sommaire

4.1 Introduction

Dans ce chapitre nous rapportos nos contributions avec les résultats expérimentaux concernant la modélisation conceptuelle et les méthode d'optimisations des processus ETL dans un environnement fortement varié. En guise de preuve de concept de nos approches, nous avons implémenté des prototypes scala pour supporter les différents opérateurs des processus ETL afin de valider et évaluer la pertinence de nos différents approches.

4.2 Vers un ETL évolutif et efficace

Dans cette section, nous présentons les principales caractéristiques et principes de notre premier système proposé qui est illustré à la figure 4.1. Tout d'abord, nous présentons comment créer une conception ETL BPMN. Deuxièmement, nous décrivons comment adapter le processus ETL traditionnel afin de le distribuer et de permettre son exécution en parallèle sur un cluster Spark. Nous soulignons l'importance du partitionnement des données, dans un environnement distribué, puis nous présentons les types de partitionnement des données fournis dans notre approche. Notre approche a pour but l'optimisation des processus ETL à travers la parallélisation de ces processus via le framework Spark, qui prend en charge le partage de données en mémoire entre plusieurs tâches.

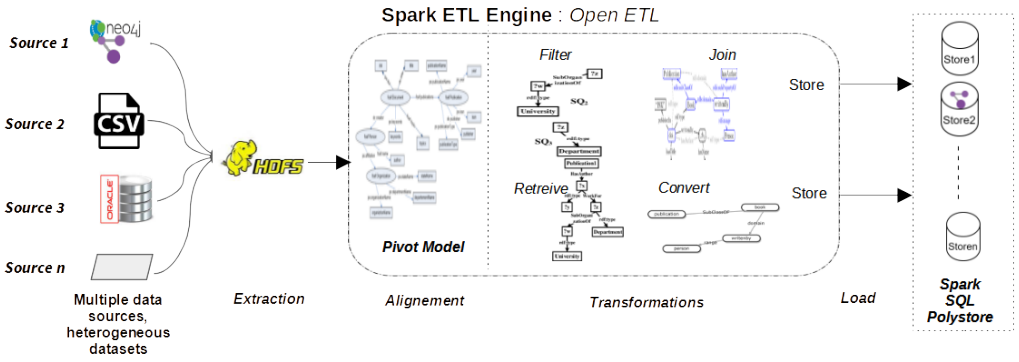

FIGURE 4.1 – *Architecture générale de notre système proposé.*

Nous traduisons le modèle BPMN directement en scripts écrits en langage scala, en suivant le modèle d'exécution Spark illustré à la figure 4.2 [92]. Les scripts scala sont automatiquement parallélisés et exécutés sur un cluster de trois machines (un Namenode et deux Datanodes).

Le modèle d'exécution est basé sur l'architecture driver/worker composée des processus driver et worker. Le processus driver crée le contexte Spark et planifie les tâches en fonction des nœuds de driver disponibles. Initialement, le processus maître doit être démarré, puis la

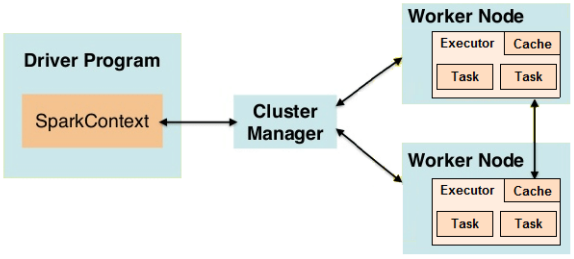

FIGURE 4.2 – *Modèle d'exécution Spark*

création des nœuds worker suit. Le driver prend la responsabilité de convertir l'application d'un utilisateur en plusieurs tâches. Ces tâches sont réparties entre les workers. Les exécuteurs sont les principaux composants de chaque application Spark. Les exécuteurs effectuent en fait le traitement des données, la lecture et l'écriture des données dans les sources externes et le système de stockage. Le Spark manager est responsable de l'allocation et de la désallocation des ressources au Spark job. Fondamentalement, Spark n'est qu'un modèle de calcul. Il n'est pas lié au stockage des données, qui est un concept différent. Il n'aide que dans les calculs et l'analyse des données de manière distribuée. Pour une exécution distribuée, la tâche est répartie entre les nœuds connectés afin que chaque nœud puisse effectuer des tâches en même temps ; il effectue l'opération souhaitée et notifie le maître à la fin de la tâche.

Étant donné que les entrepôts de données traditionnels ne sont pas suffisamment optimisés pour stocker des quantités énormes et diverses de données [56], nous utilisons dans notre approche une nouvelle architecture d'entreposage de données telle que polystore.

4.2.1 La nouvelle architecture

L'approche proposée à la figure 4.1consiste à : Concevoir un processus ETL codé en scripts scala impliquant des sources de données hétérogènes et voulunineuses afin d'assurer la scalabilité et de réduire le temps de latence entre un entrepôt de données et ses sources.

L'entrepôt est déployé sur le polystore hybride Spark SQL. La méthode proposée consiste en cinq étapes :

— Extraction et transfert de données de flux : il s'agit d'une mise à jour des sources de données vers le système de fichiers distribué HDFS (Hadoop Distributed File System). Nous avons utilisé la technique de capture que nous avons codée en scala ;

— Modélisation : elle consiste à produire un modèle UML « pivot » à partir des modèles des différentes sources de données ;

— Mapping : il consiste à transformer chaque objet BPMN en son équivalent dans le script scala ;
— Transformation : elle consiste à effectuer les transformations ETL appropriées avec des scripts scala en utilisant à la fois les opérateurs ETL traditionnels et les opérateurs ETL de gestion des représentations de graphes définis dans [89] ;
— Déploiement ETL sur le polystore hybride Spark SQL.

Le moteur Spark ETL. Apache Spark est une plate-forme informatique en cluster avec des bibliothèques intégrées et des API Python, Java, scala et SQL simples conçues pour être rapides et polyvalentes. Le concept MapReduce est étendu par Spark pour gérer efficacement des types de calcul supplémentaires, tels que les requêtes interactives et le traitement en continu. Pour les applications complexes fonctionnant sur disque [93], Spark est plus efficace que MapReduce car il permet d'effectuer des calculs en mémoire. L'ensemble de données distribuées résilientes de Spark (RDD) sert de base.

Traitement parallèle dans Spark. Les applications Spark s'exécutent en tant que processus indépendants qui résident sur des clusters et sont coordonnées par SparkContext dans le programme principal. La première étape de l'exécution d'un programme Spark consiste à soumettre la tâche à un cluster à l'aide de spark-submit. Sparkcontext achemine le programme vers les modules, tels que le nœud maître du cluster, ainsi que les RDD sont créés par ces programmes de driver Sparkcontext. Le programme est ensuite transmis au nœud maître du cluster. Chaque cluster a un nœud maître qui effectue tout le traitement nécessaire. Il transmet ensuite le programme aux nœuds driver. Le nœud de driver est le résolveur de problèmes. Les nœuds maîtres contiennent des exécuteurs qui s'exécutent avec le driver Sparkcontext.

Processus ETL vs RDD. Il existe une grande correspondance et similarité entre les processus ETL et RDD. Les RDD sont des collections immuables et distribuées d'objets de tout type. Comme son nom l'indique, il s'agit d'enregistrements résilients (tolérants aux pannes) de données qui résident sur plusieurs nœuds. Intuitivement, un processus ETL peut être considéré comme un graphe orienté acyclique (DAG), les activités et les ensembles d'enregistrements étant les nœuds du graphe et les relations d'entrée-sortie entre les nœuds étant les bords du graphe [19]. Dans RDD, lorsque l'ensemble de calcul est créé, formant un DAG, il n'effectue aucune exécution, mais il se prépare à l'exécution à la fin. Après l'extraction, le processus ETL effectue deux types de transformations sur les données : La standardisation des données (nettoyage, filtrage, conversion, ...) et permet aux secondes de fusionner, agréger. Ainsi, le processus ETL correspond bien au modèle d'exécution RDD de Spark.

4.2.2 Méta-modèle de processus ETL

L'objectif de cette section est de fournir un méta-modèle du processus ETL qui prend en charge la modélisation générique des activités dans l'environnement ETL. Un processus ETL peut être considéré comme un type particulier de processus métier. Un processus ETL est un type particulier de processus métier, mais il n'existe pas de modèle standard pour définir ce processus. Comme pour tout processus, l'environnement ETL doit d'abord être modélisé avant que l'un de ses composants puisse être utilisé. Un nombre important de recherches se sont concentrées principalement sur la modélisation des activités ETL pour produire le processus ETL [4], [94], [22] et [95].

Définition. Un processus ETL est composé de nombreuses activités et de transitions entre elles. Une transition détermine l'ordre dans lequel les activités sont exécutées pour fournir un flux de données des sources aux magasins de destination, et une activité correspond à un scénario ETL. Les éléments des sources de données, des magasins et des opérateurs ETL sont tous utilisés dans une activité pour définir des expressions qui décrivent la sémantique des données transmises au schéma cible. Chaque activité est soit simple, indécomposable, soit complexe. Afin de contribuer à la généralisation de l'ensemble de l'environnement ETL, nous fournissons un méta-modèle du processus ETL qui permet une modélisation générique des activités de l'environnement ETL. Dans (figures 4.3 et 4.4), nous décrivons les constructions composant la palette ETL proposée. Ces constructions sont regroupées en quatre catégories : les opérateurs [62], les objets d'activité et les objets de workflow. Pour les autres objets (artefacts, connexions, contrôle et flux), nous utilisons les constructions proposées par l'outil de modélisation Camunda [96].

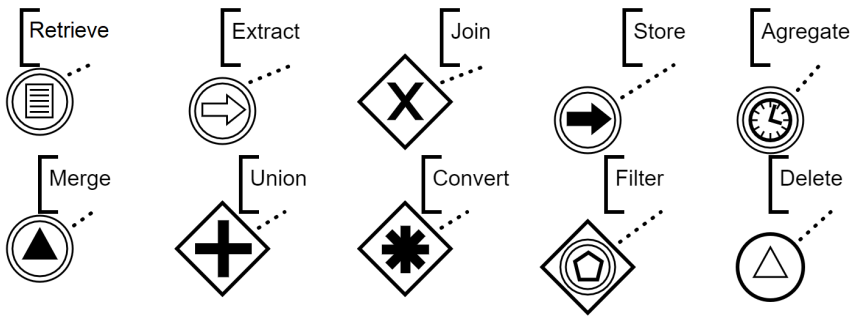

FIGURE 4.3 – *ETL Operators.*

4.2.3 Modèle vers Modèle de script

Pour traduire le modèle conceptuel ETL, nous faisons le travail de concepteurs, similaire à ce qui se passe lors de la traduction de modèles conceptuels, tels que le modèle Entity-

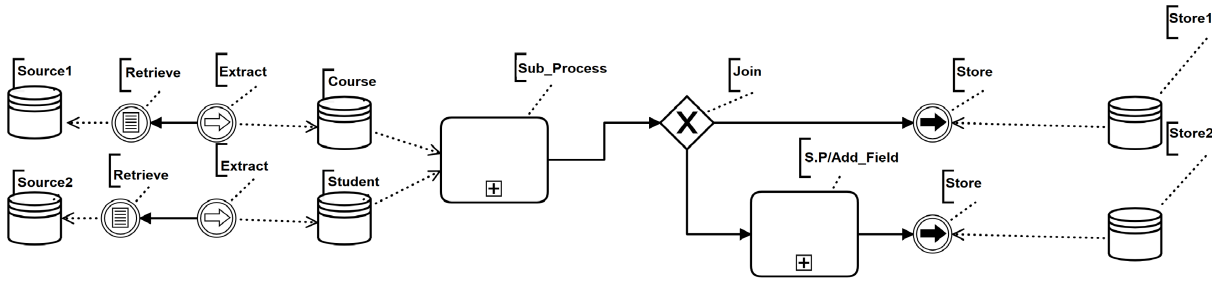

FIGURE 4.4 – *ETL Activity.*

Association, en modèles logiques. De plus, le concepteur doit décomposer chaque activité de haut niveau en un sous-processus détaillé et répéter le processus jusqu'à ce que les opérateurs ETL implémentables soient atteints. Vient ensuite le mapping, qui consiste à transformer chaque objet BPMN en son équivalent dans le script scala figure 4.5.

```
// Extraction à partir du fichier csv
val csvdata_file = spark.read.option("header", "true").option("sep", ",").option("infercSchema",
"true").csv("Path/dataset/Etudian.csv").cache();

// Transformation : appliquer le nouveau schema
val csv_datafile1 = spark.read.option("header", "true").option("sep", ",")
.schema(csvnouveaushemaDF)
.csv("Path/dataset/Etudiant.csv")
csv_datafile1.printSchema()

// Déploiement finale dans DW1
val url3="jdbc:mysql://localhost:3306/data_mart";
val table_oracle_etudiant = "datamart_etd"
DW_DF.write.mode(SaveMode.Overwrite).jdbc(url3, table_oracle_etudiant,properties)
```

FIGURE 4.5 – *Script scala appliqué au fichier CSV.*

4.2.4 Déploiement sur polystore

Les auteurs dans le travail [97], divisent les solutions existantes pour la gestion de données multi-modèles en quatre groupes : les systèmes fédérés, les systèmes polyglottes, les systèmes multi-magasins et les systèmes polystore. Un système polystore est un système de base de données avec de nombreux magasins de données hétérogènes et diverses interfaces de requête, selon [98].

Dans notre solution, nous choisissons de déployer l'entrepôt de données cible sur le polystore hybride Spark SQL dont l'architecture est illustrée à la figure 4.6 [54], l'architecture polystore Spark SQL est basée sur [52], p.29. Cette architecture fonctionne comme une bibliothèque au-dessus de Spark. Le déploiement se fait selon la cible, (i) dans une représentation verticale utilisant le SGBD Neo4j, dans laquelle on peut représenter des instances et des graphes, et (ii) dans une représentation relationnelle utilisant le SGBD Oracle.

Nous avons appliqué notre programme ETL basé sur des scripts scala pour remplir le schéma d'entrepôt de données cible. Nous supposons qu'avec n sources de données nous pouvons déployer l'entrepôt de données final selon les exigences annoncées au préalable par

le concepteur de l'entrepôt. Spark SQL est un système polystore hybride avec des sources de données externes faiblement couplées et Spark/HDFS étroitement couplés. En plus de Spark, il fonctionne comme bibliothèque.

L'interface Spark Java fournit un accès direct au moteur Spark pour le processeur de requêtes, tout en utilisant des wrappers pour accéder à des sources de données externes via l'interface Spark SQL commune (tel qu'un SGBD relationnel ou un magasin Neo4j).

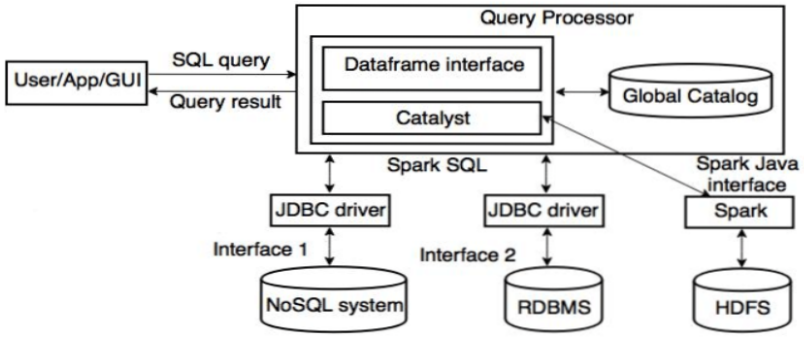

FIGURE 4.6 – *Spark SQL polystore architecture.*

4.2.5 Étude de Cas

Nous illustrons un exemple de scénario à travers un entrepôt de données. La direction de la formation du Ministère de l'Enseignement Supérieur et de la Recherche souhaite construire un entrepôt de données qui consiste à collecter des informations décisionnelles sur les performances des étudiants, à savoir leurs performances.

Ensembles de données

Le schéma de l'entrepôt de données est présenté dans la figure 4.7, tirée du benchmark LUBM (http ://swat.cse.lehigh.edu/projects/lubm/) (figure 4.8) lié au domaine de l'université. ainsi que le processus global de chargement des données des sources (nous présentons l'exemple d'un fichier RDF, figure 4.9) vers l'entrepôt de données cible. Nous utilisons l'outil UBA est fourni par LUBM pour générer des données sur l'ontologie Univ-Bench. Nous avons utilisé ces ensembles de données pour simuler des graphiques basés sur des sources de données externes. Nous avons également considéré les sources de données CSV et une base de données relationnelle comme sources internes. (i) Source 1 deux bases de données de graphes Neo4j alimentées à l'aide de l'ontologie Lehigh University Benchmark (LUBM) ; (ii) La Source 2 est un fichier CSV ; et (iii) la Source 3 est une base de données Oracle. Dans notre cas, les tables de destination sont initialement vides puis remplies de nouvelles données. Pour

ce faire, un modèle pivot (schéma RDF) est conçu. Ce modèle de données fournit une base pour l'intégration.

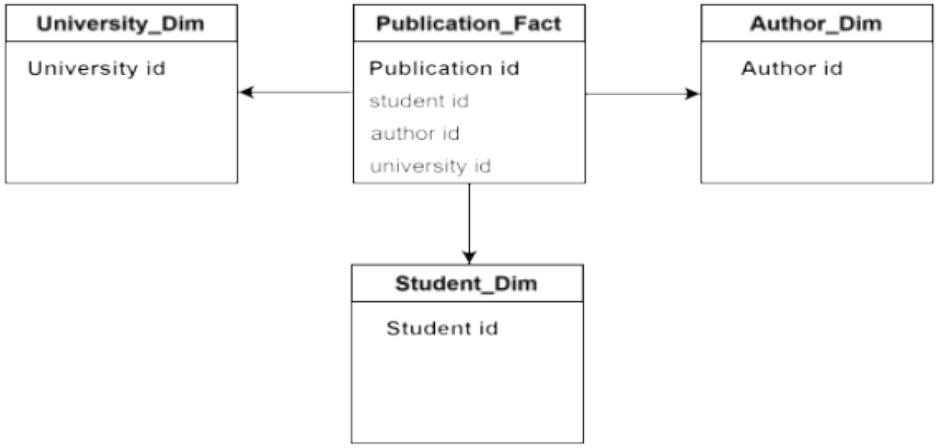

FIGURE 4.7 – *Schema de l'Entrepôt de données.*

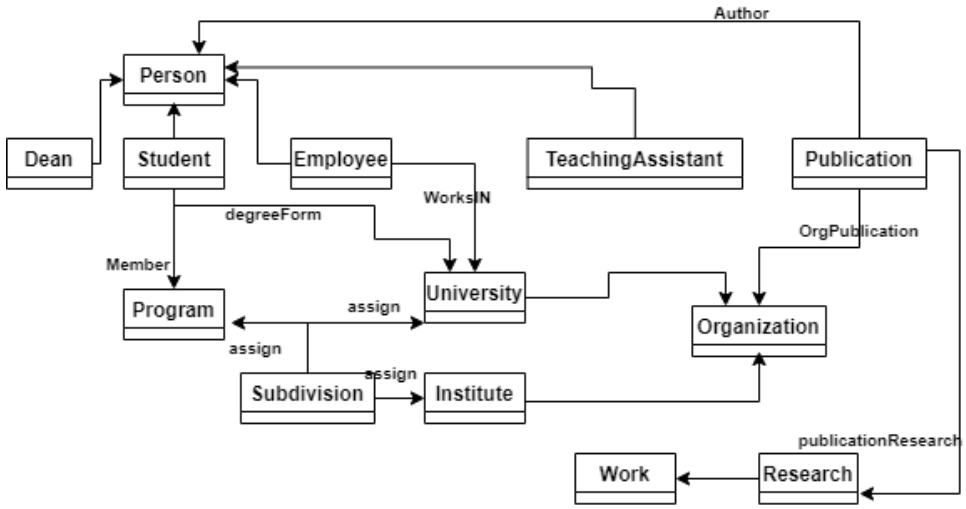

FIGURE 4.8 – *Schema de l'Ontologie LUBM.*

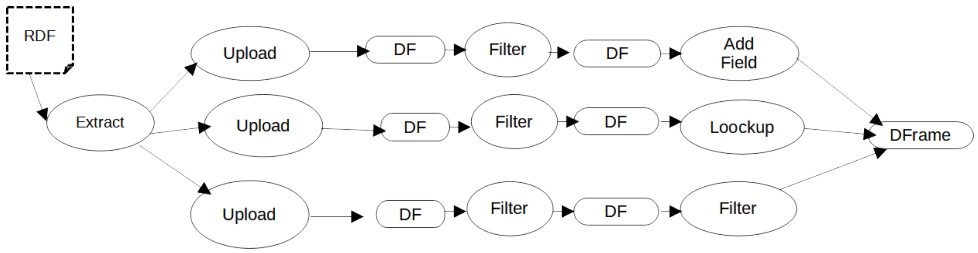

FIGURE 4.9 – *Partitionnement et distribution dans Spark.*

Mesures

Nous évaluons les performances du système proposé à travers un ensemble d'expériences en considérant trois critères : (i) la scalabilité, (ii) le temps de réponse de notre approche par

rapport à un outil ETL PDI (Pentaho data integration tool) [32], et (iii) le temps passé à charger les changements qui ont affecté les sources de données.

Le processus ETL

Le code de script ETL importe les données d'un fichier RDF vers plusieurs nœuds avec Spark. Spark change les transformations de RDD en DAG (Directed Acyclic Graph) et démarre l'exécution. DAG convertit un plan d'exécution logique en un plan d'exécution physique. Lorsqu'une activité est invoquée, le DAG est soumis au planificateur DAG. Il divise les opérations en étapes, et chaque étape consiste en une unité de travail appelée tâche. Ces tâches sont ensuite transmises au gestionnaire de tâches via un gestionnaire de cluster, ce qui permet de lancer l'application sur différentes machines. La figure 4.9 décrit l'ensemble du processus ETL appliqué au fichier RDF.

Après avoir appliqué notre algorithme Open ETL, le processus ETL dans BPMN est illustré à la figure 4.10, l'entrepôt de données résultant comporte deux magasins : une base de données de graphes Neo4j et une base de données Oracle. Le traitement est effectué dans le framework Spark du point de vue de la distribution.

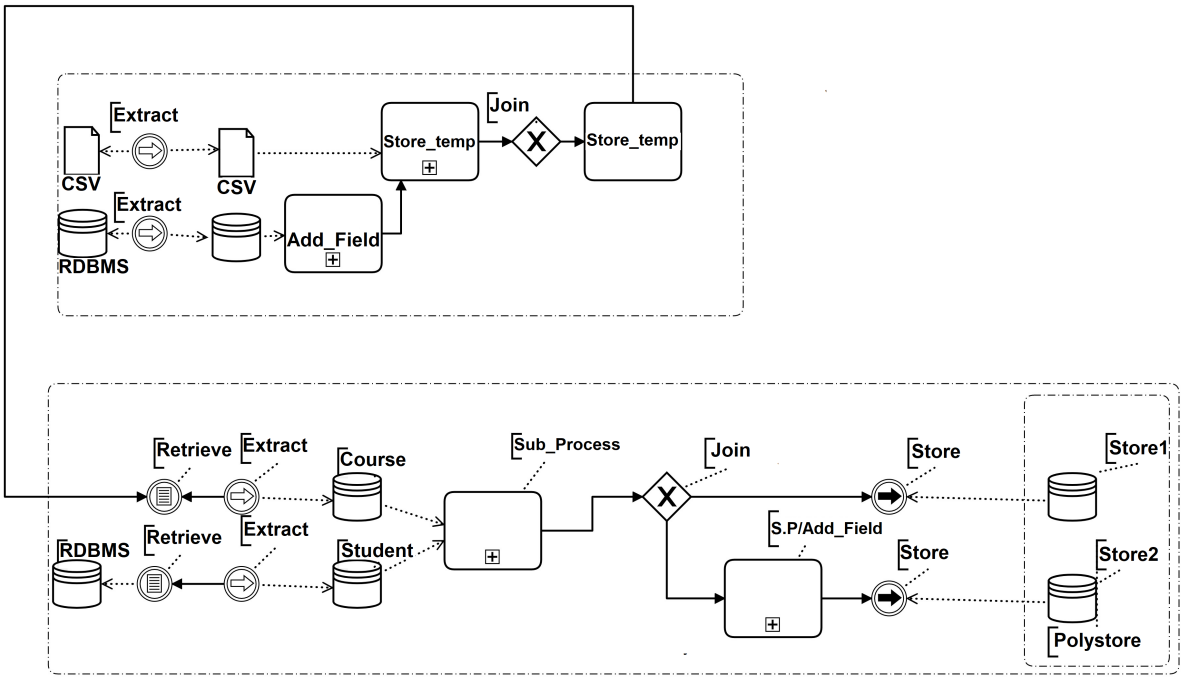

FIGURE 4.10 – *Le modèle BPMN du processus ETL.*

4.2.6 Expériences et Résultats

Cette section présente l'évaluation du système proposé, qui est basée sur l'étude de cas susmentionnée. Nous avons mené trois expériences distinctes. L'expérience initiale a examiné l'évolutivité de la solution proposée en faisant varier la taille de la source de données. La deuxième expérience s'est concentrée sur l'évaluation et la comparaison du temps de réponse du système proposé avec l'outil PDI. La troisième expérience s'est concentrée sur l'évaluation du temps de chargement du processus ETL après un changement des sources.

Nos expériences ont été menées sur un PC avec un processeur Intel(R) Core(TM) i5-8365U (1,9 GHz, 1,6 GHz), 8 Go de mémoire principale sur un disque SSD de 256 Go exécutant Windows 10x64, Oracle DBMS, base de données Neo4J desktop, Apache Spark 3.0 et scala 2.12 étaient nécessaires pour exécuter les tests. Tous les outils logiciels choisis ont été installés sur la machine locale à des fins d'évaluation.

Expérience 1.

Dans cette expérience, nous avons considéré l'évolutivité de notre approche en faisant varier la taille des sources de données et, par conséquent, le nombre de tâches qui s'exécutent en parallèle sur le cluster pour traiter les données.

La traduction du modèle conceptuel BPMN des processus ETL en scripts scala permet de remplir le schéma cible de l'entrepôt de données. Nous mesurons le temps passé à intégrer des sources de données hétérogènes de tailles différentes. La figure 4.11 illustre les résultats obtenus. On peut remarquer que l'augmentation de la taille des sources améliore le temps de traitement.

Expérience 2.

Dans cette expérience, nous visons à montrer que la modélisation du processus ETL avec BPMN et sa traduction en scripts codés en scala et exécutés sur Spark aboutissent à des processus plus efficaces que ceux résultant de la traduction de BPMN en outils ETL. Nous évaluons le temps de réponse de notre proposition par rapport à l'outil PDI. Les résultats sont présentés dans la figure 4.12. Nous notons que notre approche augmente considérablement le temps de réponse du processus ETL.

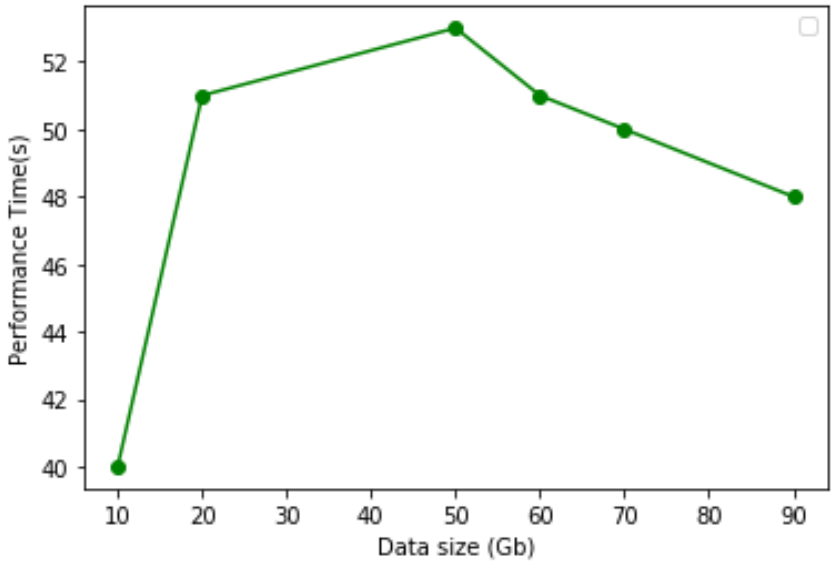

FIGURE 4.11 – *scalability of the proposed approach.*

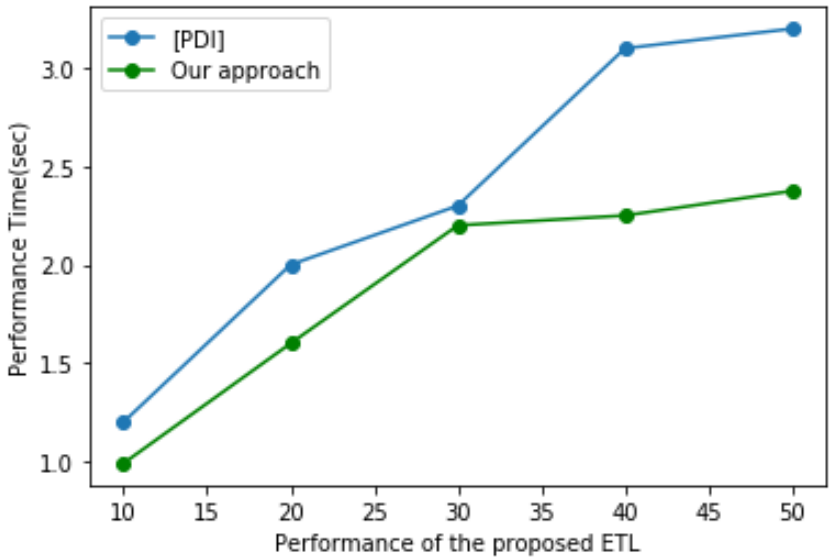

FIGURE 4.12 – *Performance of the proposed ETL.*

Expérience 3.

Dans cette expérience nous évaluons le temps de chargement du processus ETL après un changement dans les sources. La figure 4.13 illustre notre découverte selon laquelle le temps de chargement augmente après une mise à jour. Le chargement via notre approche est supérieur à celui implémenté sur l'outil PDI.

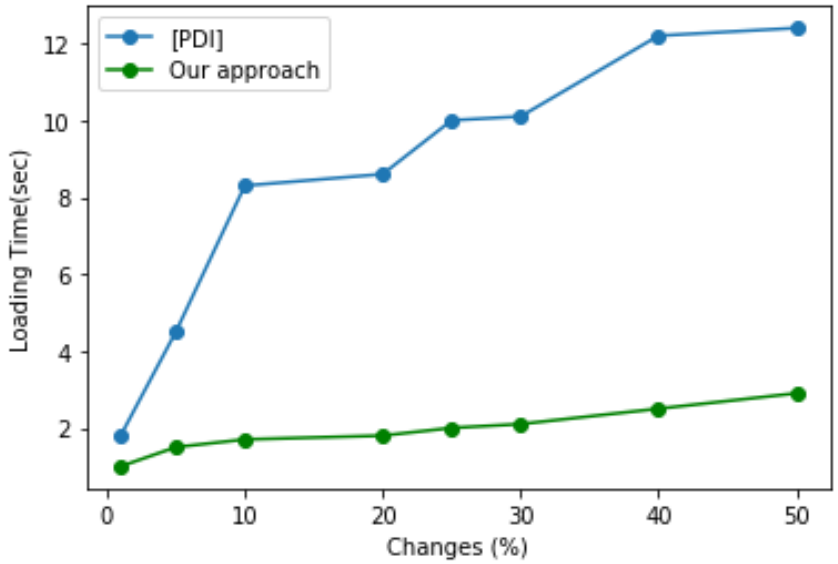

FIGURE 4.13 – *Load Time Comparison.*

4.2.7 Conclusion

Cet travail a introduit une nouvelle méthode qui comprend à la fois des méthodes de modélisation et d'optimisation des processus ETL, en respectant l'importance de la question et les défis dans ce domaine. Nous avons présenté une nouvelle approche pour optimiser les processus ETL grâce à une nouvelle architecture distribuée qui prend en charge à la fois le traitement par lots et traitement de flux. Nous avons présenté une nouvelle approche pour concevoir des processus ETL en utilisant un ensemble de notations en BPMN pour les opérateurs ETL capables de modéliser des activités ETL ainsi que des transformations de modèles en scripts scala qui sont automatiquement convertis en scripts à implémenter dans le framework Spark. Une solution de déploiement polystore hybride basée sur Spark SQL est donnée. Nous avons validé les étapes concrètes de l'utilisation de la solution proposée pour charger les données dans un schéma d'entrepôt de données.

Les résultats importants sont liés au temps de performance du système proposé par rapport aux autres alternatives ETL. De plus, nous avons également présenté l'évolutivité de la méthode proposée sur de grands ensembles de données. Cet article a introduit une nouvelle méthode qui comprend à la fois des méthodes de modélisation et d'optimisation des processus ETL, en respectant l'importance de la question et les défis dans ce domaine. Nous avons présenté une nouvelle approche pour optimiser les processus ETL grâce à une nouvelle architecture distribuée (Open ETL) qui prend en charge à la fois le traitement par lots et traitement de flux. Nous avons présenté une nouvelle approche pour concevoir des processus ETL en utilisant un ensemble de notations en BPMN pour les opérateurs ETL capables de modéliser

des activités ETL ainsi que des transformations de modèles en scripts scala qui sont automatiquement convertis en scripts à implémenter dans le framework Spark.

Une solution de déploiement polystore hybride basée sur Spark SQL est donnée. Nous avons validé les étapes concrètes de l'utilisation de l'approche Open ETL proposée pour charger les données dans un schéma d'entrepôt de données. Les résultats importants sont liés au temps de performance du système proposé par rapport aux autres alternatives ETL. De plus, nous avons également présenté l'évolutivité de la méthode proposée sur de grands ensembles de données. Ainsi, les travaux futurs pourraient se concentrer sur la conception d'un framework ETL basé sur un code unifié impliquant un support de gestion des données de séries chronologiques (time-series data). De plus, dans nos futures recherches, nous prévoyons de comparer les performances de notre approche avec d'autres alternatives ETL.

4.3 Os-ETL : une solution Open-scala efficace pour l'intégration de données hétérogènes dans l'entreposage de données big data

4.3.1 Introduction

De nos jours, les organisations génèrent de plus en plus de grandes quantités de données dans une grande variété de formats à haut débit (Figure 1). Nous sommes à l'ère du big data [37]. Les chercheurs définissent le big data par les quatre V suivants : Volume, Variété, Vélocité et Véracité. Ces quatre dimensions caractérisent et distinguent le big data des données ordinaires. Aussi, la croissance rapide des base de données de structure graphes (données RDF) est une excellente opportunité d'enrichir les entrepôts de données traditionnels avec une nouvelle dimension en V du big data : La Valeur. Ainsi, permettre aux entreprises d'exploiter ces données riches pour accroître leur valeur ajoutée dans un monde hautement concurrentiel.

Quelques études se sont principalement concentrées sur l'intégration de données RDF dans un entrepôt de données et sur le processus de déploiement ETL basé sur un système de stockage polystore. Le premier travail utilisant un véritable système polystore, mais avec des données relationnelles, a été proposé par [9]. Dans le travail [99], les auteurs ont prouvé qu'une approche de charge de travail dynamique est nécessaire pour le placement des données dans un système polystore afin de prendre en charge l'ingestion de données à faible latence. De plus les auteurs de [68], ont choisi de déployer l'entrepôt de données en représentation verticale avec le SGBD Oracle qui, propose un modèle de stockage pour représenter les instances et les graphes, en utilisant Oracle RDF Semantic Graph. Ils ont planifié une simulation pour déployer un entrepôt de données sur un polystore. Certains travaux atteignent le bon degré de parallélisme des processus ETL, en fournissant les fonctions Map et Reduce, connues sous le nom de framework MapReduce (c'est-à-dire les paramètres de partition, le nombre de nœuds) [87].

Cependant, ce dernier présente certaines limites. En effet, il n'accepte qu'un seul flux de données d'entrée à la fois dans un format clé/valeur, il ne prend pas non plus en charge un traitement en temps réel.

De plus, le développeur doit écrire du code personnalisé pour les fonctions Map et Reduce, ce qui est difficile à maintenir et à retraiter. De plus, les entreprises doivent aujourd'hui prendre des décisions en temps réel sur la base de grandes quantités de données, fournies par des sources de données structurées en graphes. Ainsi, nous soutenons que pour les applications modernes d'entrepôt de données (big data warehouse), dans lesquelles la latence

et l'évolutivité sont d'une grande importance, le processus ETL devrait être optimisé à l'aide de nouvelles technologies telles que le framework Spark. Nous affirmons que cela nécessite une nouvelle architecture comprenant un module Spark pour créer de nouvelles applications d'entrepôt de données. De plus, l'objectif de ce travail est de résoudre le processus difficile de déploiement d'un entrepôt de données à structure graphe sur un système de stockage polystore, qui implique deux phases : l'allocation et le partitionnement des données.

Ainsi, dans ce travail, nous proposons une nouvelle solution pour résoudre le problème de déploiement d'un entrepôt de données à structure de graphes sur un polystore, qui est une tâche difficile, avec deux phases à savoir le partitionnement et l'allocation des données. L'architecture proposée pour ETL nommée Open-scala-ETL, qui se compose de trois composants : (i) un collecteur de mises à jour de données (technique CDC) ; (ii) un moteur Spark ETL avec deux modules pour la fragmentation et l'affectation ; et (iii) un polystore de déploiement.

De plus, pour surmonter les limitations du framework MapReduce, nous utilisons le calcul parallèle en mémoire (framework Spark), qui utilise un modèle de graphe acyclique direct (DAG), pour calculer les composants plusieurs fois et prend en charge le partage de données en mémoire entre plusieurs tâches. Ainsi, l'utilisation de nouvelles technologies de parallélisation est la clé pour obtenir une meilleure évolutivité et des performances ETL. La principale motivation de l'architecture proposée pour les processus ETL est le besoin d'améliorer l'efficacité et les performances de ces processus qui traitent des données hétérogènes et volumineuses par rapport aux autres techniques d'optimisation existantes basées sur ETL. Les auteurs de [100], présentent une nouvelle approche basée sur le contrôle de flux dans ETL. Les auteurs de [101], introduisent une approche basée sur le flux de contrôle pour la modélisation des processus ETL et utilisent la combinaison de la parallélisation et de la mémoire cache partagée pour optimiser les performances ETL de l'entrepôt de données.

Ainsi, la solution proposée implique diverses sources de données (relationnelles, graphiques et CSV, etc.) et tire parti de la puissance de calcul et du stockage distribué, offerts par le framework Spark pour optimiser les processus ETL. Les universitaires et l'industrie ont adopté Apache Spark comme un framework rapide et évolutif [102].

Notre nouveau framework permet le développement d'entrepôts de données multidimensionnels capables de gérer de grandes masses de données. Nous assurons le déploiement sur polystore. Par conséquent, notre approche tire parti de différents systèmes de stockage (polystore) pour augmenter les performances d'exécution des algorithmes d'analyse.

4.3.2 Le Framework proposé Os-ETL

La figure 4.14 illustre l'architecture de l'approche proposée nommée processus Open scala ETL (Os-ETL). Il se compose de six étapes : (i) extraire et transférer les données de flux ; (ii) l'alignement de toutes les sources de données sur le modèle de données RDF ; (iii) partitionnement des sources de données (iv) transformation des sources de données RDF en GraphX ; (v) les transformations appropriées ; et (vi) allocation et distribution des fragments RDF obtenus sur le polystore Spark SQL. Ces étapes sont décrites en détail comme suit :

— Étape 1 : extraire et transférer les données de flux, qui consistent en une mise à jour des sources de données vers le système de fichiers distribués Hadoop (HDFS), en utilisant la technique de capture de données modifiées (CDC) qui identifie les modifications apportées aux sources de données via des déclencheurs implémentés dans chaque source participant à l'ETL.

— Étape 2 : alignement de toutes les sources de données sur le modèle de données RDF. Cette étape implique un processus manuel d'analyse des ensembles de données. Nous créons un modèle pivot pour réduire le nombre de mappages pour les différents formats des sources de données impliquées dans l'ETL.

— Étape 3 : le partitionnement des sources de données, en l'occurrence les algorithmes adaptés nommés path partitioning (Figure 4.15) introduits dans [103], a été employé. Dans notre scénario d'entrepôt de données, la configuration d'un polystore implique de résoudre les problèmes de partitionnement et d'allocation des données graphe.

— Étape 4 : transformer les sources de données RDF en GraphX [104]. GraphX est l'API Spark pour les graphes et le calcul parallèle aux graphes.

— Étape 5 : effectuez les transformations appropriées à l'aide de scripts scala. Dans cette étape, les opérateurs ETL traditionnels et les opérateurs ETL de gestion des représentations de graphes ont été utilisés.

— Étape 6 : allocation et distribution des fragments RDF obtenus sur le polystore Spark SQL. Dans notre solution d'intégration de différents types de données, nous utilisons la solution de plusieurs SGBD ensemble.

Le système Os-ETL proposé implique des sources de données hétérogènes pour assurer l'évolutivité et réduire la latence entre l'entrepôt de données et ses sources. De plus, pour l'étape (ii), l'alignement de toutes les sources de données sur le modèle de données RDF nécessite de modifier le modèle pivot qui a été présenté dans [105], afin d'unifier tous les formalismes du modèle de données.

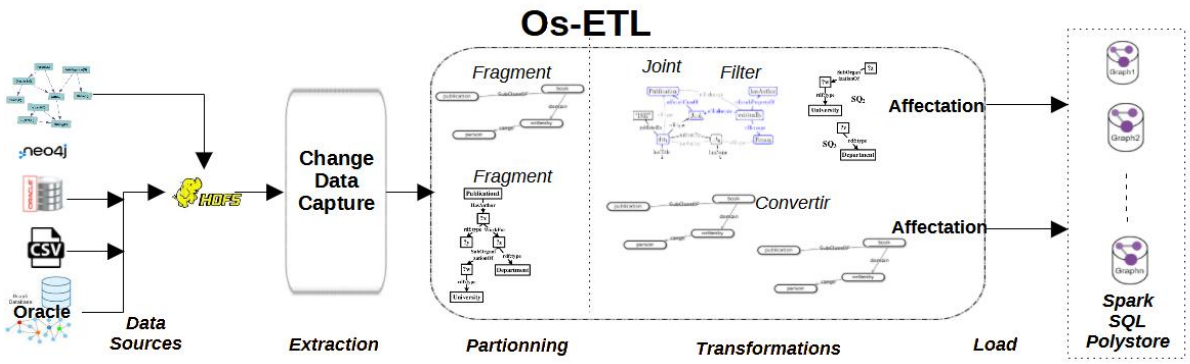

FIGURE 4.14 – *Architecture générale de l'OS-ETL.*

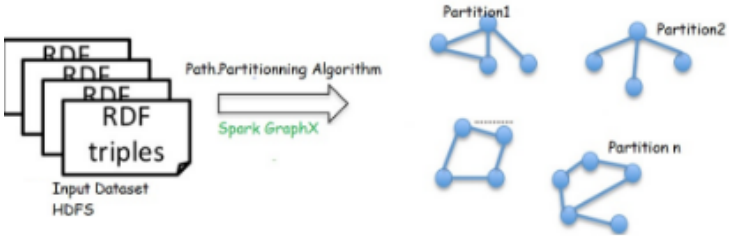

FIGURE 4.15 – *Partitionnement de chemin.*

Dans ce cas, l'alignement sur le modèle de données RDF a été employé. Ainsi, le problème de l'intégration de données hétérogènes avec des modèles de données hétérogènes dans des applications d'entrepôt de données est résolu. La solution Os-ETL est implémentée dans des scripts scala, basés sur le modèle d'exécution Spark.

4.3.3 L'algorithme Os-ETL

Dans l'architecture proposée, nous considérons le schéma de l'entrepôt de données comme prédéfini ; suivi des sources de données qui sont annotées dans l'outil protégé [106] en fonction des besoins exprimés dans l'application. Également à l'étape 1, l'horodatage de la dernière mise à jour (variable utilisée par la technique de capture de données de changement (CDC)), qui détecte et collecte les changements de données d'intérêt, nécessite des techniques connues sous le nom de CDC qui surveillent en permanence les sources de données opérationnelles [107]. La sortie de cette étape est représentée par un delta (qui représente toutes les modifications à apporter aux sources de données). Les dix-huit (18) opérateurs ETL conceptuels génériques définis dans [12] [89], ont été utilisés. L'algorithme 1 décrit les étapes du système Os-ETL proposé. Pour des besoins d'expériences, nous avons utilisé pour généré les fichiers N-Triple, une fonction écrite en scala qui transforme tous fichier RDF en format (nt), cette fonction est décrite dans l'algorithme 2.

Algorithme 4 : Algorithm 1 Os-ETL

Input : Relational database, CSV files, Flat files, Graph databases (NT files or OWL files) : all of the data source models are aligned to RDF data model, the schema of the data warehouse (DW) and the eighteen (18) conceptual ETL operators.
Output : Entrepôt de données deployé sur le polystore Spark SQL.

Begin

1. **Create** the control module CDC control module
2. **if** DW = empty **then**
   Repeat for each RDF data source
   Load changes into Temp
   Filter source by last updated timestamp
   Result (delta) to ETL engine (create and load a view)
   end if ;
3. **Built** the fragmentation by using Algorithm :P.P(Graph RDF, NBpartitions)
   Begin
   Load the N-Triple dataset and apply the path partition algorithm Specifies end-to-end paths
   Generate path groups for each starting vertex
   Merges a vertex based on the number of paths through that vertex
   end ;
4. **Make** the appropriate transformations
5. **ETL** Process and Deployment

End.

Algorithme 5 : Algorithm 2 fonction N-triple

Input : Fichier RDF.
Output : Fichier N-triples (sujet, predicat, object).

Begin

1. **Analyser** la représentation textuelle d'une ligne dans le fichier RDF
2. **Extraire** la représentation du sujet dans une RDD (RDD_sujet),
3. **Extraire** la représentation du prédicat dans une RDD (RDD_predicat),
4. **Extraire** la représentation de l'objet dans une RDD (RDD_objet),
5. **Créer** la classe N-triples (sujet : String, predicat : String, 'object' : String).

4.3.4 Déploiement sur un polystore

Selon [10], un système polystore est un système de base de données ayant divers magasins de données hétérogènes et diverses interfaces de requête. En raison de cette hétérogénéité des systèmes de stockage dans l'écosystème big data actuel, de nombreux backends ou moteurs de stockage fédérés différents sont nécessaires [9]. Le stockage de l'entrepôt de données est impacté par la variété et le polystore est bien adapté pour atteindre d'excellentes performances d'accès aux données [108]. Donc, nous affirmons que pour obtenir la valeur ajoutée, nous devons faire face à la variété. Dans notre approche, nous utilisons le polystore hybride Spark SQL pour déployer l'entrepôt de données cible.

4.3.5 Etude de Cas

Cette section décrit les outils et les logiciels utilisés dans ce travail ainsi qu'une étude de cas est considérée pour illustrer les principes sous-jacents du système Os-ETL proposé.

Ensemble de données

Afin de valider l'efficacité et l'efficience du système proposé, nous illustrons une étude de cas inspirée d'un projet de la direction de la formation du Ministère de l'Enseignement Supérieur et de la Recherche qui veut construire un entrepôt de données qui consiste à collecter des informations actionnables sur les étudiants universitaires, à savoir leur performance dans les publications.

Le benchmark LUBM est utilisé pour générer les schémas de source de données. Il génère des données de différentes universités, comme chaque université a un numéro Nous avons mené une série d'expérimentations à l'aide d'un schéma d'entrepôt de données, présenté à la

figure 4.16, issu du référentiel LUBM lié au domaine académique. L'outil UBA est utilisé pour générer des données sur l'ontologie Univ-Bench. Ces ensembles de données ont été utilisés pour créer des graphes simulés basés sur des sources de données externes. En tant que sources internes, nous avons également pris en compte des sources de données CSV et une base de données relationnelle. Les sources générées sont les suivantes :

1. Source 1 est une base de données de graphes Neo4j avec des nœuds, des arêtes : Person, Student(name), Publication, PublicationAuthor, University.
2. Source 2 est un fichier CSV composé d'attributs : Etudiant(nom), Publication, Université.
3. Source 3 est une base de données Oracle composée de tables et d'attributs : Etudiant(nom), Cours(titre), Université.

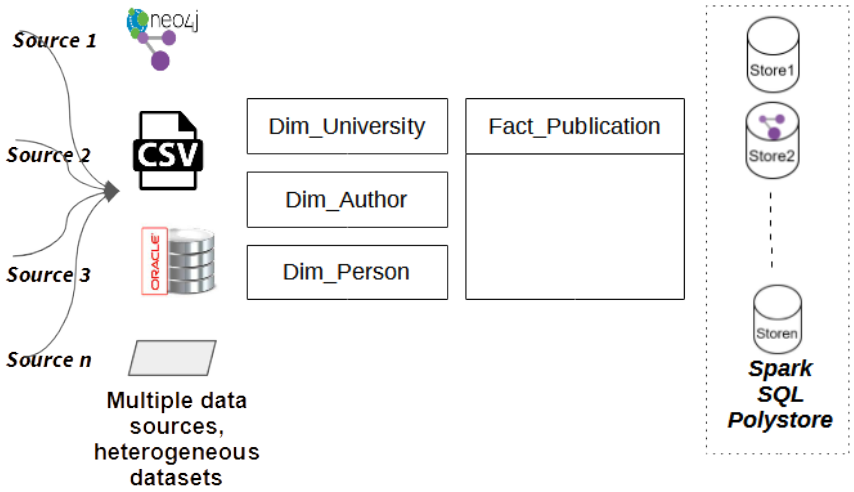

FIGURE 4.16 – *Entrepôt de données exemple.*

Nous avons appliqué l'algorithme Os-ETL et les scripts scala pour remplir le schéma d'entrepôt de données cible déployé sur le polystore Spark SQL.

Exemple d'exécution de l'algorithme Os-ETL sur un fichier CSV :

Les scripts Spark ETL importent des données CSV sur plusieurs nœuds. Spark commence l'exécution après avoir basculé les transformations de RDD à DAG. Un plan d'exécution logique est transformé en un plan d'exécution physique à l'aide d'un DAG. Le DAG est envoyé au planificateur DAG lorsqu'une activité est appelée. Il est divisé en sous-processus, qui consistent en une seule tâche. Le programme peut alors être déployé sur de nombreuses machines grâce à la transmission de ces jobs au gestionnaire de tâches via un gestionnaire de cluster. Le framework Spark gère le traitement du point de vue de la distribution. Chaque DataFrame, qu'il soit persistant ou non, est une collection partitionnée.

Le partitionnement et la distribution dans Spark du fichier csv est données à la figure 4.17

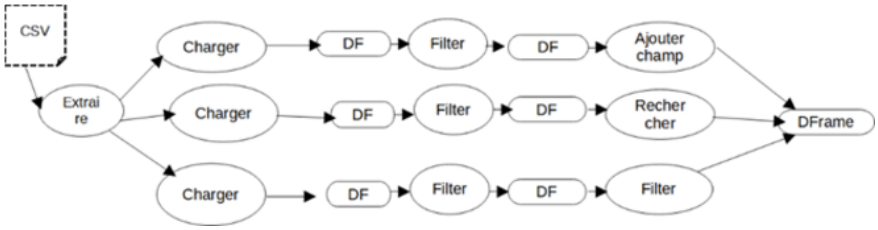

FIGURE 4.17 – *Partitionnement et distribution dans Spark.*

Métrique

Pour évaluer l'efficacité de tout système ETL, le temps de réponse est l'une des mesures importantes. Le temps de réponse après exécution de l'Os-ETL sur l'ensemble des données impliquées dans l'intégration. Une autre métrique est le temps de performance du système ETL, où la performance est mesurée en termes d'évolutivité. La taille des données factuelles est mise à l'échelle et le temps d'exécution pris pour chaque taille de données est collecté et tracé [11].

4.3.6 Expériences et Résultats

Dans cette section, nous présentons l'évaluation du système proposé sur la base de l'étude de cas ci-dessus. Ainsi, trois expériences différentes ont été menées. La première expérience est une comparaison de l'Os-ETL proposé avec une approche existante qui n'utilise pas de technique de partitionnement. La deuxième expérience est également une comparaison avec une approche existante qui utilise la technique de partitionnement. La troisième expérience considère l'évolutivité de notre solution. Le matériel utilisé pour exécuter les tests était un PC équipé d'un processeur Intel(R) Core(TM) i5-8350U (1,7 GHz), d'une mémoire principale de 8 Go sur un disque dur SSD de 500 Go. Le logiciel utilisé : Microsoft Windows 10x64 Professionnel, Spécifications logicielles : Oracle 12c, Apache Spark 2.4.4, scala 4.7, système de gestion de base de données graphique Neo4j et Eclipse IDE. Tous les outils logiciels ont été installés sur la machine locale à des fins d'évaluation.

Les schémas de source de données sont issus du benchmark LUBM. Nous utilisons des ensembles de données du monde réel : fichier CSV, base de données Oracle et la base de données Neo4j. À partir de cet ensemble, nous avons généré cinq ensembles de données RDF au format N-Triple, comme indiqué dans le tableau 4.1. L'entrepôt de données obtenu comporte deux magasins. Pour son déploiement, nous avons utilisé Oracle Database 12c comme

backend de base de données pour le magasin 1 et Neo4j Graph Database pour le magasin 2.

TABLE 4.1 – *Jeux de données RDF*

Concepts (University)	RDF (N-Triples)
3	21 057
6	42 115
9	63 173
12	84 231
15	105 289

Expérience 1.

Nous exécutons l'algorithme Os-ETL pour remplir le schéma d'entrepôt de données cible, et nous comparons le temps de réponse de notre méthode avec une étude précédente [45] (pas de stratégie de partitionnement). La figure 8 illustre les résultats obtenus, où le nombre d'instances est indiqué en milliers et les performances temporelles en millisecondes. D'après les résultats obtenus, il apparaît que notre l'Os-ETL proposé était le plus rentable car le temps de réponse de l'algorithme ETL était divisé par 4 (par exemple pour 35000 instances le temps de réponse était de 4000 ms) par rapport à l'approche de [66]. Les résultats illustrés sur la figure 4.18 montrent clairement que la stratégie de partitionnement améliore significativement le temps de réponse des processus ETL.

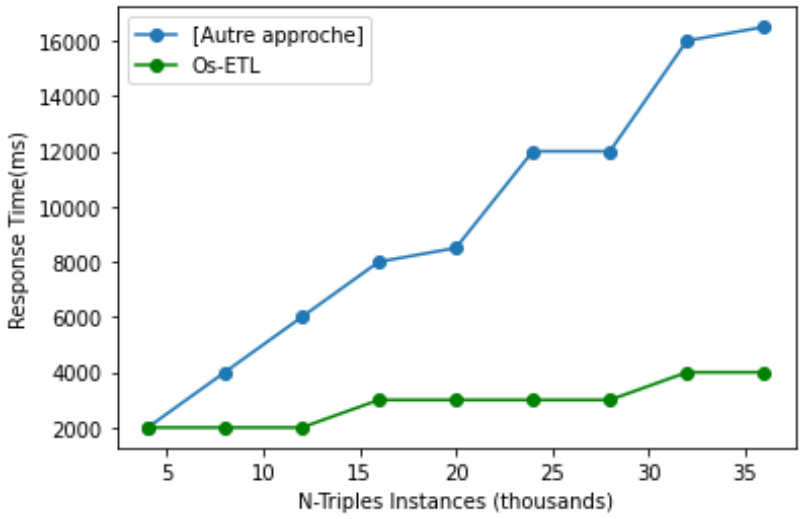

FIGURE 4.18 – *Performances de l'Os-ETL par rapport à [45] ETL.*

Expérience 2.

Nous exécutons l'algorithme Os-ETL pour remplir le schéma d'entrepôt de données cible déployé sur le polystore Spark SQL sur deux magasins (base de données Neo4j Graph et

base de données Oracle). Nous mesurons le temps passé à intégrer les instances dans chaque concept multidimensionnel. La figure 4.19 présente les résultats obtenus, où le nombre d'instances est indiqué en milliers et les performances temporelles en millisecondes. Nous comparons les performances de construction de l'entrepôt de données cible de l'Os-ETL par rapport à une autre étude précédente dans [11] (avec stratégie de partitionnement). Les résultats ont démontré que le système proposé est beaucoup plus efficace.

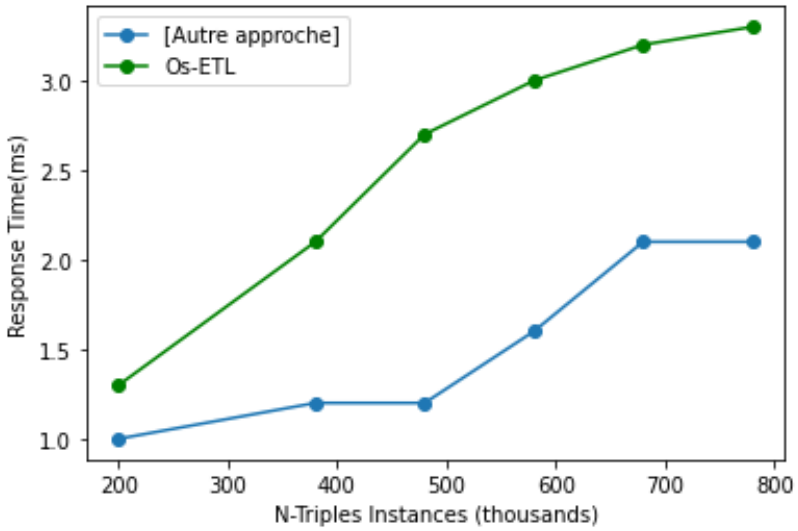

FIGURE 4.19 – *Performances de l'Os-ETL par rapport à [46] ETL.*

Expérience 3.

Nous avons considéré la scalabilité du système Os-ETL proposé en faisant varier la taille des sources de données et donc le nombre de tâches qui s'exécutent en parallèle sur le cluster Spark pour traiter les données. Nous mesurons les performances du système proposé en termes d'évolutivité. La taille des données de faits est mise à l'échelle de 20 à 120 Go. La figure 10 montre les résultats.

L'utilisation combinée des techniques : CDC, le partitionnement et le polystore, montre que les résultats illustrés sur la figure 4.20 montrent que les performances temporelles de notre système sont très bonnes par rapport à la taille du jeu de données qui augmente. Les résultats confirment qu'il s'agit d'un bon choix pour la stratégie de partitionnement et par conséquent, la présente solution confirme les résultats concernant l'évolutivité.

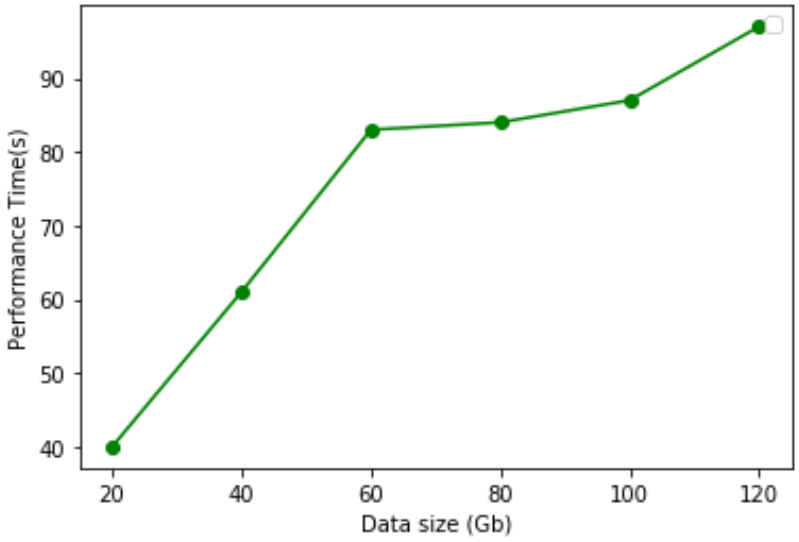

FIGURE 4.20 – *scalabilité de l'Os-ETL.*

4.3.7 Conclusion

Dans ce travail, nous avons proposé une nouvelle solution appelée Os-ETL qui consiste en six étapes : (i) extraire et transférer des données de flux ; (ii) aligner toutes les sources de données sur le modèle de données RDF ; (iii) partitionner les sources de données (iv) transformer les données RDF en données GraphX ; (v) transformations appropriées sur les données GraphX ; et (vi) allocation et distribution des fragments RDF obtenus sur le polystore Spark SQL. Tout d'abord, nous avons décrit les composants utilisés par Os-ETL qui extrait des sources de données hétérogènes impliquant des données Web en tant que données externes. Deux techniques, la méthode CDC et la stratégie de partitionnement ont été utilisées et analysées qui ont grandement amélioré le temps de réponse d'ETL.

En outre, nous avons proposé l'utilisation de systèmes de stockage polystore comme solution matérielle pour déployer l'entrepôt de données cible et les processus ETL. Le système Os-ETL améliore considérablement les processus ETL et permet aux développeurs de se concentrer sur la logique métier, plutôt que de se soucier du processus complexe d'extraction, de transformation et de chargement des données dans un environnement très varié. Les résultats les plus importants sont liés au temps de performance des processus ETL par rapport aux travaux précédents, grâce aux stratégies de partitionnement, en mémoire et en pipeline.

Les résultats obtenus confirment les meilleures performances du système Os-ETL proposé en termes de scalabilité et d'optimisation. Comme travaux futurs, le système proposé peut être étendu dans plusieurs directions. Le système peut être conçu pour prendre en charge d'autres types de sources de données hétérogènes telles que les données ouvertes liées (LOD). Le système peut être testé avec plusieurs benchmarks tels que TPC-DI.

4.4 AGRETL : un processus ETL évolutif pour l'aide à la décision dans l'agriculture à l'ère du big data à l'aide de Spark

4.4.1 Introduction

Selon le ministre algérien de l'agriculture et du développement rural invité sur radioalgerie : L'agriculture est un facteur important de l'économie de l'Algérie. Elle génère elle-même, sans les industries agroalimentaires, près de 14,7 % du produit intérieur brut (PIB) en 2022, mais avec des variations importantes selon les années en fonction des conditions climatiques. Le même ministre à dépêché les informations suivantes sur l'Algérie a produit, en 2018 :

— 4,6 millions de tonnes de pomme de terre (17e producteur mondial),
— 1,9 million de tonnes d'orge (18e producteur mondial),
— 3,9 millions de tonnes de blé (28e producteur mondial)...etc.

Donc l'agriculture algérienne est très diversifiée en termes de climat, de sol, de cultures, de cultures horticoles, de cultures de plantation, de ressources animales, de ressources halieutiques, de ressources en eau, etc.

Contexte de recherche

— Nos recherches concerne l'application des technologies big data dans le domaine de l'agriculture,
— L'objectif de ce travail est de concevoir un big entrepôt de données sur la distribution du stockage et de la vente au détail de l'agriculteur au marché. Le big entrepôt de données proposé peut être étendu au système d'aide à la décision (DSS) combiné avec des techniques d'exploration de données.
— Nous avons proposé un entrepôt de données dans l'écosystème big data pour l'agriculture qui apporte des solutions aux agriculteurs et responsables du domaines et qui répond à leurs demandes ponctuelles.

Le big data et l'Intelligence Artificielle peut jouer un rôle important dans l'agriculture en contribuant à la durabilité via des applications telles que la prévision des rendements, la protection des cultures et les systèmes d'aide à la décision. L'état de l'art démontre un potentiel énorme pour l'application de l'IA dans ce domaine. Pour l'algerie, on peut constater l'absence de nos chercheurs et institutions, ces derniers doivent collaborer davantage et former davantage de réseaux pour apporter des progrès radicaux dans le domaine. Dans ce travail, nous montrons comment les technologies du big data contribuent à atteindre l'objectif d'une agri-

culture durable et intelligente en Algérie. En outre, nous proposons un nouveau framework nommé AgroETLAnalytics pour l'intégration de données volumineuses appliquées à l'agriculture avec un modèle ETL efficace et une méthode de capture de données modifiées (CDC) conçue pour faire face à la responsabilité en temps quasi réel de nos jours. Le système de stockage est construit sur les systèmes de fichiers distribués Hadoop (HDFS), Apache Spark pour le traitement parallèle de grands ensembles de données et l'entrepôt de données Hive pour le déploiement ETL ont été utilisés. Le système proposé est évalué sur une étude de cas réelle concernant la distribution du stockage et de la vente au détail de l'agriculteur au marché. Le framework proposé est comparé à d'autres outils ETL. Selon les résultats expérimentaux, il peut accomplir des tâches fois plus rapidement.

Exemple : Guidés par les données agricoles, les auteurs de [109], ont conçu un framework d'analyse de données big data pour l'agriculture de précision. L'utilisation de la méthode big data a une certaine importance directrice pour l'intégration, l'extraction et l'utilisation de grandes quantités de données générées dans la production et l'application agricoles. L'architecture du modèle d'aide à la décision proposé est illustré à la figure 4.21.

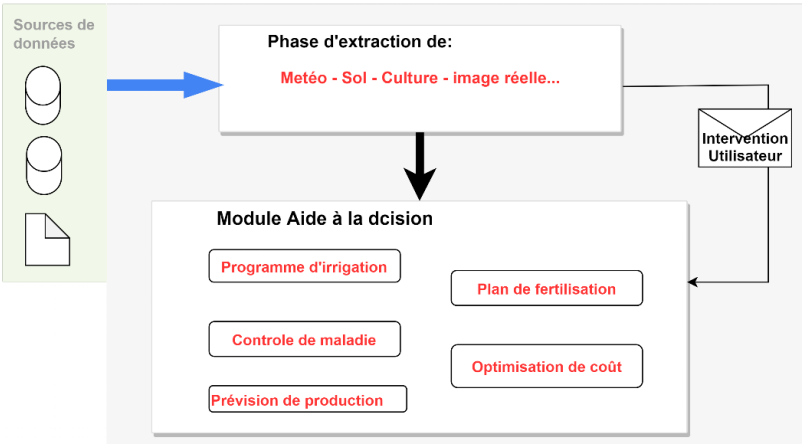

FIGURE 4.21 – *Architecture du modèle d'aide à la décision pour l'agriculture.*

4.4.2 Le nouveau framework

En un mot, atteindre une latence quasi nulle entre les systèmes OLTP et OLAP implique d'assurer une intégration continue des données du premier dans le second. Tout d'abord, nous proposons de créer un nouveau cadre générique qui pourrait convenir à une variété de situations d'ingestion de données. Cette architecture générique est illustrée à la Figure 11. Nous considérons quatre composants principaux : l'extraction des données récemment modifiées (méthode des déclencheurs CDC), le moteur Spark ETL, et l'entrepôt de données Hive.

Les architectures modernes d'intégration de données doivent fournir une gestion intégrée pour gérer plusieurs systèmes de stockage cibles [9]. Les architectures modernes d'intégration de données doivent fournir une gestion intégrée pour gérer plusieurs systèmes de stockage cibles [9].

Pour une intégration de données en temps réel, qui identifie et capture les modifications apportées aux sources de données et fournit uniquement les données modifiées au système. Nous implémentons dans notre approche la méthode CDC basée sur les déclencheurs [110]. Notre solution prévoit un déploiement sur l'entrepôt de données hive.

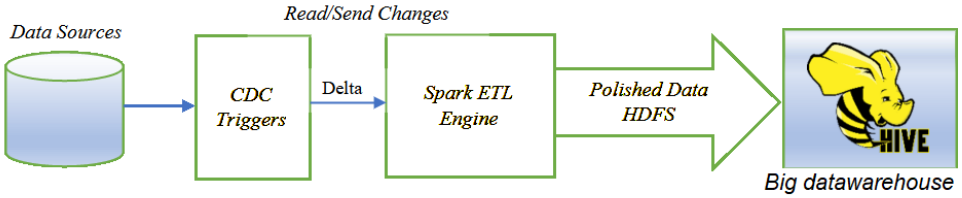

FIGURE 4.22 – *l'Architecture générale du framework AGRETL.*

4.4.3 Expériences et Résultats

La technologie du big data offre une nouvelle solution pour traiter de grandes quantités de données hétérogènes et de données non structurées [111]. Hadoop est une infrastructure de système distribué de l'organisation open source Apache. Le composant principal, HDFS (Hadoop Distributed File System), fournit une prise en charge du stockage unifié pour les données massives, structurées et non structurées. Les outils ETL aident à gérer l'analyse de l'entrepôt de données et seront largement utilisés dans les applications big data. Dans cette section, nous fournissons les résultats expérimentaux. Dans un premier temps, nous comparons notre système avec l'outil Talend. Deuxièmement, nous démontrons l'avantage du chargement incrémental par rapport au rechargement complet, et enfin nous mesurons les performances du système proposé.

À des fins d'évaluation, tous les outils logiciels sélectionnés ont été déployés sur la machine locale. Les spécifications matérielles de la machine et la description du logiciel sont données ci-dessous.

L'intégration de données consiste à centraliser des données provenant de différentes sources et des données hétérogènes afin de fournir une base de données complète pour l'analyse des données. L'extraction de données à partir de différentes bases de données ou tables

de données consiste à former un entrepôt de données sujet relatif à un certain domaine. Dans notre cas c'est le domaine agricole.

Sources de données

Une grande quantité de données est requise par l'agriculture de précision pour servir le système d'aide à la décision. Habituellement, les données peuvent être structurées, semi-structurées et non structurées. Les sources de données incluent les informations des capteurs IoT (Internet de Objets), les enregistrements de documents, les informations SIG Web (Système d'information géographique), etc.

Données du réseau de Capteur : La technologie Internet des objets relie les objets physiques du monde réel au monde numérique. Les capteurs mesurent les caractéristiques du monde naturel, quantifient les informations de caractérisation et les envoient à l'extrémité de réception des informations. Par exemple, les capteurs déployés dans les terres agricoles peuvent mesurer la valeur du PH, l'humidité, les précipitations et la température de la terre en temps réel. Ces enregistrements de données peuvent devenir la clé du contrôle du système [112]. Les données collectées par les capteurs constituent la partie la plus importante des mégadonnées agricoles et la principale source de données de notre système proposé.

Pour les capteurs il faut bien choisir le type et la marque. Il existe des méthodes différentes des fabricants de capteurs ce qui va entraîner des incohérences dans la collecte des données. Ignorer l'incohérence peut entraîner des erreurs et des erreurs dans la prise de décision. Par exemple, les précipitations collectées peuvent être mesurées par heure ou par jour, et les résultats obtenus sont complètement différents.

Données Historiques : Les données historiques concernent les journaux de terres agricoles, l'ensemencement, la fertilisation, l'insecticide, la récolte, le stockage, l'élevage, etc., ainsi que les statistiques des industries et des institutions.

Données Spatiales :

Parmi les nombreuses sources de données du big data agricole, les données spatiales ont un statut particulier. Données de télédétection par satellite, données cartographiques, données d'enquête sur les ressources, données de zonage agricole, etc.

Dans le développement actuel de l'agriculture, l'application de la technologie SIG peut atteindre efficacement les objectifs de développement de l'agriculture. Le SIG aide à organiser et à gérer les données spatiales agricoles. L'utilisation de cartes comme plate-forme pour afficher

des informations statistiques agricoles et des données en temps réel de manière visuelle est un moyen important d'améliorer la lisibilité de l'analyse des mégadonnées.

Ensemble de données

Notre jeu de données est extrait de Kaggle [46] et contient initialement 500 000 tuples insérés pour la première fois dans la base de données. Les données sont collectées depuis 2000 et offrent une bonne opportunité pour mettre en œuvre des expériences basées sur la technologie Spark. Ensuite, des ensembles de données sont créés de manière aléatoire à partir du pourcentage de données modifiées. Tout d'abord, nous devons intégrer les données de plusieurs systèmes au système de fichiers HDFS. En exécutant les commandes dans le terminal du cluster Spark, nous importons les données nécessaires directement dans l'entrepôt de données Hive, qui seront organisées sous forme de tables, telles qu'elles étaient stockées dans la base de données importée. Nous menons plusieurs expériences pour évaluer notre proposition. Ensuite, des ensembles de données sont créés de manière aléatoire à partir du pourcentage de données modifiées. Tout d'abord, nous devons intégrer les données de plusieurs systèmes au système de fichiers HDFS.

En exécutant les commandes dans le terminal du cluster Spark, nous importons les données nécessaires directement dans l'entrepôt de données Hive, qui seront organisées sous forme de tables, telles qu'elles étaient stockées dans la base de données importée. Nous menons plusieurs expériences pour évaluer notre proposition.

La figure 4.23 illustre notre constatation selon laquelle la stratégie CDC améliore considérablement le temps de réponse du processus ETL. Dans une deuxième expérience, nous essayons d'évaluer les performances en comparant le temps d'exécution des processus ETL de notre approche par rapport à une autre approche dans laquelle nous avons utilisé le système d'intégration Talend Open Studio. On observe que notre modèle est beaucoup plus efficace comme le montre la figure 4.24.

Alimentation de l'entrepôt Aprés extraction, les données vont dans un premier temps passer par nos scripts scala : elles sont sélectionnées, ramenées vers le systèmes de fichier HDFS et nettoyées afin d'être homogénéisées en vue d'être intégrées à l'entrepôt. Notre algorithme permet la connexion direct au sources de données existante.

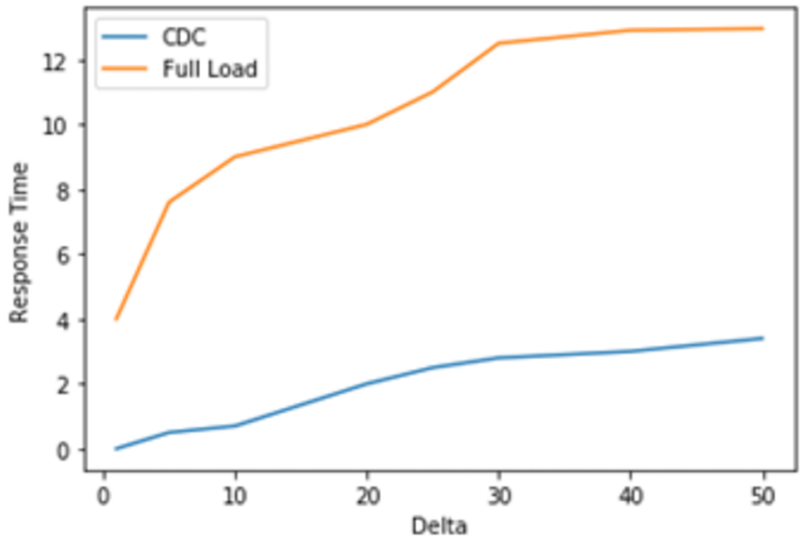

FIGURE 4.23 – *Méthode CDC Vs Chargement complet.*

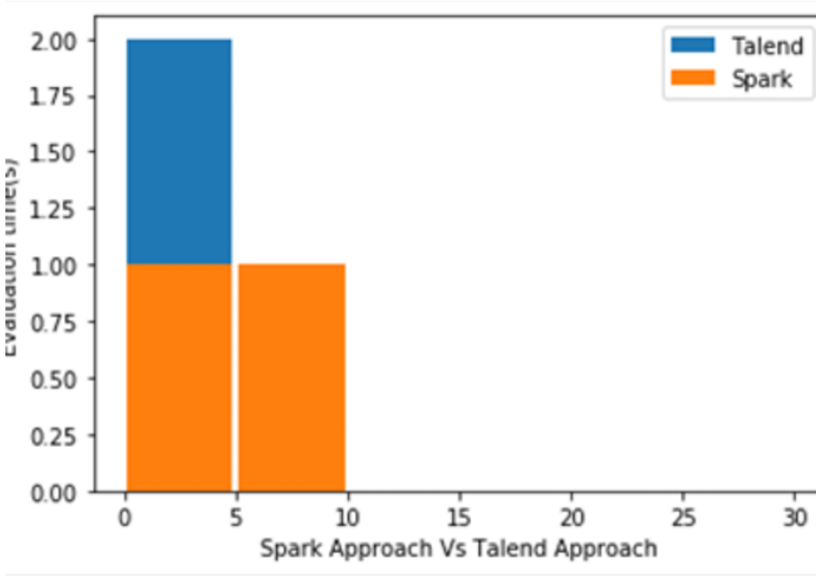

FIGURE 4.24 – *Performances temporelles du processus ETL.*

4.4.4 Conclusion

Les technologies d'entrepôts de données/ big data peuvent être utilisées pour obtenir un grand nombre de données sur les activités du secteur agricole, traiter ces données et générer des informations précieuses pour les agriculteurs, les vendeurs, les consommateurs et les institutions gouvernementales à partir de données volumineuses. Pour être précis, l'agriculture big data et l'avantage d'une solution warehousing va aider à : (i) détenir et conserver les produits depuis leur production jusqu'à ce qu'ils soient nécessaires à la consommation ; (ii) assure un flux continu de marchandises sur le marché ; (iii) protège la qualité des produits périssables et semi-périssables contre la détérioration ; (vi) certains des produits, ont une demande saisonnière ; (v) pour faire face à cette demande, la production en continu et le stockage deviennent

nécessaires ; (vi) il contribue à la stabilisation des prix en ajustant l'offre et la demande ; et enfin (vii) le stockage fournit des emplois et des revenus grâce à des avantages de prix.

Conclusion du chapitre

Dans ce chapitre nous avons présenté les différentes contributions réalisées dans cette thèse, nous avons proposé de nouveaux systèmes architecture d'intégration de données ETL. Pour valider nos propositions, nous avons développé des scripts s'exécutants dans un environnement distribué. L'étude expérimentale est basée sur des données et des modèles de benchmarks. Les résultats obtenus démontrent la faisabilité et la pertinence de notre approche en termes de performance et d'optimisation.

Conclusion générale

L'extraction, la transformation et le chargement de données (ETL) sont des processus populaire utilisés dans l'intégration de données, c'est-à-dire la collecte de données provenant de diverses sources opérationnelles dans l'entrepôt de données. Dans notre thèse, nous apportons une contribution au domaine de recherche lié a ces processus et par conséquent à la business intelligence et l'analyse de données en proposant des méthodes novatrices pour relever trois défis liés à l'exploitation des sources de données sans schéma qui doivent être extraites et transformées pour répondre aux besoins de décision ; des sources de données volumineuses, dispersées et non connues à l'avance des décideurs et des systèmes de stockages multi-magasins tels que le polystore pouvant stocker différents schémas de données qui fournissent un accès intégré aux données. Il y a plusieurs défis pour lesquels nous avons proposé une nouvelle approche pour y faire face :

1. La diversité syntaxique et sémantique des sources, qui découle de la variété des structures et des formats utilisés pour stocker les informations. L'hétérogénéité sémantique découle également des multiples interprétations des éléments du monde réel,
2. Le manque de méthode de conception standard pour les processus ETL,
3. D'un autre côté, de nouvelles applications ont émergé et ces applications doivent collecter des données opérationnelles à partir de sources opérationnelles en temps réel, telles que les systèmes de soins de santé et les systèmes qui traitent les données des capteurs. Par conséquent le besoin d'architectures pour créer un entrepôt de données avec des capacités ETL en temps réel.
4. À l'ère du big bata, les entreprises subissent une pression croissante pour stocker et analyser toutes les données collectées afin de rester compétitives sur le marché axé sur les données. Par conséquent, la nécessité de créer un framework général et rentable pour l'ETL dans l'environnement big data et couvrant plusieurs défis.
5. L'émergence des nouvelles plateformes de déploiment d'entrepôt de données tel que le polystore.

Perspectives

Dans la continuité directe de notre travail de thèse, certaines rubriques méritent encore d'être explorées dans notre travail : Nous avons l'intention d'étendre nos approches pour prendre en charge des modèles de données NoSQL. Concernant notre première contribution, nous avons l'intention de rendre automatique le passage modèles BPMN vers scripts scala et d'étudier l'impact sur le rafraîchissement incrémental des données. Nous avons aussi l'intention d'intégrer notre deuxième approche en tant que composant ETL dans un outil ETL commercial, par exemple Talend Studio for Big Data avec l'utilisation d'autres systèmes polystore tels que BigDAWG ou QOX.

Bibliographie

[1] A. Spark, "Apache spark," *Retrieved January*, vol. 17, no. 1, p. 2018, 2018. 2

[2] J. Dean and S. Ghemawat, "Mapreduce : simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008. 2

[3] J. Awiti, "Algorithms and architecture for managing evolving etl workflows," in *New Trends in Databases and Information Systems : ADBIS 2019 Short Papers, Workshops BBIGAP, QAUCA, SemBDM, SIMPDA, M2P, MADEISD, and Doctoral Consortium, Bled, Slovenia, September 8–11, 2019, Proceedings 23*, pp. 539–545, Springer, 2019. 2

[4] S. M. F. Ali and R. Wrembel, "From conceptual design to performance optimization of etl workflows : current state of research and open problems," *The VLDB Journal*, vol. 26, no. 6, pp. 777–801, 2017. 2, 56, 77

[5] S. H. A. El-Sappagh, A. M. A. Hendawi, and A. H. El Bastawissy, "A proposed model for data warehouse etl processes," *Journal of King Saud University-Computer and Information Sciences*, vol. 23, no. 2, pp. 91–104, 2011. 2

[6] N. Biswas and K. C. Mondal, "Integration of etl in cloud using spark for streaming data," in *Advanced Techniques for IoT Applications : Proceedings of EAIT 2020*, pp. 172–182, Springer, 2022. 2, 3, 58, 61

[7] E. Zdravevski, P. Lameski, A. Dimitrievski, M. Grzegorowski, and C. Apanowicz, "Cluster-size optimization within a cloud-based etl framework for big data," in *2019 IEEE International Conference on Big Data (Big Data)*, pp. 3754–3763, IEEE, 2019. 2, 3

[8] I. P. M. Atmaja, A. Saptawijaya, S. Aminah, *et al.*, "Implementation of change data capture in etl process for data warehouse using hdfs and apache spark," in *2017 International Workshop on Big Data and Information Security (IWBIS)*, pp. 49–55, IEEE, 2017. 2, 3

[9] J. Meehan, C. Aslantas, S. Zdonik, N. Tatbul, and J. Du, "Data ingestion for the connected world.," in *CIDR*, vol. 17, pp. 8–11, 2017. 2, 55, 64, 65, 66, 67, 86, 91, 99

[10] S. Bimonte, E. Gallinucci, P. Marcel, and S. Rizzi, "Data variety, come as you are in multi-model data warehouses," *Information Systems*, vol. 104, p. 101734, 2022. 2, 58, 64, 65, 66, 67, 91

[11] N. Berkani and L. Bellatreche, "Streaming etl in polystore era," in *Algorithms and Architectures for Parallel Processing : 18th International Conference, ICA3PP 2018, Guangzhou, China, November 15-17, 2018, Proceedings, Part III*, pp. 560–574, Springer, 2018. 3, 64, 65, 66, 67, 93, 95

[12] D. Skoutas and A. Simitsis, "Designing etl processes using semantic web technologies," in *Proceedings of the 9th ACM international workshop on Data warehousing and OLAP*, pp. 67–74, 2006. 4, 44, 51, 64, 65, 66, 67, 90

[13] A. Vaisman and E. Zimányi, "Data warehouse systems," *Data-Centric Systems and Applications*, 2014. 9

[14] S. Negash, "Business intelligence," *Communications of the association for information systems*, vol. 13, no. 1, p. 15, 2004. 9

[15] W. H. Inmon, *Building the data warehouse*. John wiley & sons, 2005. 10

[16] M. Ross and R. Kimball, *The data warehouse toolkit : the definitive guide to dimensional modeling*. John Wiley & Sons, 2013. 10

[17] MI, "A market intelligence and advisory firm.." 12

[18] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, and P. C. Arocena, "Data lake management : challenges and opportunities," *Proceedings of the VLDB Endowment*, vol. 12, no. 12, pp. 1986–1989, 2019. 13

[19] P. Vassiliadis and A. Simitsis, "Extraction, transformation, and loading.," *Encyclopedia of Database Systems*, vol. 10, 2009. 14, 76

[20] J. Lu and I. Holubová, "Multi-model databases : a new journey to handle the variety of data," *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 1–38, 2019. 16

[21] J. C. Nwokeji and R. Matovu, "A systematic literature review on big data extraction, transformation and loading (etl)," in *Intelligent Computing : Proceedings of the 2021 Computing Conference, Volume 2*, pp. 308–324, Springer, 2021. 17

[22] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, "Conceptual modeling for etl processes," in *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, pp. 14–21, 2002. 17, 18, 77

[23] A. Dhaouadi, K. Bousselmi, M. M. Gammoudi, S. Monnet, and S. Hammoudi, "Data warehousing process modeling from classical approaches to new trends : Main features and comparisons," *Data*, vol. 7, no. 8, p. 113, 2022. 18, 57

[24] J.-N. Mazón and J. Trujillo, "An mda approach for the development of data warehouses," *Decision Support Systems*, vol. 45, no. 1, pp. 41–58, 2008. 18

[25] B. Oliveira and O. Belo, "From etl conceptual design to etl physical sketching using patterns," 2018. 18

[26] D. Silva, J. M. Fernandes, and O. Belo, "Assisting data warehousing populating processes design through modelling using coloured petri nets," 2013. 18

[27] O. Belo, A. Cuzzocrea, and B. Oliveira, "Modeling and supporting etl processes via a pattern-oriented, task-reusable framework," in *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*, pp. 960–966, IEEE, 2014. 18

[28] J. Awiti, A. A. Vaisman, and E. Zimányi, "Design and implementation of etl processes using bpmn and relational algebra," *Data & Knowledge Engineering*, vol. 129, p. 101837, 2020. 20, 54, 64, 65, 66, 67

[29] Z. El Akkaoui, E. Zimányi, J.-N. Mazón, and J. Trujillo, "A model-driven framework for etl process development," in *Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP*, pp. 45–52, 2011. 20

[30] MOF., "Meta-object facilities (mof).." 20

[31] M. Souibgui, F. Atigui, S. B. Yahia, and S. Si-Said Cherfi, "Business intelligence and analytics : On-demand etl over document stores," in *Research Challenges in Information Science : 14th International Conference, RCIS 2020, Limassol, Cyprus, September 23–25, 2020, Proceedings 14*, pp. 556–561, Springer, 2020. 22

[32] H. V. Inc., "Pentaho data integration," 2018. 26, 54, 81

[33] I. Cloud Software Group, "Jaspersoft," 2018. 26

[34] A. Reeve, *Managing data in motion : data integration best practice techniques and technologies*. Newnes, 2013. 27

[35] D. Q. Tu, A. Kayes, W. Rahayu, and K. Nguyen, "Iot streaming data integration from multiple sources," *Computing*, vol. 102, pp. 2299–2329, 2020. 28, 57, 64, 65, 66, 67

[36] A. Fuxman, E. Fazli, and R. J. Miller, "Conquer : Efficient management of inconsistent databases," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pp. 155–166, 2005. 29

[37] E. Zdravevski, P. Lameski, C. Apanowicz, and D. Ślzak, "From big data to business analytics : The case study of churn prediction," *Applied Soft Computing*, vol. 90, p. 106164, 2020. 29, 56, 86

[38] A. Panimalar, V. Shree, and V. Kathrine, "The 17 v's of big data," *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, no. 9, pp. 3–6, 2017. 30

[39] D. Borthakur, "The hadoop distributed file system : Architecture and design," *Hadoop Project Website*, vol. 11, no. 2007, p. 21, 2007. 32

[40] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pp. 29–43, 2003. 32

[41] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, *et al.*, "Apache spark : a unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016. 34, 37

[42] M. Parsian, *Data algorithms : Recipes for scaling up with hadoop and spark*. " O'Reilly Media, Inc.", 2015. 35

[43] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia, *Learning spark : lightning-fast big data analysis*. " O'Reilly Media, Inc.", 2015. 35

[44] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, *et al.*, "Spark : Cluster computing with working sets.," *HotCloud*, vol. 10, no. 10-10, p. 95, 2010. 37

[45] A. Alexander, "Scala cookbook : Recipes for object-oriented and functional programming," 2013. 38

[46] "Kaggle, inc." https://www.kaggle.com, 2023. 38, 101

[47] W3C., "Resource description framework.." 42

[48] Y. Guo, Z. Pan, and J. Heflin, "Lubm : A benchmark for owl knowledge base systems," *Journal of Web Semantics*, vol. 3, no. 2-3, pp. 158–182, 2005. 42

[49] W3C., "Resource description framework schema.," 2020. 42

[50] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of databases*, vol. 8. Addison-Wesley Reading, 1995. 42

[51] Neo4j., "Graph database.." 43

[52] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, *et al.*, "Spark sql : Relational data processing in spark," in *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pp. 1383–1394, 2015. 45, 47, 48, 78

[53] A. P. Sheth and J. A. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases," *ACM Computing Surveys (CSUR)*, vol. 22, no. 3, pp. 183–236, 1990. 45

[54] C. Bondiombouy and P. Valduriez, "Query processing in multistore systems : an overview," *International Journal of Cloud Computing*, vol. 5, no. 4, pp. 309–346, 2016. 45, 46, 78

[55] A. Simitsis, K. Wilkinson, M. Castellanos, and U. Dayal, "Optimizing analytic data flows for multiple execution engines," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 829–840, 2012. 47, 48

[56] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, and S. Zdonik, "The bigdawg polystore system," *ACM Sigmod Record*, vol. 44, no. 2, pp. 11–16, 2015. 47, 48, 56, 75

[57] J. Wang, T. Baker, M. Balazinska, D. Halperin, B. Haynes, B. Howe, D. Hutchison, S. Jain, R. Maas, P. Mehta, *et al.*, "The myria big data management and analytics system and cloud services.," in *CIDR*, 2017. 47, 48

[58] W. H. Inmon, D. Strauss, and G. Neushloss, *DW 2.0 : The architecture for the next generation of data warehousing*. Elsevier, 2010. 50

[59] D. Skoutas, A. Simitsis, and T. Sellis, "Ontology-driven conceptual design of etl processes using graph transformations," *Journal on data semantics XIII*, pp. 120–146, 2009. 51

[60] S. Bergamaschi, F. Guerra, M. Orsini, C. Sartori, and M. Vincini, "A semantic approach to etl technologies," *Data & Knowledge Engineering*, vol. 70, no. 8, pp. 717–731, 2011. 51, 64, 65, 66, 67

[61] C. Fellbaum, "Wordnet," in *Theory and applications of ontology : computer applications*, pp. 231–243, Springer, 2010. 51

[62] D. Skoutas and A. Simitsis, "Ontology-based conceptual design of etl processes for both structured and semi-structured data," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 3, no. 4, pp. 1–24, 2007. 51, 64, 65, 66, 67, 77

[63] A. Simitsis, P. Vassiliadis, and T. Sellis, "Optimizing etl processes in data warehouses," in *21st International Conference on Data Engineering (ICDE'05)*, pp. 564–575, Ieee, 2005. 52, 61

[64] O. Romero, A. Simitsis, and A. Abelló, "Gem : Requirement-driven generation of etl and multidimensional conceptual designs," in *Data Warehousing and Knowledge Discovery : 13th International Conference, DaWaK 2011, Toulouse, France, August 29-September 2, 2011. Proceedings 13*, pp. 80–95, Springer, 2011. 52, 64, 65, 66, 67

[65] TPC., "A decision support benchmark.," 2020. 52

[66] N. Berkani, L. Bellatreche, and S. Khouri, "Towards a conceptualization of etl and physical storage of semantic data warehouses as a service," *Cluster computing*, vol. 16, no. 4, pp. 915–931, 2013. 52, 64, 65, 66, 67, 94

[67] S. K. Bansal, "Towards a semantic extract-transform-load (etl) framework for big data integration," in *2014 IEEE International Congress on Big Data*, pp. 522–529, IEEE, 2014. 53, 64, 65, 66, 67

[68] N. Berkani, L. Bellatreche, and L. Guittet, "Etl processes in the era of variety," *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXIX : Special Issue on Database-and Expert-Systems Applications*, pp. 98–129, 2018. 54, 64, 65, 66, 67, 86

[69] M. Poess, T. Rabl, H.-A. Jacobsen, and B. Caufield, "Tpc-di : the first industry benchmark for data integration," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1367–1378, 2014. 54

[70] Talend., "Talend open studio for data integration.," 2020. 54, 61

[71] J. J. K. Behan, O. Romero, and E. Zimányi, "Multidimensional integration of rdf datasets," in *Big Data Analytics and Knowledge Discovery : 21st International Conference, DaWaK 2019, Linz, Austria, August 26–29, 2019, Proceedings 21*, pp. 119–135, Springer, 2019. 55, 64, 65, 66, 67

[72] L. Etcheverry and A. A. Vaisman, "Qb4olap : a new vocabulary for olap cubes on the semantic web," in *Proceedings of the Third International Conference on Consuming Linked Data*, vol. 905, pp. 27–38, CEUR-WS. org, 2012. 55

[73] N. Garg, *Learning Apache Kafka*. Packt Publishing, 2015. 55

[74] J. Meehan, N. Tatbul, S. Zdonik, C. Aslantas, U. Cetintemel, J. Du, T. Kraska, S. Madden, D. Maier, A. Pavlo, *et al.*, "S-store : Streaming meets transaction processing," *arXiv preprint arXiv :1503.01143*, 2015. 55

[75] F. d. A. Vilela and R. R. Ciferri, "A survey of real-time etl process applied to data warehousing environments," in *ITNG 2022 19th International Conference on Information Technology-New Generations*, pp. 91–96, Springer, 2022. 56

[76] Infobright., "a high-performance enterprise database solution.," 2020. 56

[77] A. Abelló, O. Romero, T. B. Pedersen, R. Berlanga, V. Nebot, M. J. Aramburu, and A. Simitsis, "Using semantic web technologies for exploratory olap : a survey," *IEEE transactions on knowledge and data engineering*, vol. 27, no. 2, pp. 571–588, 2014. 56

[78] S. Oni, K. Pansare, S. S. Arneja, Z. Chen, A. Crainiceanu, and D. Needham, "Rdfint : A benchmark for comparing data warehouse with virtual integration approaches for integration of rdf data," in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 2820–2826, IEEE, 2020. 56, 64, 65, 66, 67

[79] C. Thomsen and T. Bach Pedersen, "pygrametl : A powerful programming framework for extract-transform-load programmers," in *Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP*, pp. 49–56, 2009. 57, 62, 64, 65, 66, 67

[80] E. Mehmood and T. Anees, "Distributed real-time etl architecture for unstructured big data," *Knowledge and Information Systems*, vol. 64, no. 12, pp. 3419–3445, 2022. 57

[81] M. Patel and D. B. Patel, "Data warehouse modernization using document-oriented etl framework for real time analytics," in *Rising Threats in Expert Applications and Solutions : Proceedings of FICR-TEAS 2022*, pp. 33–41, Springer, 2022. 58

[82] A. Pareek, B. Khaladkar, R. Sen, B. Onat, V. Nadimpalli, and M. Lakshminarayanan, "Real-time etl in striim," in *Proceedings of the international workshop on real-time business intelligence and analytics*, pp. 1–10, 2018. 59, 61

[83] N. V. Do, T. T. Mai, and L. N. Hoang, "A knowledge-based model for designing the knowledge querying system in education," in *2022 RIVF International Conference on Computing and Communication Technologies (RIVF)*, pp. 524–529, IEEE, 2022. 59

[84] S. Mhammedi and N. Gherabi, "Heterogeneous integration of big data using semantic web technologies," in *Intelligent Systems in Big Data, Semantic Web and Machine Learning*, pp. 167–177, Springer, 2021. 59

[85] X. Liu, C. Thomsen, and T. B. Pedersen, "Mapreduce-based dimensional etl made easy," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1882–1885, 2012. 59, 61

[86] X. Liu and N. Iftikhar, "An etl optimization framework using partitioning and parallelization," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pp. 1015–1022, 2015. 60, 61

[87] M. Stonebraker, D. Abadi, D. J. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin, "Mapreduce and parallel dbmss : friends or foes ?," *Communications of the ACM*, vol. 53, no. 1, pp. 64–71, 2010. 61, 86

[88] V. Nebot and R. Berlanga, "Building data warehouses with semantic web data," *Decision Support Systems*, vol. 52, no. 4, pp. 853–868, 2012. 64, 65, 66, 67

[89] N. Berkani, L. Bellatreche, and B. Benatallah, "A value-added approach to design bi applications," in *Big Data Analytics and Knowledge Discovery : 18th International Conference, DaWaK 2016, Porto, Portugal, September 6-8, 2016, Proceedings*, pp. 361–375, Springer, 2016. 64, 65, 66, 67, 76, 90

[90] N. Biswas, A. Sarkar, and K. C. Mondal, "Efficient incremental loading in etl processing for real-time data integration," *Innovations in Systems and Software Engineering*, vol. 16, pp. 53–61, 2020. 66

[91] N. Biswas, S. Chattapadhyay, G. Mahapatra, S. Chatterjee, and K. C. Mondal, "A new approach for conceptual extraction-transformation-loading process modeling," *International Journal of Ambient Computing and Intelligence (IJACI)*, vol. 10, no. 1, pp. 30–45, 2019. 67

[92] H. Nazeer, W. Iqbal, F. Bokhari, F. Bukhari, and S. U. R. Baig, "Real-time text analytics pipeline using open-source big data tools," *arXiv preprint arXiv :1712.04344*, 2017. 74

[93] C. R. Valêncio, M. H. Marioto, G. F. D. Zafalon, J. M. Machado, and J. C. Momente, "Real time delta extraction based on triggers to support data warehousing," in *2013*

International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 293–297, Ieee, 2013. 76

[94] P. Vassiliadis, A. Simitsis, P. Georgantas, M. Terrovitis, and S. Skiadopoulos, "A generic and customizable framework for the design of etl scenarios," *Information Systems*, vol. 30, no. 7, pp. 492–525, 2005. 77

[95] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, "Modeling etl activities as graphs.," in *DMDW*, vol. 58, pp. 52–61, 2002. 77

[96] C. Inc., "Camunda modeler," 2020. 77

[97] R. Tan, R. Chirkova, V. Gadepally, and T. G. Mattson, "Enabling query processing across heterogeneous data models : A survey," in *2017 IEEE International Conference on Big Data (Big Data)*, pp. 3211–3220, IEEE, 2017. 78

[98] P. Chen, V. Gadepally, and M. Stonebraker, "The bigdawg monitoring framework," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6, IEEE, 2016. 78

[99] J. Du, J. Meehan, N. Tatbul, and S. Zdonik, "Towards dynamic data placement for polystore ingestion," in *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics*, pp. 1–8, 2017. 86

[100] A. Longo, S. Giacovelli, and M. A. Bochicchio, "Fact–centered etl : A proposal for speeding business analytics up," *Procedia Technology*, vol. 16, pp. 471–480, 2014. 87

[101] M. F. Masouleh, M. A. Kazemi, M. Alborzi, and A. T. Eshlaghy, "Optimization of etl process in data warehouse through a combination of parallelization and shared cache memory," *Engineering, Technology & Applied Science Research*, vol. 6, no. 6, pp. 1241–1244, 2016. 87

[102] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on apache spark," *International Journal of Data Science and Analytics*, vol. 1, pp. 145–164, 2016. 87

[103] B. Wu, Y. Zhou, P. Yuan, L. Liu, and H. Jin, "Scalable sparql querying using path partitioning," in *2015 IEEE 31st International Conference on Data Engineering*, pp. 795–806, IEEE, 2015. 88

[104] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, "Graphx : Graph processing in a distributed dataflow framework," in *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pp. 599–613, 2014. 88

[105] I. Boukhari, L. Bellatreche, and S. Jean, "An ontological pivot model to interoperate heterogeneous user requirements," in *Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies : 5th International Symposium, ISoLA 2012, Heraklion, Crete, Greece, October 15-18, 2012, Proceedings, Part II 5*, pp. 344–358, Citeseer, 2012. 88

[106] H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen, "The protégé owl plugin : An open development environment for semantic web applications," in *The Semantic Web–ISWC 2004 : Third International Semantic Web Conference, Hiroshima, Japan, November 7-11, 2004. Proceedings 3*, pp. 229–243, Springer, 2004. 90

[107] T. Jörg and S. Deßloch, "Towards generating etl processes for incremental loading," in *Proceedings of the 2008 international symposium on Database engineering & applications*, pp. 101–110, 2008. 90

[108] M. Stonebraker and U. Çetintemel, "" one size fits all" an idea whose time has come and gone," in *Making databases work : the pragmatic wisdom of Michael Stonebraker*, pp. 441–462, 2018. 91

[109] J. Chen, S. He, and X. Li, "A study of big data application in agriculture," in *Journal of Physics : Conference Series*, vol. 1757, p. 012107, IOP Publishing, 2021. 98

[110] I. M. Sukarsa, N. W. Wisswani, and I. G. Darma, "Change data capture on oltp staging area for nearly real time data warehouse base on database trigger," *International Journal of Computer Applications*, vol. 52, no. 11, 2012. 99

[111] Y. Zhan, K. H. Tan, Y. Li, and Y. K. Tse, "Unlocking the power of big data in new product development," *Annals of Operations Research*, vol. 270, pp. 577–595, 2018. 99

[112] I. Mohanraj, K. Ashokumar, and J. Naren, "Field monitoring and automation using iot in agriculture domain," *Procedia Computer Science*, vol. 93, pp. 931–939, 2016. 100