F.S.D... Nº D'ordre:.....

Université Saad DAHLAB de Blida 1

Faculté des sciences

Département d'informatique

Mémoire présenté par :

FERKOUN Asma

ISSAD Chaïma

En vue d'obtenir le diplôme de Master

Domaine : Mathématique et informatique

Filière : Informatique

Spécialité : Informatique

Option : Ingénierie des logiciels

Sujet :

Distribution/ Parallélisation des fonctionnalités d'ETL pour l'intégration des données massives

Soutenu le :10 /10/2016, devant le jury composé de:

M. H. Derrar,	maître assistant, USDB	- Président
M.M. A .Chemchem,	maître assistant (B) USDB	- Examinateur
M. M. Bala,	maître assistant, USDB	-Promoteur

Promotion : 2015/2016

REMERCIEMENT

Suite à la clôture de notre cursus universitaire, et à la présentation de notre mémoire je tenais à remercier :

En premier lieu ALLAH LE TOUT PUISSANT, de nous avoir donné la volonté et le courage afin d'arriver à la finalité de ce modeste travail.

Nos parents qui nous ont beaucoup soutenus pendant toute notre formation, et qui continueront à nous aider dans tous les projets de l'avenir.

Par ailleurs nous souhaiterons manifester notre reconnaissance particulièrement à notre promoteur Mr BALA.M pour tout le savoir qu'il nous a apporté ainsi que pour nous avoir encadré et dirigé au cours de notre projet de fin d'études.

Nous tenons aussi à remercier toute personne qui a participé de près ou de loin pour la réalisation de ce travail.

Plus généralement tout le personnel enseignant du département d'informatique de l'université BLIDA - 1 - qui ont participé à notre formation ainsi qu'à tous les étudiants.

Notre gratitude va aussi à tous les enseignants de nos années précédentes.

MERCI A TOUS

Dédicaces

Je dédie ce modeste travail, avec une énorme joie et un plaisir infini, aux deux merveilleuse personne qui m'ont aidé et guidé vers la voie de la réussite : A mes très chers parents.

Ma mère :

Tu représentes pour moi le symbole de la beauté par excellence, la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi.

Mon père :

Aucune dédicace ne saurait exprimer l'amour, l'estime,

le dévouement et le respect que j'ai toujours eu pour toi, Rien au monde

ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être.

Ce travail est le fruit de tes sacrifices que tu as consentis pour mon

Éducation et ma formation.

Mes sœurs

Ma cousine Hadjer

Et a tous mes amis et amies, qui m'ont soutenu

et aidé dans les moments difficiles:

Hammad, Nimra, Jumma, Nawel, Tarun Kakar, Rahim, Khadidja et Nassima.

ASMA

Dédicaces

Je dédie ce modeste travail, avec une énorme joie et un plaisir infini, aux deux merveilleuse personne qui m'ont aidé et guidé vers la voie de la réussite : A mes très chers parents.

Ma mère :

Tu représentes pour moi le symbole de la beauté par excellence, la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi.

Mon père :

Aucune dédicace ne saurait exprimer l'amour, l'estime,

le dévouement et le respect que j'ai toujours eu pour toi, Rien au monde

ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être.

Ce travail est le fruit de tes sacrifices que tu as consentis pour mon

Éducation et ma formation.

Mon frère

Mes sœurs

Et a tous mes amis et amies, qui m'ont soutenu

et aidé dans les moments difficiles.

Chaima

Résumé

Ce travail s'inscrit dans le cadre des Systèmes d'Informations Décisionnels (SID) et plus particulièrement dans la phase d'intégration basée sur un processus d'extraction, de transformation et de chargement (*ETL: Extracting-Transforming-Loading*) des données dans un entrepôt de données (ED). Notre objectif est de proposer une technique pour que l'ETL peut faire face au gros volumes des données connues sous le nom (Big Data). Dans ce contexte, nous proposons une distribution et une parallélisation de processus ETL à un niveau de granularité fin grâce à un ensemble de ses fonctionnalités de base. Chacune de ces fonctions est implémentée en se basant sur une architecture et un algorithme proposés selon le paradigme de traitement intensif des données massives MapReduce (MR). Pour la mise en œuvre, nous avons retenu l'environnement Apache Spark sur une infrastructure de type cluster, ce qui permet d'améliorer les performances d'ETL.

Mots-clés:

Systèmes d'Informations Décisionnels, Entrepôt de Données, Extraction-Transformation-chargement, Big Data, Processus parallèle/distribué, Niveau de granularité, MapReduce, Apache Spark.

Abstract

This work is part of Decisional Information Systems (DIS) field, inparticular, the integration phase that is based on an process of extraction , transformation and loading data (*ETL : Extracting , Transforming , Loading*) . Our goal is to provide a technique for the ETL process that allow to cope with large volumes of data known as (Big Data) .

In this context, We propose a distribution / parallelization approach for the ETL process in afine-grained level, in terms of its corefunctionalities. Each one of these functions is implemented basing on adistributed architecture and an algorithm defined with the paradigm of processing of Big Data(*MR : MapReduce*). For the implementation, we have chosen the Apache Spark environment on a cluster which improves the performance of ETL .

Keywords:

Decisional Information Systems,Data Warehouse, Extracting-Transforming-Loading, Big Data,Parallel/distributed processing, level of granularity, MapReduce, Apache Spark.

ملخص

هذا العمل هو جزء من الأنظمة المعلوماتية لصنع القرار(SID) و خاصة في مرحلة التكامل. ويستند هذا على عملية الاستخراج، تحويل البيانات والتحميل (ETL: تحويل ، تحميل، استخراج). هدفنا هو توفير تقنية تمكن ETL التعامل مع كميات كبيرة من البيانات المعروفة باسم (البيانات الكبيرة) .

في هذا السياق ، فإننا نقترح تطبيق التوزيع/الموازاة علىETL على مستوى دقيق من التفصيل بفضل مجموعة من الوظائف الأساسية. ويتم تنفيذ كل من هذه الوظائف على أساس تصميم و خوارزمية مقترحان بحسب نموذج العلاج للبيانات الضخمة خارطة- دمج( )MR. من أجل التطبيق اخترنا أباشي سبارك على مجموعة من أجهزة الكمبيوتر وهذا ما يسمح بزيادة تحسين أداء ETL .

كلمات مفتاحية :

الأنظمة المعلوماتية لصنع القرار، مستودع البيانات ،استخراج-تحويل-تحميل، البيانات الكبيرة، المعالجة المتوازية / التوزيعية مستوى التفصيل ،خارطة- دمج، اباشي سبارك.

Table des matières

Liste de Tableaux

Liste des algorithmes

Liste des figures

INTRODUCTION GENERALE

Contexte général

Face à la mondialisation, les entreprises se trouvent confrontées à des environnements de plus en plus complexes et compétitifs dans lesquels le pilotage et la prise de décision deviennent cruciales. L'apparition des systèmes d'information décisionnels (SID) a pour vocation de faciliter l'accès, l'intégration et l'analyse de l'information d'une organisation pour ses décideurs. La dernière évolution notable des SID repose sur les concepts d'entrepôt de données (ED) ou data warehouse (DW) et les outils de restitution. L'entrepôt de données est le cœur du SID : il intègre et stocke d'importants volumes de données issues des différents domaines fonctionnels d'une organisation pour les rendre facilement accessibles aux processus d'interrogation et d'analyses décisionnelles.

Grâce aux nouveaux systèmes d'informations, les réseaux sociaux, les systèmes de paiement, et l'ouverture de multiples bases de données publiques et privées, les entreprises peuvent aujourd'hui recueillir, traiter et puis stocker de gros volumes de données dans l'ED. Devant les difficultés que présente la gestion de ces données volumineuses, les systèmes décisionnels doivent déterminer un processus d'intégration (*ETL: Extracting-Transforming-Loading*) chargé de les collecter à partir de diverses sources, les transformer et enfin les sauvegarder de la façon dont elles peuvent être efficacement consultées ultérieurement.

Comme l'ETL est responsable de l'alimentation de l'entrepôt, nous nous focalisons dans notre travail principalement sur ce processus en se plaçant dans le contexte des projets décisionnels caractérisés par des données massives. Auparavant, les chercheurs ont conduit de multiples travaux de recherches qui ont abouti à la proposition des méthodologies qui le traitent à un niveau de granularité élevéde manière distribuée. En effet, ce dernier est constitué de plusieurs fonctionnalités de base synchronisées. Celles-ci peuvent être exécutées de manière séparées et distribuées afin de satisfaire les besoins en termes de préparation de données à des fins d'analyse et d'aide à la décision.

Le modèle MapReduce est une approche bien établie dans le domaine de calcul parallèle et distribué, il s'articule en deux fonctions de base, Map et Reduce. Et il reste la solution la plus adéquate pour les traitements des données intensives. D'où l'intérêt de mettre en œuvre l'ensemble des fonctionnalités du processus ETL dans un environnement MapReduce.

Problématique

Le processus ETL joue un rôle primordial et décisif dans la réussite d'un projet décisionnel. Néanmoins, l'apparition des données massives qui se caractérisent avec des volumes inhabituels qui s'évoluent sans cesse, détériorent de manière significative ses performances. La complexité des tâches d'ETL ainsi la difficulté de la gestion de ces gros volumes constituent une problématique dans le monde décisionnel, d'où la nécessité de concevoir des outils pour l'optimisation des performances d'ETL dans un contexte du Big Data.

Objectif

Notre travail vise à traiter la problématique du volume des données , l'idée est de définir le processus ETL à un niveau très fin en se basant sur ses fonctionnalités de base telles que Changing Data Capture (CDC), SlowlyChanging Dimension (SCD), Surrogate Key Pipeline (SKP). Celles-ci seront distribuées/parallélisées selon le paradigme MapReduce sur un cluster d'ordinateurs, l'implémentation sera réalisée selon la technique « in-memory » sous Apache Spark..

L'objectif est donc de proposer une architecture distribuéepour l'intégration de données massives (Big data) dans un environnement décisionnel.Ceci nous permettra de restructurer le processus ETL à travers ses fonctionnalités, ce qui permettra une meilleure robustesse, fiabilité et performance.

Organisation du mémoire

Pour assurer une meilleure présentation du travail effectué et garantir la clarté du mémoire, outre cette introduction générale, ce mémoire se compose de cinq chapitres, une conclusion générale. Chacun met en évidence une contribution particulière du travail :

Dans le chapitre I, nous présentons les systèmes d'information décisionnels ainsi que la notion d'entreposage de données.

Dans le chapitre II, nous nous focalisons sur la définition du processus ETL et les approches utilisées pour leur intégration : l'approche de schémas, l'approche virtuelle et l'approche matérialisée.

Dans le chapitre III, nous définissons la notion des données massives (BigData) puis nous présentons notre approche, en proposant des architectures ainsi des algorithmes des fonctions de base d'ETL dans un environnement parallèle/distribué.

Dans le chapitre IV, nous présentonsla spécification des besoins du système et sa conception.

Dans le chapitre V, nous présentonsles méthodes d'implémentation, les outils de développement et lesrésultats des tests.

Enfin, la conclusion présente un bilan de notre travail et des perspectives qui permettraientde compléter notre approche.

CHAPITRE I

CONCEPTS DE BASE SUR LES SYSTEMES DESIONNELS

1. Introduction

Un système d'information(SI) est un ensemble organisé de ressources matériel, logiciel, personnel, données, procédures…permettant d'acquérir, de traiter et de stocker des informations(sous forme de données, textes, images, sons, etc.) dans et entre les organisations [1] .Il est caractérisé donc par un ensemble des moyens et des procédures afin de restituer au moment opportun, des informations sous une forme directement utilisable, par ceux qui en ont besoin pour contrôler, coordonner ou prendre des décisions.

Les SI sont classés en deux catégories : SI opérationnel (SIO) qui a pour objectif premier de servir de support à la réalisation des activités d'un ensemble de processus métier (produire et stocker des données), et un SI décisionnel (SID) qu'il a pour vocation de faciliter la prise de décision [2].

Dans ce chapitre, nous allons présenter la différence entre les systèmes d'information opérationnels et les systèmes d'information décisionnels. Puis, on va citer les concepts de base des entrepôts de données. Enfin, nous discutons sur l'analyse multidimensionnelle.

2. Système d'information opérationnel et système d'information décisionnel

Un système d'information opérationnel (SIO) supporte la réalisation des activités courantes d'une organisation. Il est conçu pour l'exécution de transactions unitaires (autrement dit d'insertions, de mises à jour et de suppressions) et prévisibles c.à.d. elles sont programmées à l'avance au sein du logiciel autour duquel est conçu le SIO, et pour que son exécution soit correcte, le modèle de données sous-jacent au SIO doit être de type entité-association est hautement normalisé ils sont, par contre, inaptes à restituer de l'information aux décideurs des entreprises, ce quia amené les organisations à construire des systèmes à part, dédiés à la restitution d'informations, appelés**systèmes d'information décisionnels** (SID).

Un système décisionnel est donc avant tout un moyen qui a pour but de faciliter la définition et la mise en œuvre de stratégies gagnantes. Il va en particulier aider au pilotage des plans d'actions (prévision, planification, suivi), à l'apprentissage (acquisition de savoir-faire, de connaissances, de compétences) et à la réalisation d'innovations

incrémentales (adaptation du modèle d'affaires : produits/services, organisation, etc. …) [3].

Les demandes d'informations de la part des décideurs sont imprévisibles et elles doivent être accéder non pas unitairement mais en masse .De plus, contrairement aux données des systèmes transactionnels qui sont constamment modifiées et parfois supprimées, l'historisation des valeurs modifiées fournissent des informations.

L'entreprise ne doit pas avoir seulement une vue verticale de ses métiers (SIO) mais une vue transversale (SID). Donc les systèmes de gestion sont dédiés aux métiers tandis que les systèmes décisionnels sont dédiés au pilotage de l'entreprise [4].

Le tableau 1 illustre les principales différences entre les SIO et les SID.

Critère	SIO	SID
Objectif	Exécution de processus métier	Évaluation de la performance des processus métier
Mode d'interaction entre les utilisateurs et la base de données	Insertion, mise à jour, suppression et sélection de données	Sélection de données
Périmètre d'interaction entre les utilisateurs et la base de données	Transactions unitaires	Sélection de données en masse
Type d'utilisation	Prédéfinie, prévisible	Non prédéfinie, imprévisible
Complexité des requêtes des utilisateurs	Faible	Elevée
Optimisé pour	La performance des transactions unitaires	La performance des requêtes de sélection des données en masse
Fréquence de mise à jour de la base de données	Mises à jour en temps réel, au fur et à mesure de l'exécution des processus métier	Mises à jour périodiques en mode « batch »[3]
Historique des données utilisées	Données courantes	Données courantes mais aussi et surtout historiques
Degré de normalisation des données	Hautement normalisé (3e forme normale)	Dénormalisé

TABLEAU 1:DIFFERENCE ENTRE LES SIO ET LES SID [5].

2.1 Séparation physique :

Il est important de séparer les bases de production (SGBD : Système de Gestion de Bases de Données) des bases décisionnelles pour des raisons de performances. Premièrement car

les systèmes de production ne sont pas prévus pour répondre efficacement aux requêtes des systèmes d'aide à la décision puisque ils ne conservent aucune historisation, alors que le SID repose sur des données historisées provenant des systèmes de production différents dont les données ne sont pas nécessairement uniformisées. Ils sont ainsi basés sur deux systèmes différents :

Système OLTP versus Système OLAP

Un environnement dédié pour le traitement transactionnel en ligne (*OLTP : On Line Transaction Processing)* est le modèle utilisé pour la mise à jour des SGBDs. L'objectif est de pouvoir insérer, modifier et interroger rapidement et en sécurité la base. Ces actions doivent pouvoir être effectuées très rapidement par de nombreux utilisateurs simultanément.

Chaque transaction travaille sur de faibles quantités d'informations, et toujours sur les versions les plus récentes des données.

D'autre part les entrepôts de données se reposent sur le système (*OLAP : On Line Analytical Processing)*. Ce système travail en lecture seulement. Les programmes consultent d'importantes quantités de données pour procéder à des analyses. Les objectifs principaux sont regrouper, organiser des données provenant de sources diverses, les intégrer et les stocker pour donner à l'utilisateur une vue orientée métier, retrouver et analyser l'information facilement et rapidement. Cela nécessite de consulter des versions historiques de la base et peut se permettre d'ignorer temporairement les dernières mises à jour. Ces bases sont souvent d'un ordre de grandeur nettement supérieur à celle des bases OLTP, du fait de la conservation de l'historique [6].

Le tableau 2 récapitule les différences entre OLTP et OLAP.

Caractéristiques	OLTP	OLAP
Utilisation	SGBD (base de production)	Entrepôt de données
Opération typique	Mise à jour	Analyse
Type d'accès	Lecture écriture	Lecture
Niveau d'analyse	Elémentaire	Global
Quantité d'informations échangées	Faible	Importante
Orientation	Ligne	Multidimensionnel
Taille BD	Faible (max qlqs GB)	Importante (pouvant aller à plusieurs ZB).
Ancienneté des données	Récente	Historique

TABLEAU 2: DIFFERENCES ENTRE OLTP ET OLAP [6].

3. Les entrepôts de données (ED)

3.1Définition :

"Un Entrepôt de Données est une collection de données orientées sujet, intégrées, non volatiles et historiées, organisées pour le support d'un processus d'aide à la décision"[7].

Nous détaillons ces caractéristiques :

Orientées sujet: Les données des entrepôts sont organisées par sujet plutôt que par application. Par exemple, une chaine de magasins d'alimentation organise les données de son entrepôt par rapport aux ventes qui ont été réalisées par produit et par magasin, au cours d'un certain temps.

Intégrées : Les données provenant des différentes sources doivent être intégrées ,avant leur stockage dans l'entrepôt de données. L'intégration (mise en correspondance des formats, par exemple), permet d'avoir une cohérence de l'information.

Non volatiles: A la différence des données opérationnelles, celles de l'entrepôt sont permanentes et ne peuvent pas être modifiées. Le rafraichissement de l'entrepôt, consiste à ajouter de nouvelles données, sans modifier ou perdre celles qui existent.

Historiées: La prise en compte de l'évolution des données est essentielle pour la prise de décision qui, par exemple, utilise des techniques de prédiction en s'appuyant sur les évolutions passées pour prévoir les évolutions futures.

3.2 Les classes de données dans un ED :

Un entrepôt de données peut être structuré en quatre classes de données organisées selon un axe historique et un axe de synthèse.

Les données agrégées

Les données agrégées correspondent à des éléments d'analyse représentant les besoins des utilisateurs. Elles constituent déjà un résultat d'analyse et une synthèse de l'information contenue dans le système décisionnel, et doivent être facilement accessibles et compréhensibles.[8]

Les données détaillées

Les données détaillées reflètent les événements les plus récents. Les intégrations régulières des données issues des systèmes de production vont habituellement être réalisées à ce niveau.

Les données historiées

Chaque nouvelle insertion de données provenant du système de production ne détruit pas les anciennes valeurs, mais créée une nouvelle occurrence de la donnée.

Les métadonnées

Les métadonnées sont des données sur les données indispensables à une exploitation efficace d'un entrepôt de données. Elles constituent l'ensemble des données qui décrivent le schéma d'un entrepôt de données, l'ensemble des règles, des définitions, des transformations et des processus associés à une donnée.[9]

Selon Ralph Kimball, on distingue deux types de métadonnées:

- **Les métadonnées de la zone de construction (back room):** des outils d'arrière-plan sont procédurales, elles guident les processus d'extraction, de nettoyage et de chargement.

- **Les métadonnées des outils frontaux (front room)** : sont plus descriptives et aident les outils de requête et les générateurs d'états à fonctionner du mieux possible.[10]

3.3 Les magasins de données (Data Mart) :

Les entrepôts de données sont généralement très volumineux et très complexes à concevoir, nous avons décidé de les diviser en bouchées plus faciles à créer et à entretenir. Ce sont les *Data Marts*. On peut faire des divisions par fonction (un data mart pour les ventes, pour les commandes, pour les ressources humaines)[11].

Un magasin de données est une vue partielle de l'entrepôt de données orientée métier. C'est un sous-ensemble de l'entrepôt de données contenant des informations se rapportant à un secteur d'activité particulier de l'entreprise ou à un métier qui y est exercé. Il se situe en aval de l'entrepôt de données et est alimenté par celui-ci. On peut donc créer plusieurs magasins de données correspondants aux différents besoins des utilisateurs, ce qui permet de réduire le nombre d'opérations sur l'entrepôt de données. De plus, cela permet d'offrir aux utilisateurs un outil spécifiquement adapté à leurs besoins. Cet outil sera plus petit et permettra donc un accès plus rapide à l'information.

La figure 1 illustre la notion de magasin de données en représentant l'extraction de deux magasins DM Marketing et DM RH, dédiés respectivement aux services Marketing et Ressources Humaines, à partir d'un entrepôt de données ED.

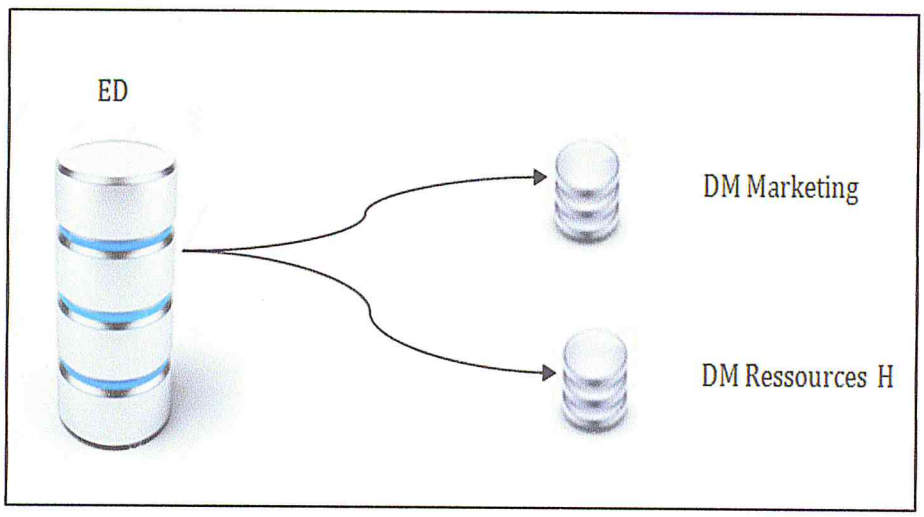

Figure 1 : Exemples de magasins de donnée

3.4Modélisation dimensionnel d'un entrepôt de données(ED) :

La modélisation dimensionnelle est une technique de conception conceptuelle permettant de structurer les données de manière à les rendre **intuitives** aux utilisateurs d'affaires et offrir une **bonne performance** aux requêtes [12].

Cette technique consiste à organiser les données autour des faits que l'on cherche à analyser, caractérisés à l'aide d'indicateurs (appelés mesures) qui sont des données normalement numériques et additives, permettant de mesurer l'activité modélisée. Ces faits sont décrits par un ensemble d'axes d'analyse, ou dimensions, d'où le terme de modèle multidimensionnel. Elle se présente comme une alternative au modèle relationnel et elle correspond mieux aux besoins du décideur tout en intégrant la modélisation par sujet.

3.4.1 Niveau conceptuel :

A ce niveau on s'intéresse à la description de la base multidimensionnelle indépendamment des choix d'implantation.

Pour une base de données opérationnelle, on parle en termes de tables et de relations, une table étant une représentation d'une entité et une relation un lien entre ces entités. Et bien pour un entrepôt de données, on parle en termes de Dimension et de Faits.

- **Fait**

Le fait représente le sujet d'analyse. Il est composé d'un ensemble de mesures qui représentent les différentes valeurs de l'activité analysée.
La table de faits est la clef de voûte du modèle dimensionnel où sont stockés les indicateurs de performances (mesures). Le concepteur s'efforce de considérer comme indicateurs les informations d'un processus d'entreprise dans un système d'information [8].

- **Dimension**

Une dimension est une table qui représente un axe d'analyse selon lequel on veut étudier des données observables(les faits) qui, soumises à une analyse multidimensionnelle, donnent aux utilisateurs des renseignements nécessaires à la prise de décision [2].

- **Cube**

Dans le modèle multidimensionnel, le concept central est le cube, lequel est constitué des éléments appelés cellules qui peuvent contenir une ou plusieurs mesures. La localisation de la cellule est faite a travers les axes, qui correspondent chacun a une dimension [7] où Chaque cellule est définie par une seule valeur de chaque dimension. Il aboutit à présenter les données non plus sous forme de tables mais de **cube** centré sur une activité. Un cube de dimension n (n > 3) est aussi dit **hyper cube**.

La figure 2 montre un exemple de cube permettant l'analyse de «l'effectif des étudiants » selon trois dimensions : cycle, spécialité et année.

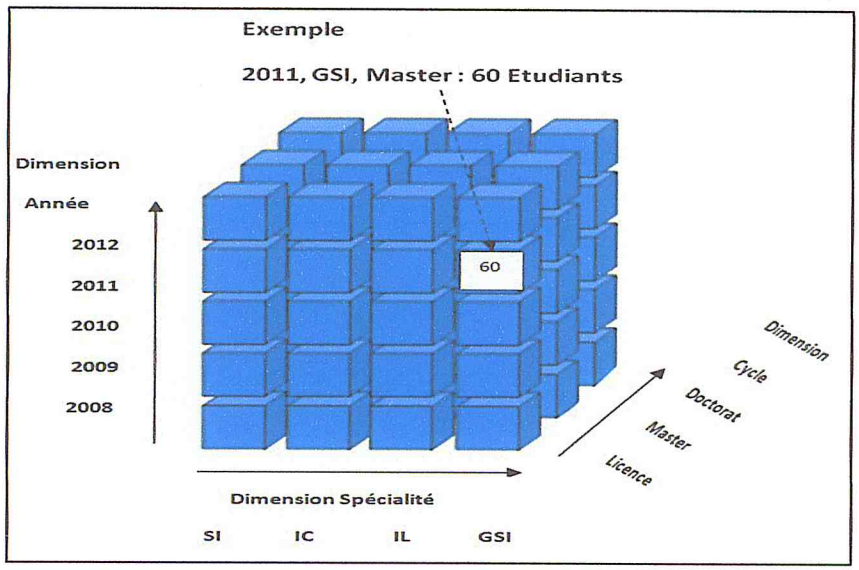

Figure 2 : Cube multidimensionnel à trois perspectives d'analyse.

3.4.1.1 Schémas multidimensionnels :

Au sein de l'entrepôt de données les données sont redondantes et dé normalisées, cela permet de faciliter l'utilisation et d'améliorer les performances lors de l'analyse des données. Pour les représenter Ils existent trois types de schémas qui sont fréquemment rencontrés, le schéma en étoile, le schéma en flocon et le schéma en constellation de faits.

❖ Schéma en étoile :

Il se compose du fait central et de leurs dimensions. Dans ce schéma il existe une relation pour les faits et plusieurs pour les différentes dimensions autour de la relation centrale. La relation de faits contient les différentes mesures et une clé étrangère pour faire référence à chacune de leurs dimensions.

➢ **Avantages:**

- Facilité de navigation
- Performances : nombres de jointures limités

➢ **Inconvénients :**

- Toutes les dimensions ne concernent pas les mesures
- Redondances dans les dimensions

La figure 3 montre le schéma en étoile en décrivant l'effectif des étudiants par spécialité, cycle au cours d'une année. Dans ce cas, nous avons une étoile centrale avec une table de faits appelée TF_ Etud et autour leurs diverses dimensions TD_ Spécialité, TD_ Cycle et TD_ Année.

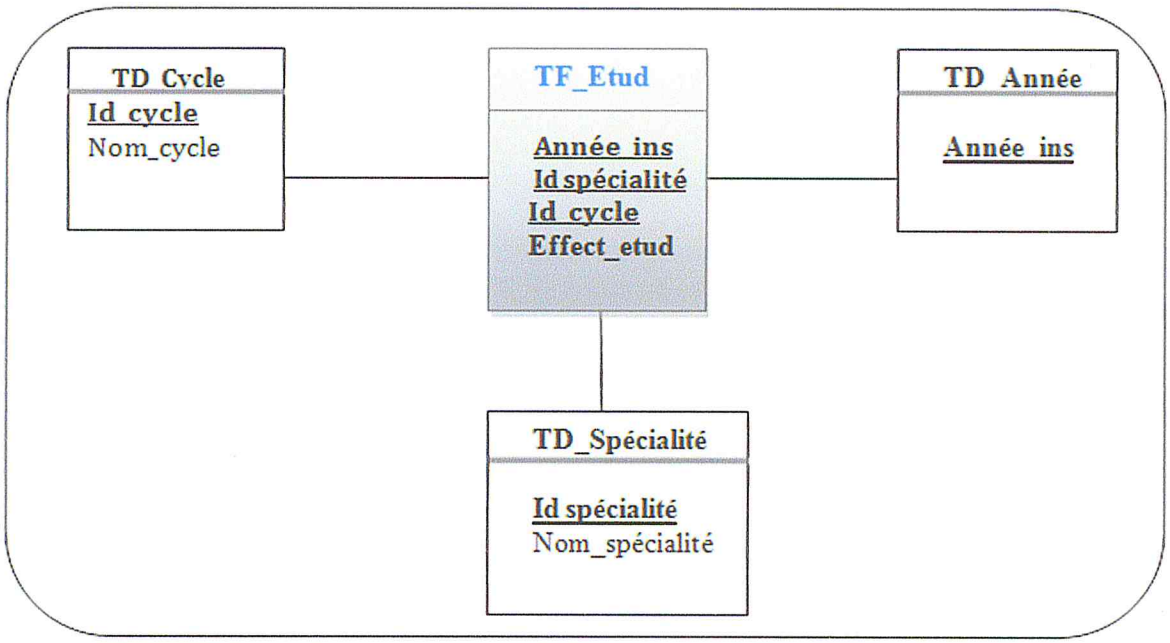

Figure 3 : Exemple de modélisation en étoile.

❖ Schéma en flocon de neige (Snow flake):

Il dérive du schéma précédent avec une relation centrale et autour d'elle les différentes dimensions, qui sont éclatées ou décomposées en sous hiérarchies.

➢ **Avantages:**

- Réduction du volume,
- Permettre des analyses par pallier (drill down/Roll up) sur la dimension hiérarchisée.

➢ **Inconvénients :**

- Navigation difficile ;
- Nombreuses jointure.

La figure 4 montre le schéma en flocon de neige dont la dimension « TD_Spécialité» est éclatée en sous hiérarchies TD_Département et TD_Faculté.

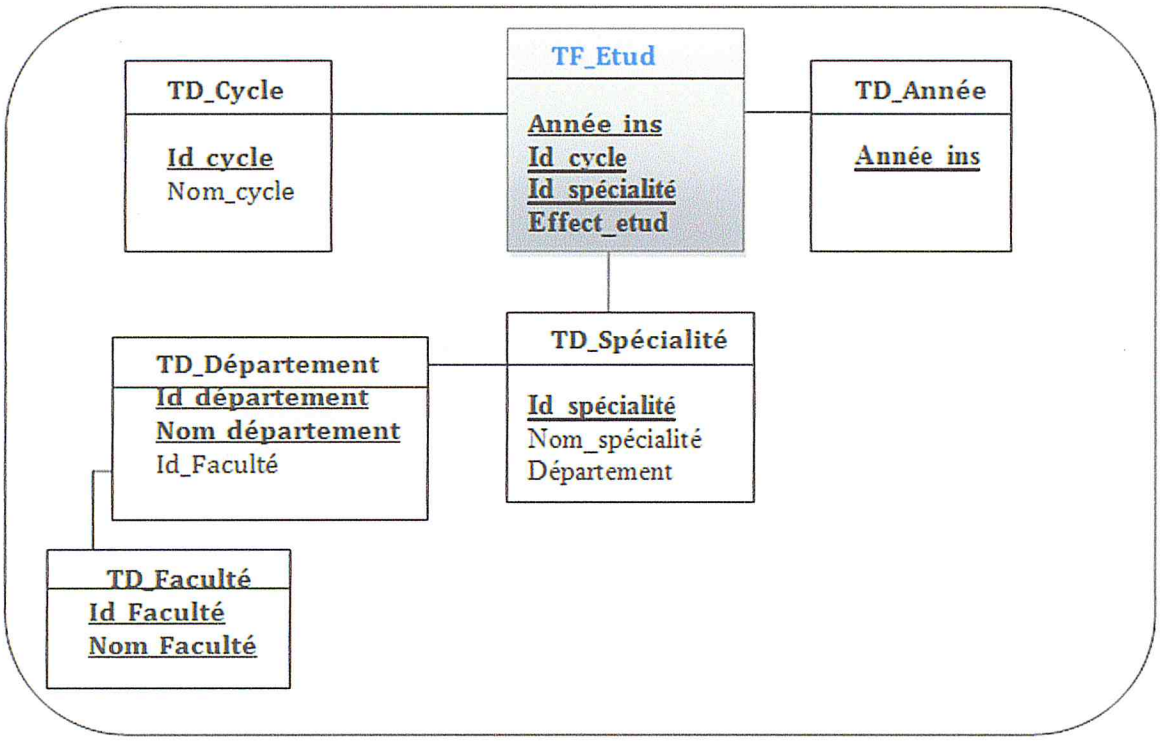

Figure 4 : Exemple de modélisation en flocon de neige.

❖ Schéma en constellation :

Le schéma en constellation représente plusieurs relations de faits qui partagent des dimensions communes.

La figure 5 montre le schéma en constellation qui est composé de deux relations de faits. La première s'appelle TF_Etud qui enregistre l'effectif des étudiants inscrit par spécialité, cycle et année d'inscription. La deuxième relation gère le taux de réussite des étudiants par spécialité, année et sexe.

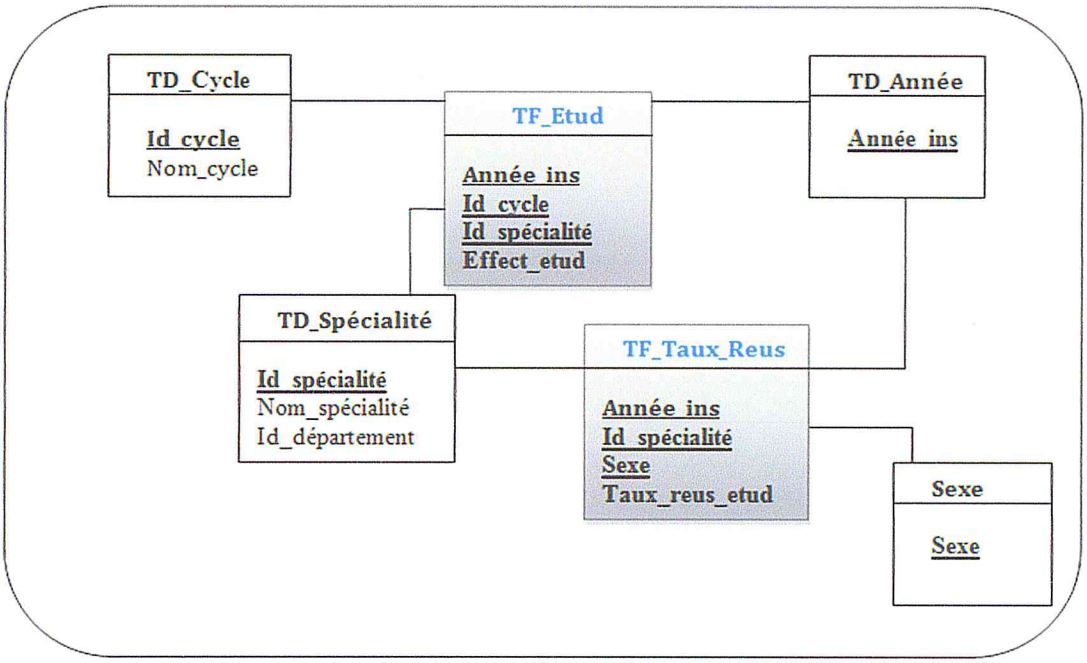

Figure 5 : Exemple de modélisation en constellation.

3.4.2 Niveau Logique :

Le niveau logique consiste à donner une description de la base multidimensionnelle suivant la technologie utilisée (Relational-OLAP, Multidimensionnel-OLAP ou Hybrid-OLAP) afin de définir la façon dont les données seront stockées physiquement pour permettre des analyses multidimensionnelles. Nous allons expliquer ces possibilités, mais il est important de comprendre que chacune c'est du OLAP (Online AnalyticalProcessing).

❖ OLAP

Les systèmes décisionnels complets reposent sur la technologie OLAP, conçue pour répondre aux besoins d'analyse des applications de gestion. L'acronyme FASMI (*FastAnalysis of SharedMultidimensional Information*) permet de résumer la définition des produits OLAP. Cette définition fut utilisée pour la première fois en 1995 et depuis aucune autre définition n'est plus proche pour résumer le terme OLAP. [13]

Fast: Le temps de réponse aux demandes des utilisateurs oscille entre 1 et 20 secondes. Les constructeurs utilisent des pré-calculs pour réduire les durées des requêtes.

Analysis: Le système doit pouvoir faire face à toutes les logiques d'affaires et de statistiques, ainsi que fournir la possibilité aux utilisateurs de construire leurs calculs et leurs analyses sans avoir à programmer. Pour cela, il y a des outils qui seront fournis par le constructeur.

Shared: Le système doit créer un contexte où la confidentialité est préservée et doit gérer les cas où plusieurs utilisateurs ont des droits en écritures. Ce point constitue la plus grosse faiblesse des produits actuels.

Multi dimensionnel: C'est la caractéristique clé. Le système doit fournir des vues conceptuelles multidimensionnelles des données. Il doit supporter aussi les hiérarchies.

Informations : L'ensemble des données et les informations nécessaires pour un produit OLAP. (Nous exposons dans la suite les divers types de stockage des informations dans les systèmes décisionnels).

❖ ROLAP

Relationnel OLAP, Comme son nom l'indique, il utilise le concept relationnel pour stocker des données modélisées dans le format multi dimensionnel. Les analyses sont transformées en requêtes SQL classiques qui sont exécutées sur les tables. R-OLAP utilise aussi la notion de tables d'agrégats, c'est-à-dire créer des tables contenant des données sommaires et les stocker en mémoire en cas d'utilisation [14].

La technologie ROLAP a deux avantages principaux : elle permet la définition de données complexes et multidimensionnelles en utilisant un modèle relativement simple, cela permet une facilité dans la mise à jour des données, et elle réduit le nombre de jointures à réaliser dans l'exécution d'une requête. Le désavantage est que le langage de requêtes tel qu'il

existe, n'est pas assez puissant ou n'est pas assez flexible pour supporter de vraies capacités d'OLAP. [13]

❖ MOLAP

Le Multidimensionnel OLAP consiste à utiliser un système multidimensionnel pur, qui gère des structures multidimensionnelles natives. Elles utilisent des tableaux à n dimensions. L'accès aux données se fait directement dans le cube. Cela permet une rapidité d'accès à l'information mais augmente le temps de mise à jour. [6]

D'un côté, les requêtes MOLAP sont très puissantes et flexibles en termes du processus OLAP, tandis que, d'un autre côté, le modèle physique correspond plus étroitement au modèle multidimensionnel. Néanmoins, il existe des désavantages au modèle physique MOLAP. Le plus important, c'est qu'il n'existe pas de standard du modèle physique. [13] par addition il montre très rapidement ses limites quand on commence à jouer avec de gros volumes de données.

❖ HOLAP (Hybrid OLAP)

HOLAP est un hybride entre ROLAP et HOLAP. Les parties tables de faits et tables de dimensions sont stockées dans une base relationnelle standard tandis que le reste des données (les calculs) sont stockées dans une base multidimensionnelle. Elle permet de minimiser les défaillances des technologies R-OLAP et M-OLAP[6].

❖ OOLAP

C'est la technologie la plus récente, Object OLAP, qui s'appuie sur le paradigme objet. Le modèle multidimensionnel se traduit ainsi :

- chaque fait correspond à une classe, appelée classe de fait.
- chaque dimension correspond à une classe, appelée classe de dimension.

Le principal avantage est d'augmenter le niveau d'abstraction[6].

3.4.3 Niveau physique :

La modélisation physique consiste à choisir un mode d'implantation particulier dépendant du logiciel utilisé (notamment le SGBD)et à définir des scripts SQL pour la création et l'alimentation de l'entrepôt [15].

4. Conclusion

Dans ce chapitre, nous avons présenté tous d'abord montré les différents points faisant la différence entre un système d'information opérationnelle (SIO) et un système d'information décisionnel(SID). Par la suite, nous avons exposé les concepts de base d'un SID, ainsi les trois niveaux de la modélisation d'un entrepôt de données, et les outils d'accès à ce dernier afin de restituer les informations pour des fins d'analyses et d'aide à la décision.

Pour que les données sources soient adaptées et adéquates aux besoins des décideurs, celles-ci sont filtrées, nettoyées et intégrées avant d'être chargées, par la suite, dans l'entrepôt de données. Toutes ces tâches sont prises en charge par un processus d'Extraction, de Transformation et de Chargement (*ETL: Extracting- Transforming-Loading*). Pour cela nous allons consacrer le prochain chapitre pour la présentation de ce processus.

CHAPITRE II

PROCESSUS D'INTEGRATION DE DONNEES ETL

1. Introduction

Les processus d'extraction, de transformation et de chargement (*ETL: Extracting-Transforming-Loading*) ont pour objectif de relier les différentes sources de données pour les extraire vers une unique destination qui est l'entrepôt de données.

Le processus ETL est décomposé en trois phases principales, qui sont : l'extraction (E) qui se connecte sur des systèmes opérationnels hétérogènes (SGBD, ERP, applications spécifiques, fichiers plats..) pour extraire les données sources pertinentes aux besoins d'analyse. La transformation (T) qui consiste à nettoyer, conformer, standardiser, corriger et intégrer les données au niveau d'une zone de préparation (*DSA : Data Staging Area*) Enfin, le chargement (L) consiste à transférer les données préparées vers leur destination appropriées en se basant sur un schéma de mappage. La figure 7 résume l'enchaînement des trois étapes d'un processus ETL.

Dans ce chapitre, nous allons présenter les différentes approches d'intégration de données ainsi que les étapes d'un processus ETL et les structures de données sous-jacentes.

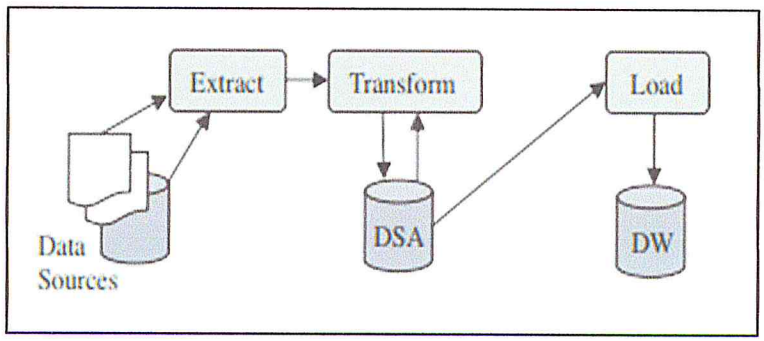

Figure 6: ENVIRONNEMENT D'UN PROCESSUS ETL [17].

2. Les approches d'intégration de données :

La nécessité de résoudre les conflits d'intégration inhérents à l'hétérogénéité des bases de données afin de disposer un ensemble intégré d'informations constitue, depuis les années 1980, un enjeu majeur dans le domaine de la recherche en informatique. Cette intégration peut se faire via différentes approches, dépendant le plus souvent de l'application finale visée. Cependant, dans tous les cas elle nécessitera de résoudre l'hétérogénéité sémantique des bases de données à intégrer [18].

Une première classification des différentes approches repose sur le contexte d'intégration, et par conséquent, le type des entrées/sorties du processus d'intégration, et le but du processus lui-même. Nous distinguons l'intégration de schémas, l'intégration de données virtuelle, et l'intégration de données matérialisée.

2.1 L'intégration de schémas :

Dans ce cas, l'entrée de l'intégration est un ensemble de schémas sources, et la sortie est un schéma de données correspondant à la représentation intensionnelle réconciliée de tous les schémas en entrée. L'entrée comporte également la spécification de la façon d'associer les schémas des données sources à des parties du schéma résultant (cible) [19].

2.2 L'intégration de données virtuelle :

L'intégration de données virtuelle pour laquelle l'entrée est constituée d'un ensemble de sources de données hétérogènes, et la sortie d'une spécification décrivant comment fournir un accès global et unifié aux sources de données afin de bénéficier des informations qui y sont contenues sans toutefois interférer avec l'autonomie de ces sources.

Un exemple typique d'intégration de données virtuelle est l'approche médiateur, proposée par Wieder hold (1992), qui consiste à définir une interface destinée à fournir un accès unifié à différentes sources de données qui ne sont pas nécessairement des bases de données, et peuvent par conséquent être peu structurées et documentées. L'objectif est de donner l'impression d'interroger un système centralisé et unifié, alors que les différentes sources de données sont réparties, autonomes et hétérogènes. Un médiateur comprend une base de connaissances qui modélise le domaine d'application, fournit un vocabulaire commun pour l'expression des requêtes, et fait le lien entre les différentes sources de données accessibles. Un médiateur est donc une couche logicielle permettant à l'utilisateur d'interroger de façon transparente plusieurs sources de données réparties et hétérogènes [18].

2.3 L'intégration de données matérialisée :

L'intégration de données matérialisée pour laquelle l'entrée est également constituée d'un ensemble de sources de données, mais dont la sortie consiste en un ensemble de données représentant une vue réconciliée des sources, à la fois au niveau des schémas et au niveau des données.

C'est le cas de l'approche entrepôt de données où les données sont organisées, coordonnées, intégrées et stockées pour donner à l'utilisateur une vue globale et unifiée des

données. Les entrepôts de données sont conçus dans le but de rassembler les données d'une entreprise au sein d'une base unique afin de faciliter l'analyse et la prise de décisions. Il y a duplication des données et il n'est pas nécessaire d'accéder aux sources pour répondre à une requête. Le schéma global n'est pas figé : il est amené à évoluer, à chaque ajout ou réorganisation de données [18].

Le tableau suivant montre les différences majeures entre l'approche virtuelle et l'approche matérialisée :

Critère	approche virtuelle	approche matérialisée
Performance	défi principale	Bonne
Accès aux données	accès direct aux sources	accès à une copie des données
Migration de requêtes	vers les sources	vers l'entrepôt
Rafraichissement des données	données toujours fraîches	données pas toujours fraîches
Nettoyage et filtrage de données	Non	oui
Versionnement	Non	Oui

TABLEAU 3 : L'approche matérialisée versus l'approche virtuelle [20]

3. Définition du processus ETL

« **Extract-Transform-Load**» est connu sous le terme **ETL** (ou parfois : ***data pumping***). Il s'agit d'une technologie informatique intergicielle permettant d'effectuer des synchronisations massives d'information d'une banque de données vers une autre.

Elle est basée sur des connecteurs servant à exporter ou importer les données dans les applications, des transformateurs qui manipulent les données (agrégations, filtres, conversions...), et des mises en correspondance (mappages) [4].

Il permet la consolidation des données à l'aide des trois opérations suivantes:

3.1 Extraction des données :

Cette opération s'agit en premier lieu d'aller chercher et identifier les données de sources ayant subi une modification depuis la dernière exécution. L'extraction peut se faire à travers un outil d'alimentation qui doit travailler de façon native avec les différentes applications, bases de données ou fichiers sources, ou alors créer des programmes extracteurs. L'outil doit être à même de lire sélectivement les données sources, et donc de filtrer les données en lecture afin de n'extraire que l'information pertinente.
Pour cela, plusieurs technologies sont utilisables [21] :

- **Les passerelles :**

Fournies principalement par les éditeurs de bases de données, ces passerelles sont généralement insuffisantes car elles sont mal adaptées aux processus de transformation complexes.

- **Les utilities de replication :**

Utilisables si les systèmes de production et décisionnels sont homogènes et si la transformation à appliquer aux données est légère.

- **Les outies specifies detraction** :

Ces outils sont certainement la solution opérationnelle au problème de l'extraction, mais leur prix relativement élevé est un frein à leur utilisation dans les premières applications.

3.1.1 Les politiques d'Extraction :

Selon la nature du système source ainsi que le volume des données qui doivent être traitées, les différentes politiques qui peuvent être adoptées sont [22] :

- **Politique 1 :**

Elle consiste à extraire toutes les données sources et de les traiter comme si le chargement initial de l'ED.

- **Politique 2 :**

Elle consiste à l'extraction d'une capture des données (Snapshot), qui est ensuite comparée à une capture de données précédente, par la suite les insertions, les suppressions et les mises à jour seront détectées. Dans ce cas, seulement les données affectées par des modifications sont soumises à l'étape de transformation.

- **Politique 3 :**

Elle implique l'utilisation de déclencheurs (Trigger) dans la source qui sont activés à chaque fois qu'une modification se produit dans la base de données source. Evidemment, cela peut se faire que si la base de données source est un système relationnel.

- **Politique 4 :**

Elle s'agit de prendre une capture des changements à partir des journaux de transactions des bases de données à un certain temps et la parcourir pour détecter les modifications qui affectent les tables participant au processus ETL.

3.2 Transformation de données :

Les données extraites de la source ne sont pas forcément dans l'état dans lequel elles seront stockées. Elles doivent être vérifiées, reformatées et nettoyées afin d'éliminer les valeurs incohérentes ainsi que les doublons. Il s'agit donc d'une suite d'opérations qui a pour but de rendre les données cibles homogènes et puissent être traitées de façon cohérente.

3.2.1 Les tâches de transformation :

L'ensemble de manipulations nécessaire au processus de transformation dépend des données sources. Certaines d'eux exigentpeu de transformations, tandis que d'autres peuvent nécessiter une ou plusieurs techniques de transformation pour répondre aux exigences d'entreprise. Les procédés les plus couramment utilisés pour la transformation sont :

- **Nettoyage de données :**

Le nettoyage des données s'occupe de détecter et de corriger ou de supprimer les erreurs et les incohérences présentes sur les données afin d'améliorer leur qualité. Par la suite,nous classons les principaux problèmes de qualité des données à être résolus par le nettoyage des données.

- **Les problèmes de qualité de données :**
  - **Problèmes au niveau du schéma :**

Les principaux problèmes en ce qui concerne le niveau de schéma sont :

-*Des conflits de dénomination* :Lemême nom est utilisé pour des objets différents (homonymes) ou des noms différents sont utilisés pour le même objet (synonymes).

-*Des conflits structurels* : Deux différentes représentations du même objet dans deux différentes sources.

-*Des conflits de modélisation* : Différents modèles de données sont utilisés.

-*Des conflits de granularité* :Différents niveaux de détail de représentation (par exemple, année →mois →jour).

- **Problèmes au niveau d'enregistrement** :
  - Enregistrementdupliqués (Redondance).
  - Enregistrements contradictoires (par exemple, emp1= (nom="John Smith",datN=12/06/1999...);emp2=(nom="JohnSmith",datN=12/06/1964...)
  - Différents niveaux d'agrégation(par exemple, les ventes par jour dans source 1 / ventes par an dans source 2).
- **Problèmes au niveau de la valeur :**
  - Différentes représentations (par exemple, pour le sexe: 'Male', 'M', '1').
  - Value's manquante soundless.
  - Violation des contraintes d'intégrité (par exemple, date 28/13/2016).
  - Variété des abréviations : (par exemple, le mot 'Identificateur' est désigné par : 'identif','ident' et 'id').
  - Erreurs de frappe.

❖ **Génération de clé de substitution SK :**

Une clé de substitution (Surrogate Key) est une clé non intelligente utilisée afin de substituer la clé naturelle (Business key) qui provient des systèmes opérationnels. Elle est alors utilisée dans l'entrepôt de données pour remplacer et compléter la clé naturelle du système opérationnel afin de rendre un élément unique dans la dimension et elle sert aussi à faire les jointures avec les tables de faits ou les autres tables de dimension.

- **Avantages**
  - *Performance* :Accélère l'accès aux données du moment ou l'on va utiliser un index numérique vu que le type de données de la clé de substitution est numérique.
  - *Indépendance du système source:* On ne peut garantir que la clé d'affaire ne change pas dans les systèmes sources.
  - *Historique des changements et granularité* infinie : Si l'on désire garder l'historique des changements de la dimension selon certains critères, on doit gérer la clé de substitution. On se retrouve facilement avec plusieurs enregistrements de la même clé d'affaire dans la dimension [23].

❖ **Jointure** : faire joindre les données provenant de sources multiples.

- **Filtrage** : sélectionner seulement les données à charger (par exemple, sélectionner que certaines colonnes).
- **Eclatement** : Fractionner une colonne en plusieurs colonnes (ex. éclater la colonne qui contient une adresse composée de plusieurs champs en différentes colonnes).
- **Validation** :appliquer des règles afin de chercher les données pertinentes à partir des tables ou des fichiers référentiels pour les dimensions à variation lente (par exemple, rejeter une ligne si elle contient 3 colonnes vides).
- **Agrégation** : c'est un cumul résumant plusieurs lignes de données (par exemple, effectif des employés par âge, type contrat), ce qui permet d'avoir des temps de réponse très courts. Le niveau d'agrégation est choisi au moment de la construction de l'ED.
- **Tri et arrangement.**
- **Eliminer les doubles.**
- **La conversion** : Elle sert à transformer les données provenant des différentes sources dans un format cible.
- **La normalisation** :Cette tâche consiste à normaliser les données provenant des sources hétérogènes pour les rendre homogènes. Elle nécessite des règles précises servant de référentiel qui sont mémorisées sous forme de métadonnées.

3.3 Chargement de données :

C'est l'opération qui consiste à charger les données nettoyées et préparées dans l'entrepôt et à gérer les changements aux données existantes en définissant des stratégies d'historisation et de rafraîchissement (ex: stratégies SCD qu'on va traiter par la suite).

Elle est une phase plutôt mécanique et la moins complexe mais elle risque d'être assez longue lorsque les données sont volumineuses, donc il est nécessaire de veiller à ce que la charge est exécutée correctement et avec le moins de ressources que possible. Pour cela et afin d'améliorer les performances de chargement dans une base de données –l'ED est souvent une BD-, il est utile de désactiver les indexes de la base de données et les contraintes préalablement au chargement et leur permettre de retour postérieurement.

Étant donné que toutes les tables de faits doivent préserver l'intégrité référentielle, la clé de dimension primaire est reliée à une clé étrangère correspondante dans la table de faits. C'est pour cette raison qu'il est important d'alimenter les tables de dimensions, puis les tables de faits.

Les tables de dimensions sont généralement alimentées à partir d'une ou plusieurs sources, elles contiennent autant que possible des attributs permettant de qualifier ou d'expliquer l'activité. Donc elles représentent des entités complémentaires à la conception de la table de faits qui contiennent les mesures d'activité de l'entreprise.

4. Définition d'ETL en fonction de ses fonctionnalités de base

Dans la section précédente, nous avons traité le processus ETL de manière globale à un niveau de granularité élevé. Dans cette section, nous définissons ce dernier comme étant un ensemble de plusieurs fonctionnalités de base synchronisées de manière à satisfaire les besoins en termes de préparation de données à des fins d'analyse et d'aide à la décision.
Une fonctionnalité ETL est une fonction de base qui prend en charge un aspect particulier dans le processus telles que Changing Data Capture (CDC), Surrogate Key (SK), SlowlyChanging Dimension (SCD), Surrogate Key Pipeline (SKP).

4.1 Changing Data Capture (CDC) :

C'est une fonctionnalité qui permet de suivre et de capturer les différentes modifications ayant eu lieu sur une table. Les modifications capturées par « CDC » sont toutes celles effectuées sur la table via les instructions d'insertion (INSERT), de mise à jour (UPDATE) ou de suppression (DELETE).
Plusieurs techniques de CDCexistent, nous citons les plus connues : techniques basées sur les déclencheurs(triggers), techniques basées sur l'heurodatage et techniques basées sur des captures instantanées(snapshots). Cette dernière techniqueassure de trouver tous les changements car elle garde une capture complète de la version précédente de la table, et la compare, tuple par tuple avec la version récente (actuelle) en utilisant des algorithmes CRC (contrôle de redondance cyclique) qui peuvent déterminer si un tuple a reconnu des changements.

- **Importance du CDC :**

L'objectif du CDC est d'optimiser l'intégration des données (du processus ETL) en requêtant directement les modifications faites sur une table au lieu de travailler sur l'intégralité de la table et ce faisant augmenter les temps de traitement. Elle permet entre autres de faire de l'audit de base, de faire de la synchronisation entre deux bases[24].

4.2 Surrogate Key Pipeline (SKP) :

A la fin de la phase de transformation du processus ETL, l'alimentation des tables de dimension s'exécute avant celui des tables de fait pour préserver l'intégrité référentielle envers les dimensions associées. Pour ce faire, tous les tuples sources à charger dans la table de fait passent par un processus pour remplacer chacune de ses NK par la clef SK correspondante la plus récente dans la table de dimension appropriée. Lorsqu'un tuple Ti est dans la phase de remplacement de sa clef NKj, le tuple Ti+1 est dans une phase de remplacement de sa clef NKj-1 et ainsi de suite, jusqu'à ce que toutes ses NK seront substituées, d'où l'appellation SK Pipeline.

4.3 Slowly Changing Dimension (SCD):

Slowly Changing Dimension (SCD) est l'une des fonctionnalités de base d'un processus ETL. Lorsque les données sources sont préparées dans la phase de transformation, une partie de celles-ci sont destinées à être chargées dans des tables de dimension en appliquant des approches de visionnement qu'elles permettent de spécifier pour chaque attribut de la dimension, la manière avec laquelle celui-ci sera maintenu (aucune mise à jour, mise à jour sans archivage, mise à jour avec archivage).

❖ **Les types de SCD:**

Ils existent plusieurs situations, qu'elles définissent si le tuple qui a subi une modification doit être écrasé ou archivé, en se basant sur le type SCD de l'attribut modifié. Nous les citons dans ce qui suit :

- **SCD type 1:**

Les attributs de la dimension de type SCD 1 sont écrasés lors des mises à jour puisqu'ils ne nécessitent aucune historisation. Bien que cette approche est facile à mettre en œuvre et ne crée pas de tuples de dimension supplémentaires, il faut prendre en considération que les tables de fait agrégées et les cubes OLAP affectés par ce changement sont recalculés.

- **SCD type 2:**

Les attributs SCD type 2 nécessitent une historisation de leurs différentes valeurs (versionnement). Pour ce faire, au moment d'une mise à jour concernant un attribut SCD 2, l'ancien tuple est préservé sans mise à jour. Celui-ci est dupliqué avec la même clé naturelle (NK) mais pour distinguer entre les deux versions, le nouveau tuple lui est affecté une nouvelle valeur pour sa clé de substitution (SK). C'est dans ce nouveau tuple que sera opérée la mise à jour sur les attributs de type SCD 2.

- **SCD type 4 :**

Les attributs de type SCD 4 obéissent au même principe de SCD type 2 puisque eux aussi nécessitent un versionnement. La seule différence est au lieu de préserverles anciennes versions du tuple dans la table de base, celles-ci sont transférées dansune autre table dédiée pour l'historisation. Cette manière de gérer le versionnement permetde maintenir la table de base propre (contient seulement les tuples les plus récents)avec une taille modérée.

5. Structures de données du processus ETL

5.1 Sources de données :

Afin d'alimenter les entrepôts, les informations doivent être identifiées et extraites de leurs emplacements originels. Il s'agit majoritairement de données internes à l'entreprise, stockées dans des bases de données de production de différents services, ou des sources externes récupérées via des services distants, des web services, par exemple. Ce sont des données complexes ayant plusieurs formats ce qui entrent en jeu pour les acquérir.

❖ **Fichiers plats**

- **Fichiers CSV :**

  Un fichier CSV (Comma Separated Values) est un simple fichier (texte) dans lequel les valeurs sont séparées généralement par une virgule, ce qui permet de sauvegarder les données dans un format de tableur dont chaque ligne comporte le même nombre des champs. Ce type de fichier est très facile à manipuler (par exemple, il est utilisé pour envoyer des données (par FTP, par email, etc.).

  Exemple :

	A	B	C	D
1	1,2002-09-07,Master,SI,Null,Null			
2	2,2002-10-16,Licence,Null,Non,Non			
3	3,2005-09-23,Null,SI,Non,Non			
4	4,2004-09-22,Master,IC,Null,Null			
5	5,2004-09-10,Licence,SI,Non,Oui			
6	6,1996-10-11,Null,SI,Oui,Non			
7	7,1998-10-22,Null,IC,Non,Non			
8	8,2004-10-03,Master,Null,Null,Null			
9	9,2010-09-21,Null,SI,Null,Oui			
10	10,2011-09-11,Master,SI,Oui,Oui			
11	11,1997-10-20,Null,SI,Null,Oui			
12	12,2009-09-12,Licence,SI,Oui,Oui			
13	13,1996-09-13,Null,IC,Oui,Null			
14	14,1999-10-05,Licence,SI,Oui,Non			
15	15,2003-10-08,Licence,IC,Non,Oui			

Figure 7: Données en format CSV.

- **Fichiers XML :**

  Un fichier XML (eXtensibleMarkupLanguage) décrit tout simplement un fichier texte contenant à la fois des données et des métadonnées, permettant de stocker des informations en respectant une structure donnée en utilisant des balises qui sont délimitées par les caractères "inférieur" et "supérieur". Un fichier XML est un moyen extrêmement efficace pour établir la communication entre les systèmes, puisque XML (et les schémas XML qui suivent) fournir suffisamment d'informations pour échanger les données entre eux, si par exemple l'entrepôt de données comprend des données qui vient desources externes de l'entreprise, ces sources seront fournies en XML.

❖ **Les systèmes ERP:**

L'ERP (Enterprise Ressources Planning) ou PGI (Progiciel de Gestion Intégré) est un progiciel centralisant les données et les fonctions de gestion de l'entreprise. Un système ERP comporte différents modules, correspondant chacun à une fonction de l'entreprise. Gestion financière et comptable, Gestion de production, Gestion des ventes, des achats, des stocks, des ressources humaines. Les données partagées sont centralisées et les interfaces standardisées [25].

❖ **Bases de données relationnelles:**

Une base de données est une collection de données organisées et classées de façon à être facilement accessibles, administrées et mises à jour via des systèmes de gestion de base de données (SGBDs).

5.2 Data Staging Area (DSA) :

Le DSA (Data Staging Area) est une zone de stockage dédiée pour la préparation de données, la structure de cette zone est semblable à celle des systèmes sources dont les données peuvent être stockées dans des fichiers plats ou des bases relationnelles. Le seul rôle de cette zone est de stocker ces données temporairement en attendant de les faire passer dans la moulinette de la transformation, afin de les rendre homogènes puis les charger dans l'entrepôt. Une fois le processus de chargement est terminé, les données sont supprimées de la zone intermédiaire. Néanmoins, parfois, les données sont préservées

pour soutenir les fonctionnalités de la phase transformation (T) qui nécessitent de l'historique, dans ce cas,cette zone est dite persistante (Persistent Staging Area).

❖ **Les raisons de l'utilisation du DSA:**

- Extract Once, transformmany : ce qui veux dire que l'on fait une seule extraction des systèmes sources, et autant de transformations dans le DSA.
- Ne pas impacter le fonctionnement des systèmes sources. En plus, le temps nécessaire à extraire les données, devrait être minimale, donc en général les gens ne font pas de transformation en même temps que l'extraction : On extrait les données le plus rapidement possible, on les mets dans le DSA, puis on transforme.
- Ou cas ou on a des problèmes de transformation (plantage lors de la transformation, erreur ...), on ne sera pas obligé de refaire l'extraction, du moment ou la source de la transformation est le DSA .
- Dans le cas ou on aura à intégrer les données de plusieurs sources, le DSA est en quelques sortes, un endroit de synchronisation, avant de commencer à transformer les données.[26]

5.3 Entrepôts de données (ED) :

L'ED (vu dans le chapitre I, Section 3) est une base de données architecturée pour des requêtes et des analyses, plutôt que pour le traitement transactionnel des données, et les résultats de ces requêtes doivent être obtenues rapidement. Il est souvent organisé sur le modèle multidimensionnel évoqué précédemment. Il y a néanmoins deux types de stockage :

- L'entrepôt (Data Warehouse) qui concentre toutes les données.
- Le magazin de données (Data Mart) qui se focalise sur une partie du métier, comme les relations clients, par exemple.[27]

L'entrepôt de données reçoit un flux de données entrant et fournit un flux de données sortant. Le flux entrant passe par un processus d'extraction des données à partir de sources multiples et hétérogènes. Ces données sont ensuite transformées, filtrées, homogénéisées et nettoyées avant d'être stockées dans l'entrepôt de données au niveau de la zone de préparation (DSA). La zone de stockage, représentant l'entrepôt de données, est évidemment une zone de stockage permanente des données. La zone de présentation donne accès aux données contenues dans l'entrepôt de données. Elle peut contenir des outils d'analyse programmés (rapports, requêtes, …).Le flux sortant met ces données à la

disposition des utilisateurs. La Figure 8 illustre l'architecture générale d'un entrepôt de données:

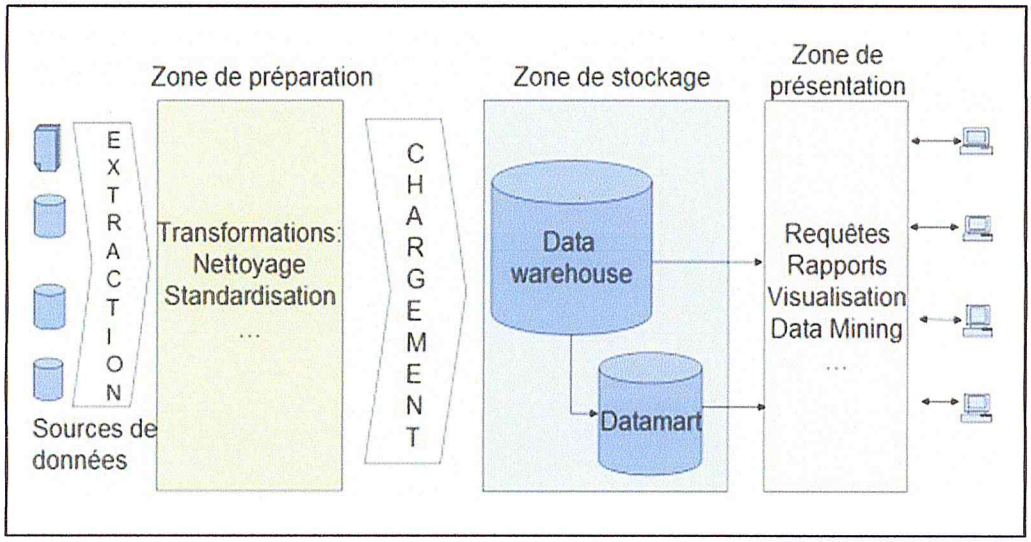

Figure 8: Architecture générale d'un entrepôt de données [27].

6. Conclusion

Dans ce chapitre, nous avons vu en détail le processus d'intégration de données dans un entrepôt de données, d'une manière globale à un niveau de granularité élevé en expliquant ses différentes étapes ainsi que l'ensemble des structures de données qu'il les utilise. Puis, nous avons définis ce dernier à un niveau plus fin en montrant ses fonctionnalités de base, ce qui nous permit par la suite de développer notre approche dans le prochain chapitre.

CHAPITRE III

PARALLELISATION DES FONCTIONNALITES D'ETL POUR L'INTEGRATION DES DONNEES MASSIVES

1. Introduction

Chaque jour, une grande quantité de données est produite grâce à l'utilisation d'internet, des réseaux sociaux, des images, des signaux GPS etc. Ces données seront incorporées dans les systèmes informatiques et par la suite passées par un processus d'intégration pour construire une mine de données pour les SIDs afin d'améliorer leur résultats d'analyses. Néanmoins, elles deviennent un véritable enjeu dû aux exigences en termes de rapidité de traitement, de capacité de stockage et d'hétérogénéité des sources, d'où l'émergence de plusieurs outils qui permettent de réaliser l'intégration de ces données massives en se basant sur des approches de distribution et de parallélisation du processus ETL selon le paradigme MapReduce afin de réduire les couts en termes de temps et de ressources.
Nous allons éxposer dans ce chapitre, le concept de données massives et le paradigme MapReduce puis nous présenterons un nouveau schéma adapté du processus ETL dans un environnement caractérisé par des données massives. Nous présenterons particulièrementle processus ETL à un niveau très fin en se basant sur ses fonctionnalités de base telles que Changing Data Capture (CDC), Surrogate Key Pipeline (SKP), Slowly Changing Dimension (SCD). Cette approche nous permettra de restructurer le processus ETL à travers ses fonctionnalités, ce qui permettra une meilleure robustesse, fiabilité et performance.

2. Les données massives(BigData)

La production croissante de données, le partage des informations entre utilisateurs, la diffusion des données via les réseaux engendrent de très gros volumes de données disponibles et intéressantes à analyser. L'expression anglaise Big Data est utilisée pour désigner des données dont le volume est tel qu'il devient difficile de les stocker, de les interroger, de les modéliser, de les analyser et de les visualiser avec les outils et architectures informatiques existants. L'utilisation de nouvelles unités de mesures de stockage, telles que les peta-octets voire les zeta-octets sont aujourd'hui des réalités. Outre le stockage, l'exploitation de telles données soulève également de nouveaux challenges c'est pourquoi de nombreux travaux de recherche proposent aujourd'hui des solutions de gestion de données à très grande échelle. [28]

Le Big Data est caractérisé par les quatre« V » :

- **Le volume** qui désigne la quantité inhabituelle de données.
- **La vélocité** qui est la vitesse avec laquelle les données sont générées, collectées et traitées en temps réel pour le but de les intégrer dans des processus métiers.
- **La variété** qui signifie la diversité des formats et des structures.
- **La véracité** qui est la fiabilité et l'exactitude des données.

La figure 9 présentede façon plus détaillé, les quatre dimensions de complexité deBig Data.

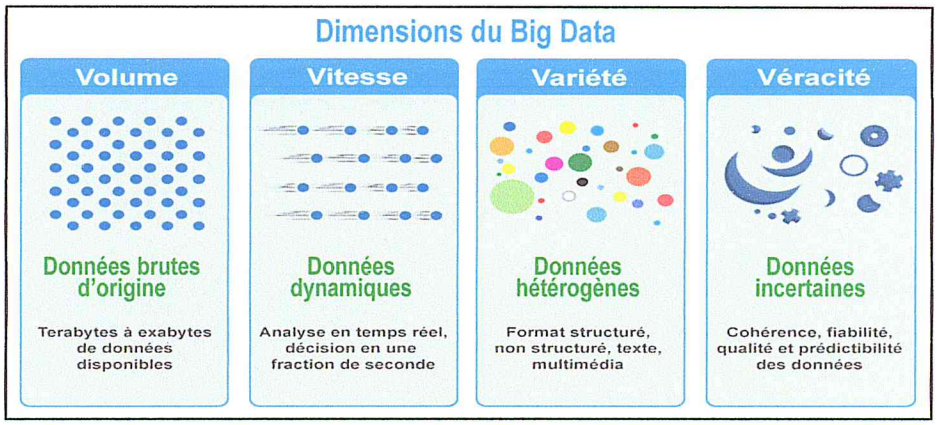

Figure 9: Les quatre principaux caractéristiques du BigData [29].

3. Le paradigme MapReduce « MR »

Il s'agit d'une stratégie qui gère la répartition et l'exécution des requêtes sur les données stockées par le cluster. L'objectif de MapReduce et de son mécanisme avancé de distribution de tâches est de tirer parti de la localité entre données et traitements sur le même nœud de façon à minimiser l'impact des transferts de données entre les nœuds du cluster au profit de la performance.

Il se compose de deux étapes :

- **Map**: Dans cette étape, le nœud maitre divise le problème posé en sous-problèmes et les distribue entre nœuds de traitement. Les réponses sont ensuite remontées de nœuds en nœuds jusqu'au nœud maitre ayant assigné les travaux à l'origine.
- **Reduce**: le nœud maitre collationne les réponses remontées à partir des nœuds de traitement et les combine afin de fournir la réponse à la question posée à l'origine. Entre ces deux étapes principales, il existe une phase appelée **Répartition (shuffle)** dans laquelle nous devons regrouper toutes les valeurs produites pour une même clé par les nœuds de calcul. La phase de Reduce n'aura plus qu'à les agréger.

La figure 10 explique le principe de fonctionnement du modèle MapReduce :

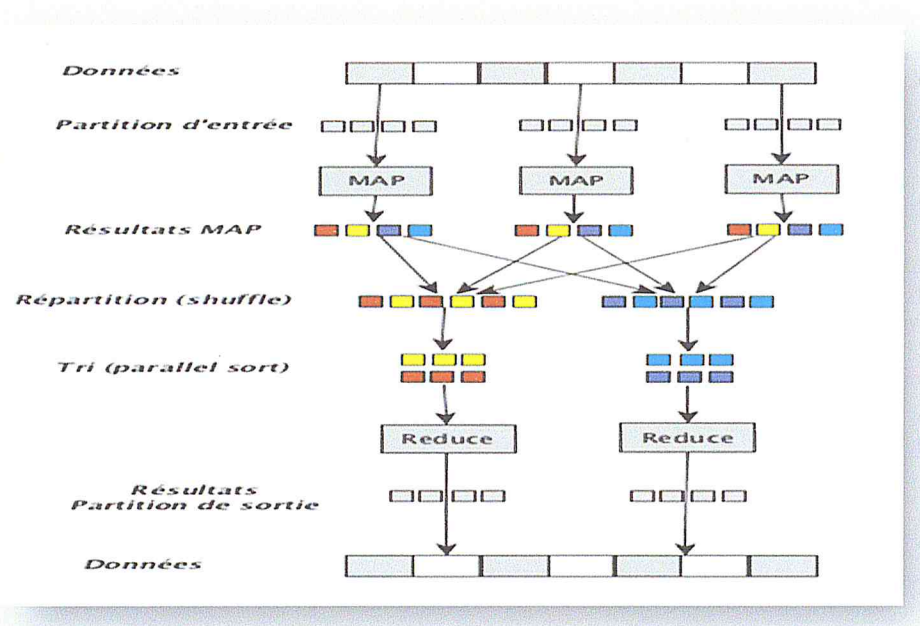

Figure 10 : Fonctionnement de MapReduce [30].

4. Approche ETL Classique versus ETL Distribué

Lorsque le processus ETL traite une quantité raisonnable de données, sur une seule machine et en une seule instance est dit processus ETL classique. Dans ce cas, seulement les fonctionnalités indépendantes peuvent s'exécuter en parallèle.

Cette approche n'est plus efficace due à l'émergence de BigData, pour cela plusieurs chercheurs ont pensé à la parallélisation du processus selon le paradigme MapReduce qui consiste à diviser l'ensemble des données en plusieurs partitions, affecter à chaque instance du processus ETL une partition dans le but de les traiter en même temps sur le cluster (ensemble des machines).

5. Parallélisation des fonctionnalités d'ETL selon le paradigme MapReduce

La majorité des travaux dans la littérature, définissent la problématique d'intégration des données de manière globale à un niveau de granularité élevé du processus ETL. Nous nous intéressons, dans ce travail, à la parallélisation et à la distribution du processus ETL à un niveau de granularité très fin en se basant sur ses fonctionnalités de base selon le

paradigme MapReduce. Cependant, cette fois nous la déployons au niveau fonctionnalité, ce qui permettra une meilleure robustesse, fiabilité et performance.

Afin de permettre aux fonctionnalités d'ETL de fonctionner dans un environnement Big data, nous proposons une nouvelle architecture et des algorithmes basés sur le paradigme MapReduce. Nous nous intéressons particulièrement aux fonctionnalités suivantes: CDC (changing Data Capture) pour la phase d'extraction, SKP(Surrogate Key Pipeline) pour la phase de transformation et SCD(SlowlyChanging Dimension) pour la phase de chargement

.

5.1 CDC dans un environnement parallèle/distribué :

Changing Data Capture (CDC) étant la première fonctionnalité sur laquelle se base la phase d'Extraction (E) d'un processus ETL. Lorsque les données changent au niveau des systèmes sources, ce sont ces changements qui nous intéressent pour le rafraichissement de l'entrepôt de données. C'est la vocation de la fonctionnalité CDC. Ils existent plusieurs politiques qui permettent de les détecter (déjà citées dans le chapitre précédant), nous nous intéressons à celle qui utilise des captures instantanées (Snapshots).Comme le montre l'exemple suivant, la table source TS contient la version récente des données (snapshot j), la table notée TSvp contient leur version précédente (snapshot j-1). Le tuple 01 a connu une modification donc il sera capturé comme modification (UPDATE), le tuple 01sera rejeté par CDC puisqu'il ne présente aucun changement, le tuple 03 est considéré comme suppression (DELETE) puisqu'il apparait dans TSvp mais n'apparait plus dans TS et le tuple 05 sera capturé comme nouveau (INSERT).

Exemple :

Table TSvp (Snapshot j-1)

#Matricule	cycle	Spécialité	Bourse
01	licence	SI	oui
02	master	GSI	non
03	licence	IL	non

Table TS (Snapshot j)

#Matricule	cycle	Spécialité	Bourse
01	licence	SI	non
02	master	GSI	non
05	master	IC	oui

Changements

#Matricule	cycle	Spécialité	Bourse	Changement
01	licence	SI	oui	modification
03	licence	IL	non	suppression
05	master	IC	oui	insertion

Figure 11 : Exemple illustrant le fonctionnement du CDC.

❖ **Identification des changements:**

Afin d'identifier les changements entre deux tuples correspondants nous adaptons une fonction de hachage appelée CRC (CyclicRedundancy Check) de la manière suivante: Etant donnés deux tuples tuple1 et tuple2 stockés respectivement dans les tables ST et STpv. Si tuple1 et tuple2 satisfont les deux équations 1 et 2, ceci implique que les deux tuples sont similaires. En revanche, si uniquement l'équation 1 est vérifiée, ceci exprime que le tuple1 a été affecté par des changements et devra alors être extrait par CDC.

Equation 1: tuple1.KEY = tuple2.KEY

Equation 2: CRC (tuple1) = CRC(tuple2)

5.1.1 Partitionnement des données :

Pour faire face au gros volume de données de la table TS, nous adoptons la règle « diviser pour régner» qui consiste à partitionner la table TS et la table TSvp de manière à obtenir des volumes de données habituels. La table source TS sera chargée dans le DSA et partitionnée pour permettre un traitement parallèle des partitions ainsi générées. La table TSvp, considérée aussi volumineuse, sera partitionnée pour éviter de rechercher les tuples de TS dans un grand volume de données. Nous choisissons la technique de **Partitionnement Simple** qui consiste à diviser le volume initial V en n partitions approximativement égales comme le montre la figure12.

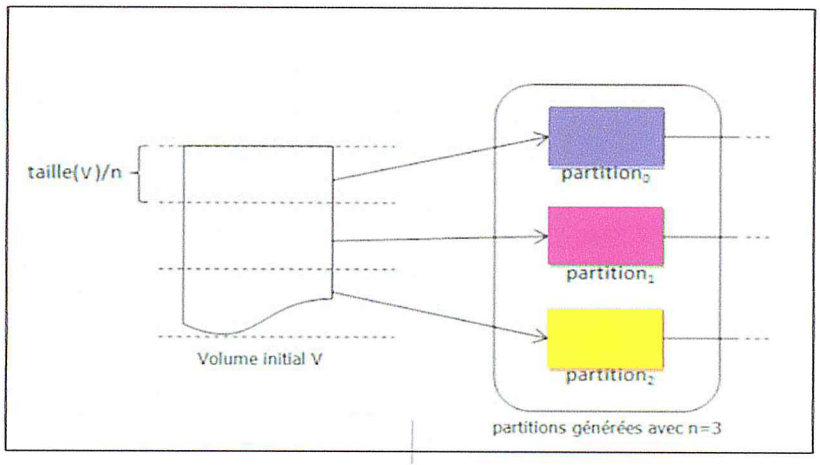

Figure 12 : Technique de partitionnement simple.

5.1.2 Tables Lookup :

Lorsqu'un tuple de TS contenu dans une partition Ptsi doit être recherché dans TSvp, nous n'avons aucune idée sur la partition de TSvp pouvant contenir celui-ci.

Pour éviter de rechercher dans toutes les partitions de TSvp et dans toutes les partitions de TS, nous utilisons des tables Lookup avec les principes suivants :

- LookupTSvp et LookupTS contiennent, respectivement, les valeurs min et max des clefs naturelles (#NK) de chaque partition de TSvp et de TS ;
- Pour un tuple Ti de TS, il s'agit de rechercher dans LookupTSvp la partition Ptsvpk de TSvp vérifiant **l'expression 1** ;

$$Lookup.TSvp.NKmin \leq T\ i.NK \leq LookupTSvp.NKmax \quad (1)$$

- Il est possible qu'un tuple Ti de TS, vérifiant bien l'expression 1, n'existe pas dans Ptsvpk. Il s'agit, dans ce cas, d'un nouveau tuple inséré dans la table source TS ;
- Pour un tupleTj de TSvp, il s'agit de rechercher dans LookupTS la partition Ptsk de TS vérifiant **l'expression 2** ;

$$Lookup.TS.NKmin \leq T\ j.NK \leq LookupTS.NKma(2)$$

- Il est possible qu'un tupleTj de TSvp, vérifiant bien l'expression 2, n'existe pas dans Ptsk. Il s'agit, dans ce cas, d'un tuple supprimé dans la table source TS.

5.1.3 Processus IUDCP et DDCP :

Nous proposons deux processus parallèles pour la capture des données (CDC) : un processus de capture des insertions et modifications (IUDCP) et un processus de capture des suppressions (DDCP).

❖ **IUDC Process : Traitement parallèle des partitions TS**

La figure 13 décrit le processus IUDC. Chaque partition Ptsi est confiée à une tâche (Mapi) qui est chargée de vérifier, pour chacun de ses tuples, l'existence ou non de celui-ci sur la table TSvp (structurée en partitions).

Pour ce faire, le mapper passe par la table LookupTSvp pour identifier la partition de TSvp pouvant contenir le tuple en question.

Dès que la partition Ptsvpi est identifiée, trois cas peuvent se présenter :

- #NK inexistante dans Ptsvpi : la tâche prend en considération le tuple pour insertion (INSERT) ;
- #NK existe avec une copie similaire du tuple dans Ptsvpi : la tâche rejette le tuple puisqu'aucun changement n'a eu lieu ;

- #NK existe dans Ptsvpi avec des changements: la tâche prend en considération le tuple pour modification (UPDATE).

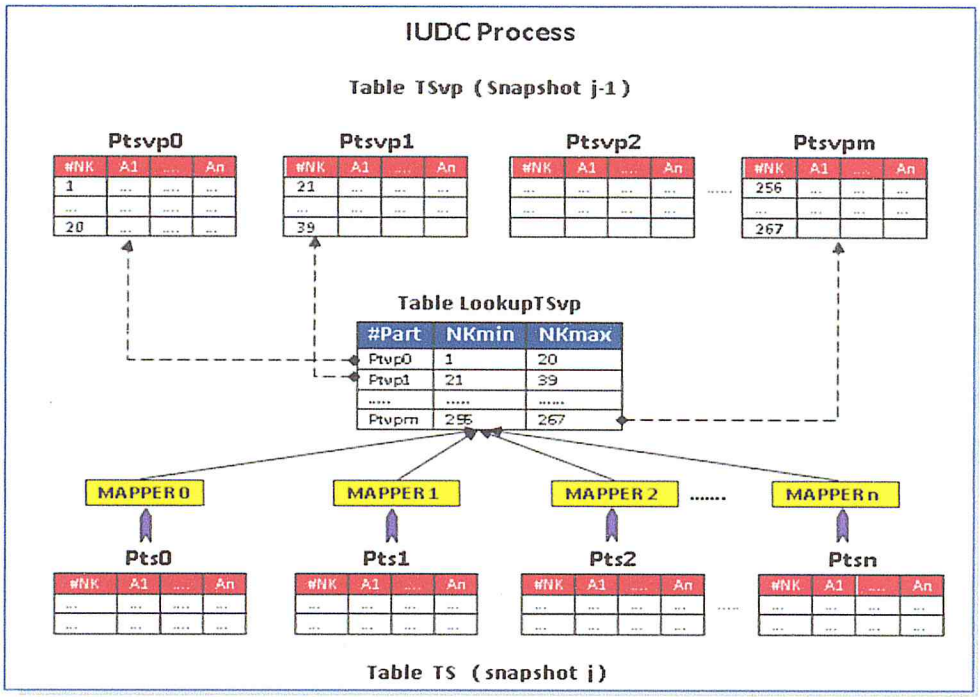

Figure 13: Architecture distribuée du processus IUDC.

❖ **DDC Process : Traitement parallèle des partitions de TSvp**

Dans IUDCP deux mappers traitant deux partitions différentes Ptsi et Ptsj peuvent être orientés par LookupTSvp vers une même partition Ptsvpk. Ainsi, un même tuple de celle-ci pourra alors être capturé plusieurs fois comme suppression. C'est la raison pour laquelle, nous avons proposé un autre processus appelé DDCP. Comme le montre la figure 14, chaque partition Ptsvpi est confiée à une tâche Mapi chargée de vérifier, pour chacun de ses tuples, l'existence de celui-ci dans TS. Pour ce faire, le mapper passe par LookupTS pour identifier Ptsk de TS qui pourra contenir le tuple et dès que celle-ci est identifiée, nous ne retenons que le cas où le tuple est supprimé.

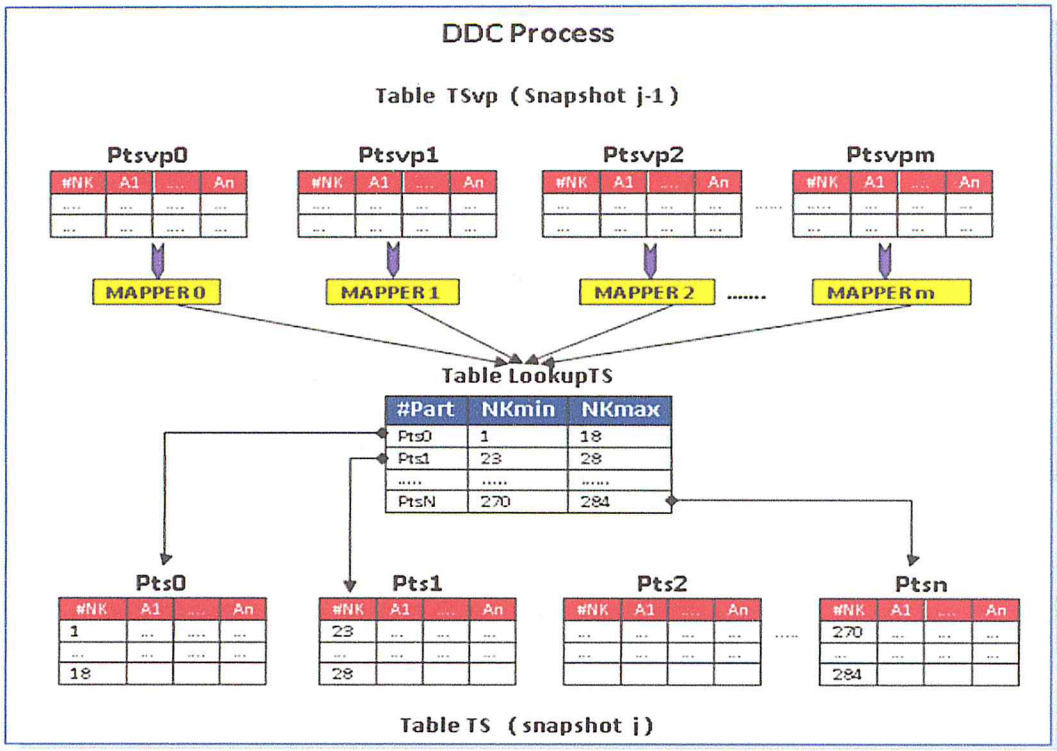

Figure 14: Architecture distribuée du processus DDC.

5.1.4 Algorithmes :

Le programme principal CDC_BigData consiste à partitionner TS et TSvp, générer LookupTS et LookupTSvp et enfin lancer, en parallèle, IUDCP et DDCP.

- **L'algorithme 1 (Processus distribué IUDCP)** montre comment IUDCP affecte les partitions de TS au processus MapReduce qui s'exécutera en un nombre d'instances égal au nombre de partitions de TS. L'algorithme décrivant DDCP fonctionne avec le même principe.

```
   Entrées :
1  LookupTS : table lookup de TS,
2  VT : enregistrement de type LookupTS,
3  i : entier,
   Sortie :
4  CHANGES : Liste des tuples de TS affectés par des changements,
5  début
6      ouvrir (LookupTS) ;
7      i ← 1 ;
8      tantque non fin (LookupTS) faire
9          Lire (LookupTS, VT) ;
10         soumettre VT.Part à iu_mapreader(i) ;
11         i ← i + 1 ;
12     fait
13     iu_map() ; // appel de la fonction iu_map()
14 fin
15 retourner (CHANGES) ;
```

Algorithme 1 : Processus distribué IUDCP.

❖ **L'algorithme 3(D_MAP) : Tâches de capture de suppression de données**

Il est chargé de capturer les suppressions. Une partition Ptsvpi sera traitée par une instance de D_map(). Les lignes 12-16 localisent la partition Ptsi pouvant contenir le tuple lu dans ligne 10. Les lignes 16-21 traitent le cas où la partition Ptsvpk est localisée. Si celle-ci ne contient pas le tuple (ligne 22), il sera capturé comme suppression (ligne 23). Les lignes 25-27 capturent comme suppression les tuples où LookupTS montre qu'ils n'apparaissent dans aucune partition de TS.

```
   Entrées :
1  Ptsvp : Partition de TSvp,
2  LookupTS : Table Lookup de TS,
3  VT1 : enreg. de type TSvp,
4  VT2 : enreg. de type LookupTS,
5  VT3 : enreg. de type Pts,
   Sortie :
6  CHANGES : Liste des tuples de TS affectés par des changements,
7  début
8      ouvrir (Ptsvp) ;
9      tantque non fin (Ptsvp) faire
10         lire (Ptsvp, VT1) ;
11         ouvrir (LookupTS) ;
12         lire (LookupTS, VT2) ;
13         tantque non fin (LookupTS) et (VT1.NK > VT2.NKmax) faire
14             lire (LookupTS, VT2) ;
15         fait
16         si non fin (LookupTS) alors
17             ouvrir (VT2.Part) ;
18             lire (VT2.Part, VT3) ;
19             tantque non fin (VT2.Part) et (VT1.NK > VT3.NK) faire
20                 lire (VT2.Part, VT3) ;
21             fait
22             si VT1.NK < (VT3.NK) alors
23                 capturer le tuple VT1 comme suppression (delete) ;
24             fin
25             sinon
26                 capturer le tuple VT1 comme suppression (delete)
27             fin
28         fin
29     fait
30     retourner (CHANGES)
31 fin
```

Algorithme 3 : D_MAP : Tâches de capture de suppression de données.

5.2 SKP dans un environnement parallèle/distribué :

Avant de procéder au chargement des données de la table de fait, chacun de ses tuples doit passer par le processus SKP qui consiste à rechercher, pour chaque valeur de ses clefs NKi, la valeur de la clef SKi correspondante dans une table Lookupi. Pour chaque clef NKi, est associée une table d'index Lookupi générée à partir de dimensioni correspondante contenant les valeurs de NKi avec la valeur la plus récente de SKi correspondante. Voici un exemple illustrant le principe de SKP.

Exemple:

La table de fait TF_Etud permet d'analyser la mesure effectif d'étudiants par rapport aux dimensions Spécialité, Cycle et Année (voir figure 15).

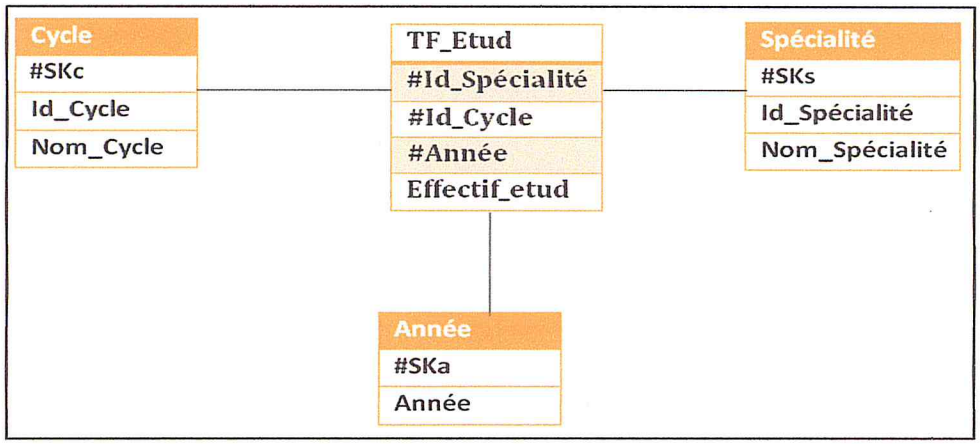

Figure 15 : Exemple d'un schéma en étoile pour l'analyse de l'effectif des étudiants.

Nous présentons dans la figure 16 le traitement de trois tuples sources de la table TF_Etud par le processus SKP afin de remplacer toutes les clefs NK par SK, la table TF_Etud est chargée par la suite dans l'entrepôt.

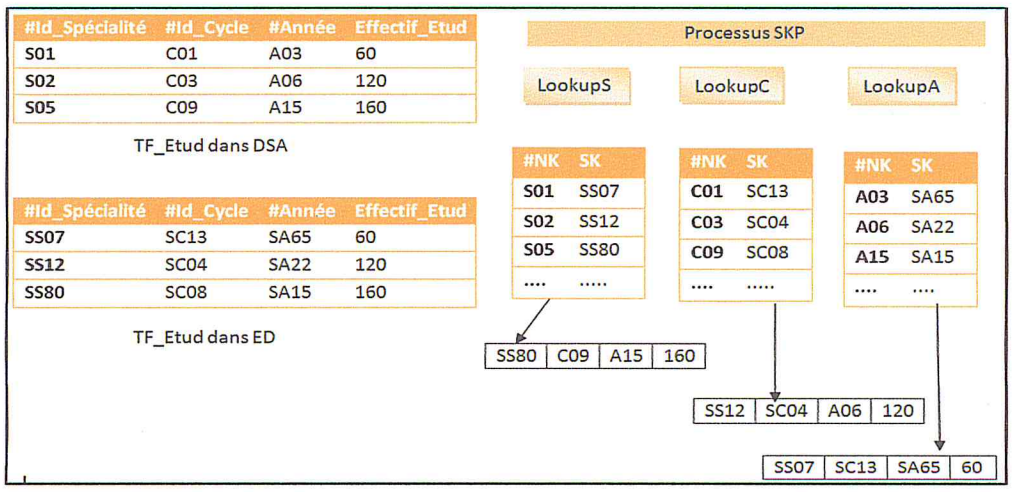

Figure16 : Exemple du processus SKP.

5.2.1 Principe de processus SKP :

Dans le context de Big Data nous proposons d'adapter le processus SKP selon le modèle MapReduce. Notre contribution consiste à:

(1) Partitionner la table de fait TF en plusieurs volumes appelés partitions;

(2) Traiter chaque partition de TF, dans une phase Map, par une instance du tunnel SKP (mapper) de façon parallèle;

(3) Stocker les résultats partiels (partitions de TF avec SK) dans la zone locale du mapper;

(4) Fusionner, dans la phase Reduce, grâce à plusieurs instances (reducers), les différents résultats obtenus par les mappers et leur chargement dans la table de fait cible au sein du DW.

La figure 17 récapitule les principes de notre approche.

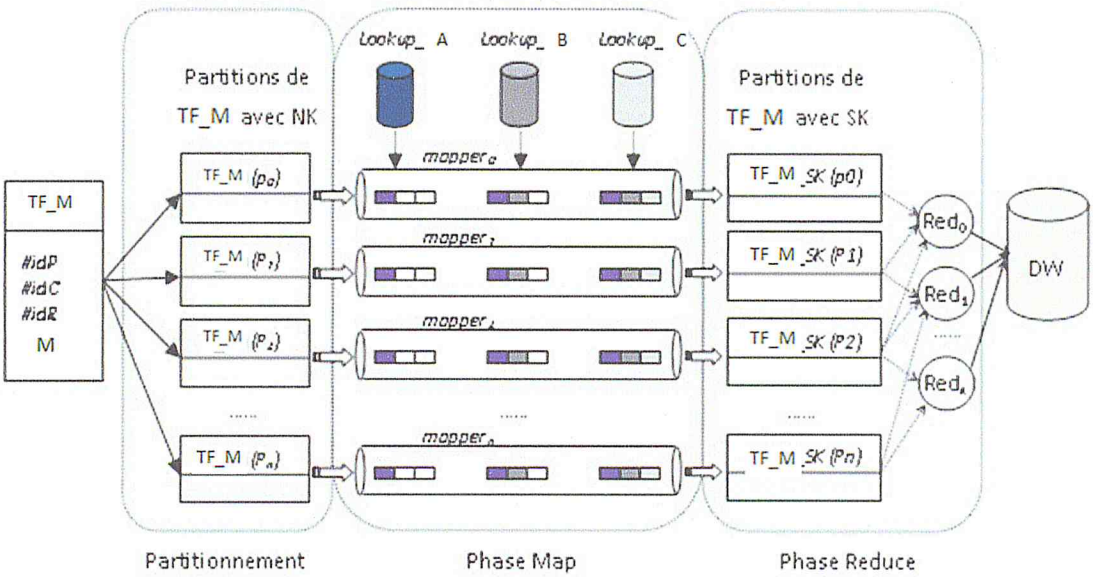

Figure 17: Architecture distribuée du processus SKP.

5.2.2 Algorithmes :

Dans ce qui suit nous proposons les algorithmes décrivant le fonctionnement de SKP dans le modèle MapReduce. Ces algorithmes sont instanciables (plusieurs exécutions de la même fonction en même temps) où chaque instance traite sur une partition de la table de fait.

L'algorithme 4 (SKP_BigData) montre comment le processus SKP_BigData pousse les partitions de TF_NK vers le processus MapReduce qui s'exécutera en un nombre d'instances égal au nombre de partitions de TF_NK.

```
Entrées : TFNK : Table de fait TFNK avec les clefs NK
N : nombre de tables Lookup
1 début
2 |  Ouvrir (PTFNK) ;
3 |  Partitionner (TFNK, N) ; // partitionner TFNK en un nombre de partitions égal à N
4 |  Pour i ← 1 à N Faire
5 |  |  mapper_SKP (PTFNK_i) // lancer une instance mapper_SKP_i pour traiter la partition
  |  |  PTFNK_i
6 |  FinPour
7 fin
```

Algorithme 4 :SKP_BigData.

L'algorithme 5 (mapper_SKP(PTFNK))décrit le programme principal de SKP. Il consiste à charger les tables TLookup1, TLookup2 et TLookup3 en mémoire cache pour un accès rapide, lire à partir de la table TFNK en bloc pour accélérer le processus SKP et Déclencher le processus SKP en appelant la première procédure mapper_Lookup1 pour le remplacement de NK1 par SK1.

```
Entrées : PTFNK : Partition de la table de fait TFNK avec les clefs NK.
1  début
2  |  Variables VT1, VT2, VT3 : enregistrements de type TFNK
3  |  // charger les tables d'index Lookup en cache pour accélérer SKP
4  |  mise en cache de TLookup1, TLookup2, TLookup3
5  |  Ouvrir (PTFNK) ;
6  |  tantque non fin (PTFNK) faire
7  |  |  // charger les tuples de PTFNK par bloc dans buffer pour accélérer SKP
8  |  |  Buffer ← bloc(PTFNK) ; // charger dans buffer un nombre de tuples multiple de 3
9  |  |  tantque non fin (Buffer) faire
10 |  |  |  Lire (Buffer, VT1) ; Lire (Buffer, VT2) ; Lire (Buffer, VT3) ;
11 |  |  |  mapper_Lookup1 (VT1) ; mapper_Lookup1 (VT2) ; mapper_Lookup1 (VT3) ;
12 |  |  fait
13 |  fait
14 fin
```

Algorithme 5: mapper_SKP(PTFNK).

L'algorithme 6 (Reduce( )) décrit la fonction reduce( ) chargée de la fusion des résultats obtenus par les mappers et du chargement des tuples dans la table de fait du ED. Puisque la taille des données à charger est importante, la phase Reduce doit être prise en charge parplusieurs reducers de façon parallèle. Chaque reducer lance une fonction appelée shuffle () chargée de récupérer les tuples qui lui sont destinés pour opérer la fusion et le chargement.

```
1 début
2 |  Buffer_reducer ← shuffle() // récupérer les tuples à partir des partitions des mappers
3 |  // charger le contenu du buffer, en bloc, dans la table de fait du DW
4 |  TFSK(DW) ← BULK LOAD (Buffer_reducer)
5 fin
```

Algorithme 6 : Reduce( ).

5.2.3 Génération des tables Lookup :

Lorsqu'il s'agit de grandes dimensions la recherche de SK dans celle-ci devient complexe sachant que (1) Une même clef NK apparait plusieurs fois avec des valeurs de SK différentes à cause de versionnement et (2) la dimension peut contenir des dizaines voire des centaines d'attributs en plus des clefs NK et SK.

Une table Lookup représente une forme compressée de la dimension correspondante. En termes de colonnes, elle n'en contient que deux (NK, SK) et en termes de lignes, elle ne contient que les tuples avec la valeur de SK la plus récente. Les tables Lookup doivent être mises à jour de manière continue suite aux insertions de nouveaux tuples dans les dimensions correspondantes.

La figure 18 montre une dimension Etudiant dont Matricule représente la clef naturelle NK et SK la clef de substitution, à droite la table Lookup correspondante qui contient seulement les clefs NK et SK dont les valeurs des clefs SK sont les plus récentes.

Table e dimension Etudiant

#SK	Matricule	cycle	Spécialité	Bourse
....	.....	.....	.....	.....
SK09	M03	.....	.....	.....
SK13	M01	.....	.....	.....
SK14	M05	.....	.....	.....
SK28	M01	.....	.....	.....
.....	.....	.....	.....	.....
SK60	M05	.....	.....	.....

Table e Lookup

Matricule	SK
.....	.....
M01	SK28
M03	SK09
.....	.....
M05	SK60
.....	.....

Figure 18: Exemple de table de dimension et table lookup.

L'algorithme 7 (Generer_TLookup) décrit le chargement initial à partir d'une table de dimension vers sa table Lookup.

Entrées : $TDim$: Table de dimension
Sortie : $TLookup$: Table Lookup à générer

```
1 début
2 |  INSERT INTO TLookup
3 |  SELECT NK, MAX(SK) as SK FROM TDim GROUP BY NK
4 fin
```

Algorithme 7 : Generer_TLookup.

Après le chargement initial dans la table Lookup, celle-ci doit être mise à jour suite aux ajouts des nouveaux tuples et par conséquent de nouvelles valeurs de SK. Nous définissons un déclencheur (figure 19) qui fait appel au **algorithme 11 (MAJ_TLookup)** pour faire ceci.

```
CREATE TRIGGER trigger_TLookup
AFTER INSERT
    ON  TDim
    FOR  EACH ROW
BEGIN
    CALL  MAJ_TLookup
END;
```

Figure 19 :Déclencheur pour mettre à jour une table Lookup.

5.3 SCD dans un environnement parallèle/distribué :

Malgré que les tables de dimension d'un entrepôt de données sont rarement mises à jour, elles méritent une importance particulière puisque la dimension décrit le contexte du contenu de la table de fait. Lorsque une table de dimension subit une mise à jour Plusieurs stratégies de versionnement des tuples d'une dimension existent (traitées dans le chapitre précédent section 4.2), nous nous intéressons particulièrement aux trois types (i) SCD type 1, (ii) SCD type 2 et (iii) SCD type4.

Exemple : Les attributs SCD type 1 seront écrasés s'ils subissent une modification, par contre les attributs type 4 nécessitent une historisation (versionnement) . Pour ce faire, au moment d'une mise à jour l'ancien est dupliqué avec la même clé naturelle (NK) et sera

affecté au table d'historisation TDH. Le nouveau tuple lui est affecté une nouvelle valeur pour sa clé de substitution (SK). C'est dans ce nouveau tuple que sera opérée la mise à jour sur les attributs de type SCD 4.

Soit la table de dimension TD suivante possédante comme attributs: cycle, spécialité et bourse soumises aux règles SCD type 4, SCD type 1 et SCD type1respectivement, la table TS est la table source et TDH est la table d'historisation. La figure 20 montre les changements affectés aux TD et TDH, lorsqu'on utilise la couleur verte il s'agit d'un écrasement (attribut du type SCD1), et le rouge signifie une affectation de l'ancienne version de tuple au TDH et le mettre à jour au niveau de la table TD (attribut du type SCD 4).

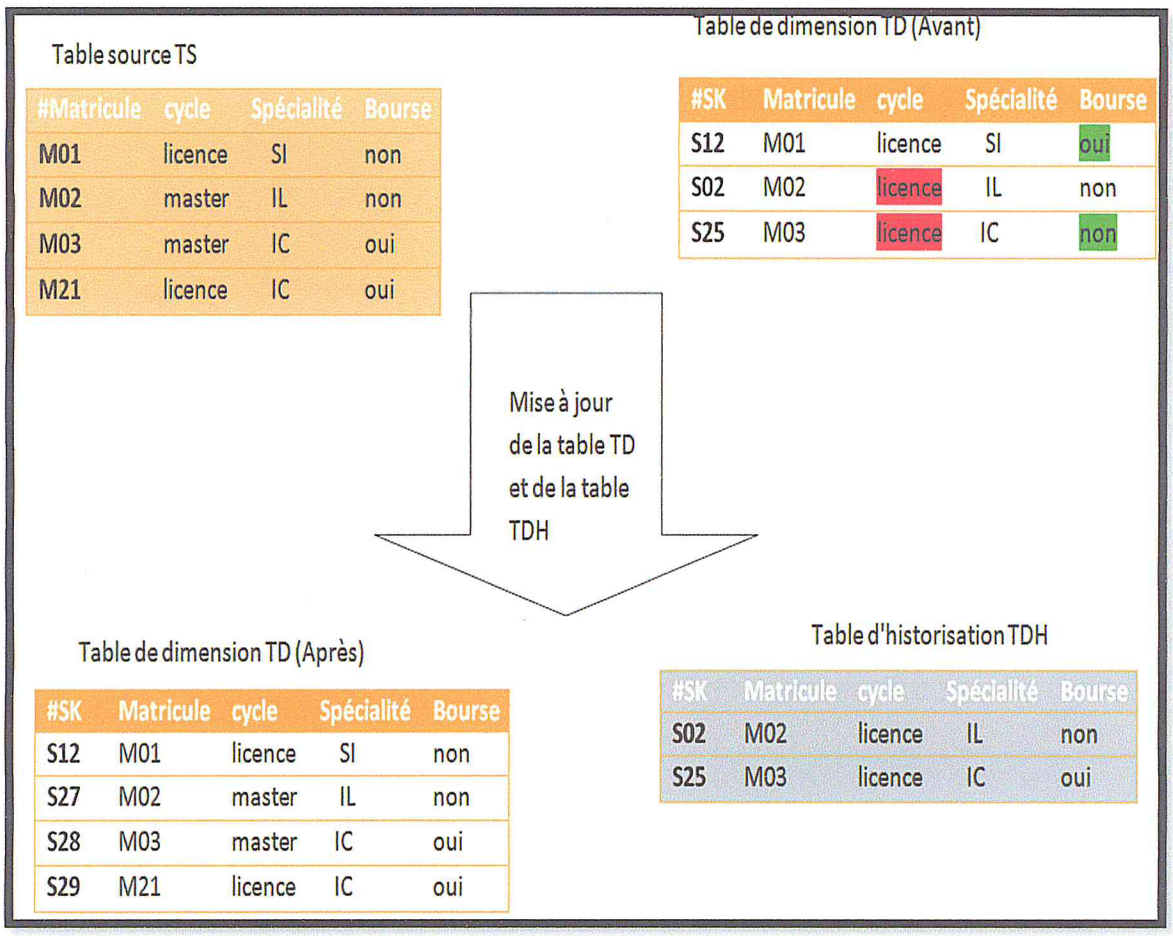

Figure 20 : Processus SlowlyChanging Dimension (SCD), Types 1 et 4.

5.3.1 Principe de processus SCD :

Nous proposons pour le processus SCD une architecture parallèle/distribuée selon le paradigme MapReduce. Ainsi, la table source TS après avoir subi les transformations nécessaires au sein du DSA, sera partitionnée. Chaque partition PTSi sera soumise à un mapper chargé d'opérer les mises à jour au sein de la table de dimension TD selon les approches SCD des attributs concernés. La figure 21 décrit l'architecture de SCD dans un environnement distribué.

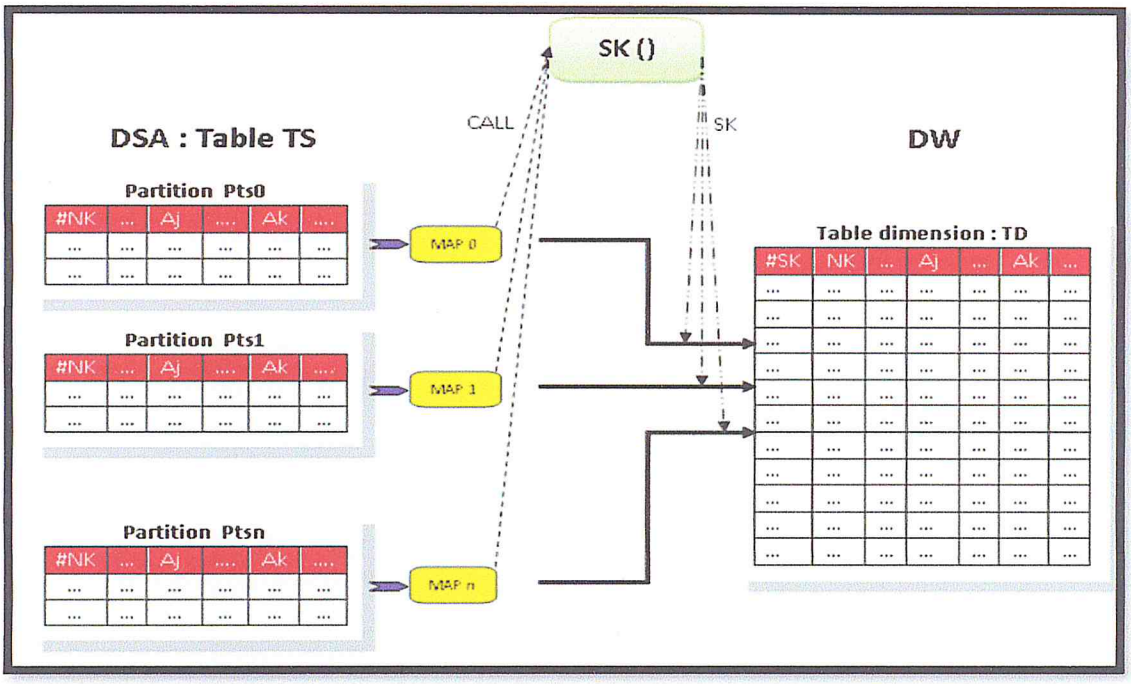

Figure21 : Architecture distribuée du processus SCD.

❖ **La table TSCD :**

Le processus SCD doit disposer de métadonnées sur les attributs des dimensions afin d'associer chaque attribut pouvant être concerné par des mises à jour à l'approche SCD qui lui correspond. Pour ce faire, nous proposons une table nommée TSCD dont la structure est décrite par le tableau 4.

Dimension	Attribut	Type SCD
Etudiant	Cycle	4
Etudiant	Bourse	1
Spécialité	Nom_spécialité	1
Spécialité	Département	4

Tableau 4: Table TSCD : Approches SCD appliquées aux attributs des dimensions.

5.3.2 Algorithmes :

Afin de faire fonctionner SCD selon cette architecture, nous proposons deux algorithmes. **L'algorithme 8 (SCD_BigData)** est le programme principal de SCD qui, à partir de la table TS (ligne 5), lance un partitionnement et génère des partitions de tailles plus petites (ligne 6). Les lignes 7-9 consistent à affecter chacune des partitions à une instance de **mapper_SCD( ) (algorithme 9)** et à déclencher ainsi le processus MapReduce.

Entrées :
1 TS : Table source (dans DSA) après transformation
Sortie :
2 TD : Table de dimension de base actualisée,
3 TDH : Table de dimension historique actualisée,
4 **début**
5 Ouvrir (TS) ;
6 Partitionner (TS, N) ; // partitionner TS en un nombre de partitions égal à N
7 **Pour** $i \leftarrow 1$ *à* N **Faire**
8 mapper_SCD (PST_i) // lancer l'instance $mapper\_SCD_i$ chargée de traiter la partition PST_i
9 **FinPour**
10 **retourner** (TD) ;
11 **retourner** (TDH) ;
12 **fin**

Algorithme 8:SCD_BigData.

lgorithme 9 (mapper_SCD) décrit les stratégies SCD type 1 et SCD type 4.

```
   Entrées :
1  PTS : Partition de la table source (dans DSA) après transformation,
2  TD : table de dimension (dans DW),
3  TDH : Table Historique de TD (dans DW),
4  VT1, VT2, VT3 : enregistrements de type respectivement PTS, TD et TD ,
5  VSK : Integer,
6  VTYPE, VT : entier,
   Sortie :
7  TD : Table de dimension de base actualisée,
8  TDH : Table de dimension historique actualisée,
9  début
10 |  VTYPE ← 1
11 |  OUVRIR (PTS) ;
12 |  tantque non fin (PTS) faire
13 |  |  LIRE (PTS, VT1) ;
14 |  |  si EXISTS (SELECT * FROM TD WHERE TD.NK=VT1.NK) alors
15 |  |  |  VT2 ← SELECT * FROM TD WHERE (TD.NK=VT1.NK)
16 |  |  |  VT3 ← VT2 // VT3 est une variable de travail
17 |  |  |  Pour CHAQUE attribut de VT1 et VT2 Faire
18 |  |  |  |  si VT1.Attribut ≠ VT2.Attribut alors
19 |  |  |  |  |  VT ← SELECT TypeSCD FROM TSCD WHERE SCD.Dimension='TD'
   |  |  |  |  |  and SCD.attribut=VT2.attribut
20 |  |  |  |  |  si VT = 1 alors
21 |  |  |  |  |  |  UPDATE TDH SET TDH.attribut=VT1.attribut where
   |  |  |  |  |  |  TDH.NK=VT1.NK ;
22 |  |  |  |  |  |  sinon
23 |  |  |  |  |  |  |  VTYPE ← VT ; //VTYPE contiendra le plus grand type SCD
24 |  |  |  |  |  |  fin
25 |  |  |  |  |  fin
26 |  |  |  |  |  VT3.attribut ← VT1.attribut ;
27 |  |  |  |  fin
28 |  |  |  |  suivant (VT1, attribut) ; suivant (VT2, attribut) ;
29 |  |  |  FinPour
30 |  |  |  si VTYPE=4 alors
31 |  |  |  |  INSERT (TDH, VT2) ; // insérer la version précédente du tuple dans TDH
32 |  |  |  |  DELETE (TD, VT2) ; // supprimer la version précédente du tuple à partir de TD
33 |  |  |  |  VT3.SK ← SK (TD) ; INSERT (TD, VT3) ; // insérer la nouvelle version du tuple dans TD
34 |  |  |  fin
35 |  |  |  sinon
36 |  |  |  |  VT3 ← VT1 ; VT3.SK ← SK (TD) ; INSERT (TD, VT3) ; // Il s'agit d'un nouveau
   |  |  |  |  membre à insérer dans TD
37 |  |  |  fin
38 |  |  fin
39 |  fait
40 |  retourner (TD) ;
41 |  retourner (TDH) ;
42 fin
```

Algorithme 9 : mapper_SCD.

A partir d'une partition Ptsi de TS, l'instance mapper_SCDi consiste à rafraichir une partie de la table TD et, en même temps, à historiser dans TDH les anciennes versions des tuples. Les lignes 14-29 identifient, pour un tuple source de Pts, celui qui lui correspond dans la table TD (lignes 14-15) et contrôlent l'existence de changements ayant affecté le tuple source (lignes 17-29). Si un des attributs concernés par les changements est de type SCD 1, une mise à jour de celui-ci sur tous les tuples correspondants (même NK) dans TDH est opérée (ligne 21). La ligne 23 consiste à retenir les changements détectés sur le tuple source pour opérer la mise à jour appropriée à la situation qui se présente. Les lignes 30-34 traitent le cas où au moins un attribut SCD4 a été concerné par des changements, auquel cas une historisation de la version précédente du tuple de TD dans TDH s'impose avant d'insérer sa nouvelle version dans TD. Les lignes 35-37 traitent le cas d'un nouveau tuple source (inexistant dans T D) qui doit être inséré dans TD.

6. Conclusion

Le processus ETL est la moulinette qui a pour vocation la préparation des données afin de les rendre aisément consommable par le système décisionnel, donc il est nécessaire de le renforcer par des approches consistantes pour qu'il puisse s'adapter dans le contexte de Big Data. Dans ce chapitre nous avons présenté notre approche de parallélisation/distribution de processus ETL à un niveau de granularité très fin vu que nous avons proposé des architectures et des algorithmes pour ses fonctionnalités de base (particulièrement: CDC, SKP et SCD) selon le paradigme MapReduce afin de réduire sa complexité et d'améliorer les performances.

Le chapitre suivant est consacré pour la spécification des besoins, la conception et la modélisation de notre système.

CHAPITRE IV

CONCEPTION DE SYSTEME

1. Introduction

La réalisation d'un projet informatique doit être accompagnée par un processus qui puisse garantir le développement efficace de logiciels de qualité, valable quel que soit la grandeur et la complexité du projet, et présentant de bonnes pratiques adaptées à la méthode en question, surtout que, de nos jours, les logiciels demandés sont de plus en plus imposants et exigeants qu'auparavant. Le processus unifié semble être la solution idéale pour les développeurs.

A la base de processus unifié qui utilise le langage UML (Unified Modeling Language) nous avons présenté la spécification et l'analyse des besoins, en deuxième étape nous avons établi la conception de notre système en exposant son architecture générale puis les diagrammes de séquences des cas importantes.

2. Langage de modélisation UML

UML (UnifiedModelingLanguage), se définit comme un langage de modélisation graphique et textuel destiné à comprendre et à définir des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. UML modélise l'ensemble des données et des traitements en élaborant des différents diagrammes. [31]

En clair, le langage UML ne préconise aucune démarche, qui propose un enchaînement d'étapes et d'activités qui mènent à la résolution d'un problème posé, mais plutôt comme une boite d'outils qui sert à améliorer les méthodes de travail, ce n'est donc pas une méthode. D'où chacun est libre d'utiliser les types de diagramme qu'il souhaite, dans l'ordre qu'il veut. Il suffit que les diagrammes réalisés soient cohérents entre eux, avant de passer à la réalisation du logiciel, nous dans le cadre de notre projet, nous avons utilisé les diagrammes des cas d'utilisations, de séquence et de classes.

3. Etude de cas

L'étude de cas traite sur un établissement universitaire où nous nous intéressons particulièrement aux données relatives aux étudiants. En terme d'analyses nous nous intéressons à observer les effectifs des étudiants par rapport spécialité, cycle et année d'inscription.Nous présentons dans cette section le schéma des données sources ainsi que le schéma des données cibles en déployant le schéma dimensionnel en étoile pour la vocation de faciliter la réaliser de notre système.

3.1 Schéma des données sources :

Nous pouvons présenter les données sources lesquelles nous utilisons dans notre système comme une seule table Etudiant. Elle contient l'attribut Matricule comme une clé primaire, plus cinq autres attributs: Date_inscription, cycle, spécialité, bourse et sport. Voir la figure 22 :

Etudiant
#Matricule
Date_inscription
Cycle
Spécialité
Bourse
Sport

Figure 22: Schéma des données sources du système« Table Etudiant ».

3.2 Schéma des données cibles :

Pour concevoir le modèle de données cibles de notre systèmes nous utilisons le même schéma en étoile présenté dans le chapitre III section 5.2 (figure 16). Il s'agit donc d'un schéma représentant l'analyse de l'effectif des étudiants par rapport spécialité, cycle et année.

3.3 Spécification des besoins :

La spécification des besoins doit décrire sans ambiguïté le logiciel à développer en fournissant un ensemble de documents et de modèles.
Les cas d'utilisation jouent un rôle fondamental dans le cycle de vie d'un projet de

développement logiciel, ils permettent d'identifier les possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire toutes les fonctionnalités que doit fournir le système, il peut comprendre de plus des notes servant à décrire des contraintes ou faire des commentaires.

Acteur du système :

- **L'utilisateur** : dont ses principaux tâches consistent à créer des processus ETL, les paramétrer et les exécuter.

3.3.1 Diagramme global du système:

Le système permet aux différents utilisateurs de gérer les zones de stockage de données, paramétrer le cluster ainsi que la gestion de processus d'intégration de données , dans le but de rafraichir l'entrepôt de données périodiquement.

La figure 23 représente les fonctionnalités globales qui assurent le fonctionnement du système. Les deux cas d'utilisation «Gérer les zones de stockage de données » et « Paramétrer cluster » sera développé ultérieurement afin de le décrire plus en détail.

3.3.1.1 Diagramme de cas d'utilisation global :

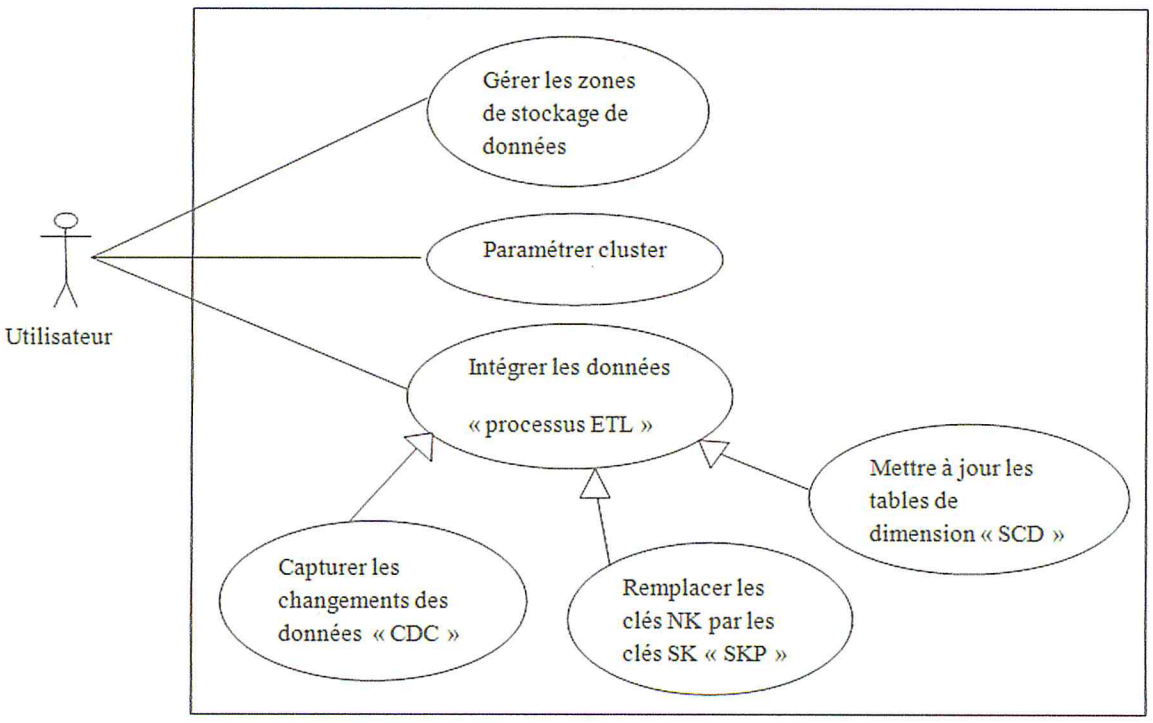

Figure 23 : Diagramme de cas d'utilisation global du système.

Description du diagramme :

- **Gérer les zones de stockage de données :** La gestion des zones de stockage de données consiste à spécifier les emplacements des données sources , de la zone de préparation ainsi que de l'entrepôt de données utilisées tout au long de l'exécution de processus ETL.
- **Paramétrer cluster :** gérer le cluster en rajoutant /supprimant des nœuds ou spécifiant leurs caractéristiques techniques.
- **Intégrer les données « processus ETL » :** En ce qui concerne le processus d'intégration ETL, nous nous sommes focalisés particulièrement sur les fonctionnalités CDC, SKP et SCD en vue de leur distribution sur un cluster d'ordinateur.
- **Capturer les changements de données « CDC »** : Cette fonctionnalité consiste à comparer différents instantanés des données sources afin d'identifier les éventuels changements (insertions, modifications, suppressions). Ce sont ces données affectées par des changements qui nous intéresse pour le rafraichissement de l'entrepôt de données.
- **Remplacer les clés NK par les clés SK « SKP » :** Afin de maintenir l'intégrité référentielle entre les tables de dimensions et les tables de faits, cette fonctionnalité assure le remplacement les clés NK d'une table du fait par les clés SK correspondantes des dimensions associées.
- **Mettre à jour les tables de dimension « SCD » :** Cette fonctionnalité est chargée d'appliquer des stratégies de mise à jour des membres d'une dimension selon les besoins d'archivage et de versionnement des valeurs des attributs de la dimension.

3.3.1.2 Diagramme «Gestion de zones de stockage de données» :

Diagramme :

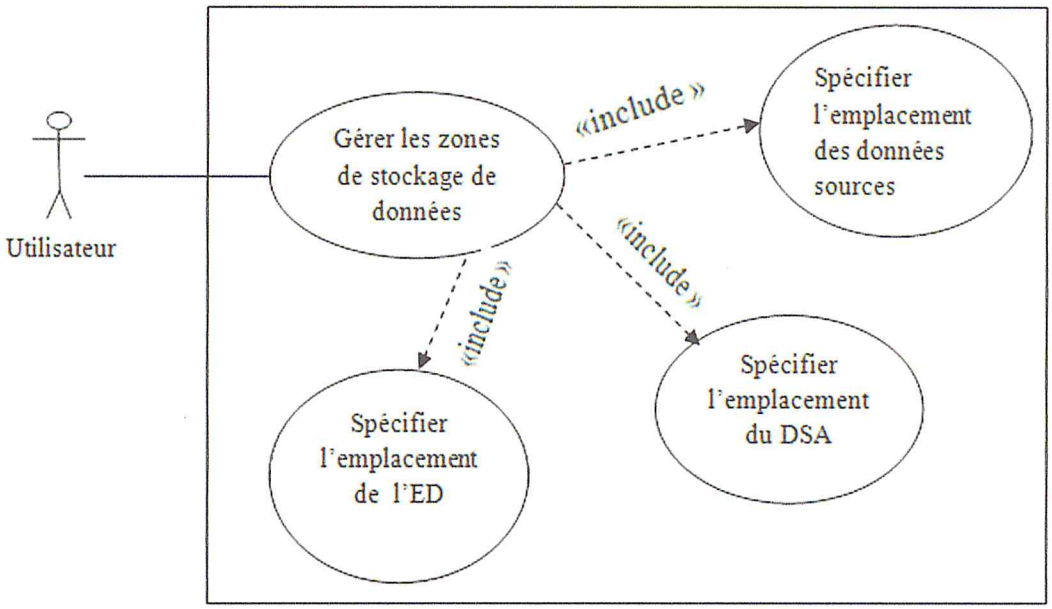

Figure 24 : Diagramme de cas d'utilisation « Gestion de zones de stockage de données».

Description du diagramme :

- **Spécifier l'emplacement des données sources :** Spécifier l'emplacement physique des données sources à partir duquel la fonctionnalité CDC pourra assurerl'extraction.
- **Spécifier l'emplacement du DSA :** Afin de préparer les données extraites à partir des sources paramétrées dans un processus ETL, il est nécessaire de spécifier la zone où seront réalisées les opérations de transformation des données.
- **Spécifier l'emplacement de l'ED :** De même, il est nécessaire de spécifier l'emplacement cible (entrepôt de données) dans lequel seront chargées les données nettoyées et intégrées.

3.3.1.3 Diagramme «Paramétrage de cluster » :

Diagramme :

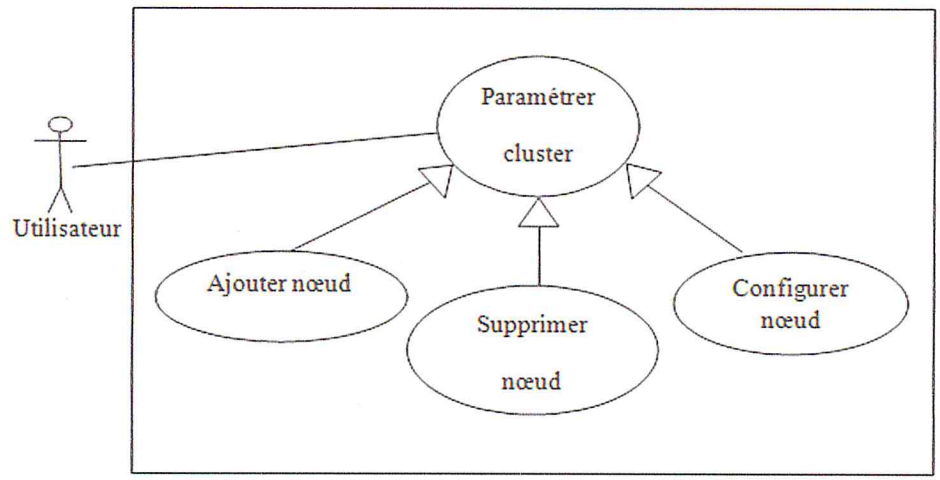

Figure 25: Diagramme de cas d'utilisation « Paramétrage de cluster ».

Description du diagramme :

- **Ajouter nœud :** l'ajout d'un nouveau nœud au cluster en spécifiant son adresse IP au fichier de configuration.
- **Supprimer nœud :** Enlever un nœud de cluster en retirant son adresse IP du fichier de configuration.
- **Configurer nœud :** spécifier les caractéristiques techniques d'un nœud à savoir sa taille de mémoire utilisée, nombre de noyaux utilisés.

4.Architecture globale du système

Notre système est organisé en cinq modules : (E)xtracting, (P)artitioning, (T)ransforming, (R)educing et (L)oading (Figure 26).

- ✓ **(E)xtracting :**ce module est consacré pour la définition des sources de données à éxtraire pour alimenter l'ED.
- ✓ **(P)artitioning :** Le paradigme *Map/Reduce*permet de partitionner de gros volumes de données dont chaque partition sera soumise à une instance du processus ETL.

- ✓ **(T)ransforming :**une fois les données extraites, elles peuvent être transformées. Les opérations les plus courantes à ce niveau incluent des opérations de filtrage de données, de dérivation de valeurs, de transformation de formats de données, de génération automatique de numéros de séquence etc. Pour traiter ces différentes catégories de transformations, les Mappers consiste à énumérer les transformations à opérer sur les fragments de données. Ces transformations sont exécutées, en parallèle, par chaque mapper sur sa partition.
- ✓ **(R)educing :**

  Les reducers fusionnent les résultats intermédiaires remontées à partir des mappers et les combine afin de fournir les résultats finaux qui sont transférés par la suite à l'entrepôt de données.

  Entre la phase Map et la phase reduce, le partitionnement montre comment les résultats partiels des mappers sont soumis aux reducers, c'est la phase de shuffle.
- ✓ **(L)oading :** après avoir fusionner les résultats fournis par les reducers on les charge dans l'entrepôt de données.

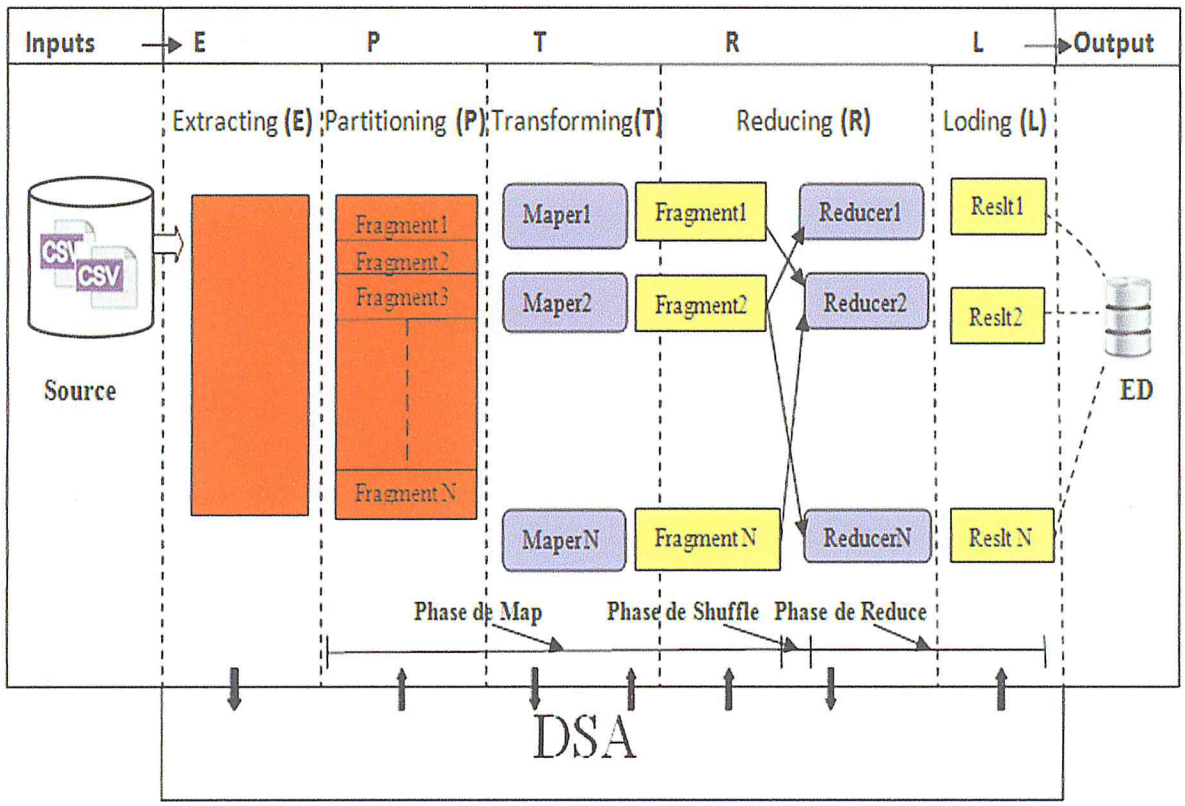

Figure 26 : Architecture globale du système.

5. Diagrammes de séquence

Le diagramme de séquence montre les interactions entre les objets, agencées en séquence dans le temps. Il montre en particulier les objets participants à l'interaction par leurs lignes de vie et les messages qu'ils s'échangent de façon ordonnancée dans le temps [32]. Dans ce qui suit, nous présentons les principaux diagrammes de séquence :

5.1 Diagramme de séquence Capturer les changements de données « CDC » :

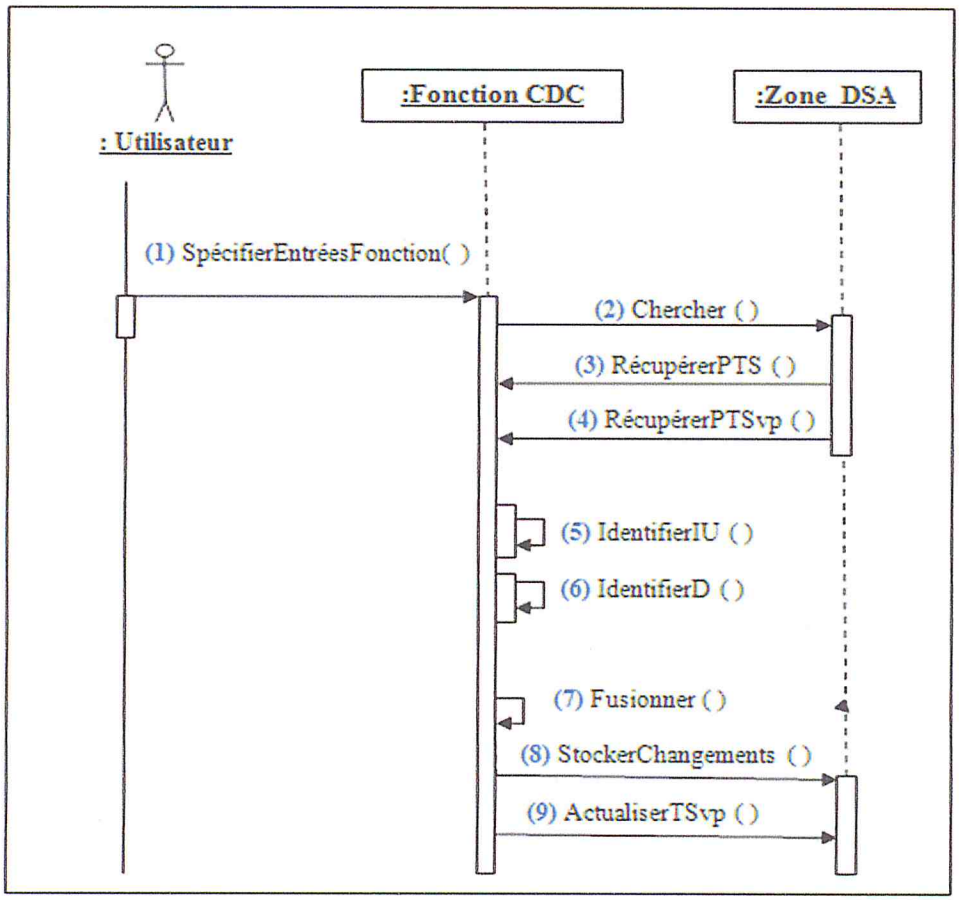

Figure 27: Diagramme de séquence Capturer les changements de données « CDC ».

Description du scénario :

Le diagramme décrit les interactions entre les objets dans le cas d'utilisation « Capturer les changements de données « CDC » ».

(1)L'utilisateur spécifier les entrées de la fonction CDC y compris l'emplacement de la table source PTS,l'emplacement de sa version précédent PTSvp et le chemin où les résultats seront stockées.

(2)L'objet fonction CDC envoie une requête au Zone DSA pour récupérer les tables PTS et PTSvp.

(3)L'objet Zone DSA envoie la table PTS à l'objet Fonction CDC.

(4)L'objet Zone DSA envoie la table PTSvp à l'objet Fonction CDC.

(5)L'objet Fonction CDC détermine les enregistrements de la table PTS qui ont connu des changements de type INSERT ou UPDATE.

(6)L'objet Fonction CDC détermine les enregistrements de la table PTSvp qui ont été supprimés.

(7) Après avoir identifiertous les changements les résultatsde toutes les partitions seront fusionnés (Reduce).

(8) Après avoir fusionner les résultats de changement, l'objet Fonction CDC les stocke dans la Zone DSA.

(9) La table TSvp sera mise à jour à la fin de traitement.

5.2 Diagramme de séquence Remplacer les clés NK par les clés SK « SKP » :

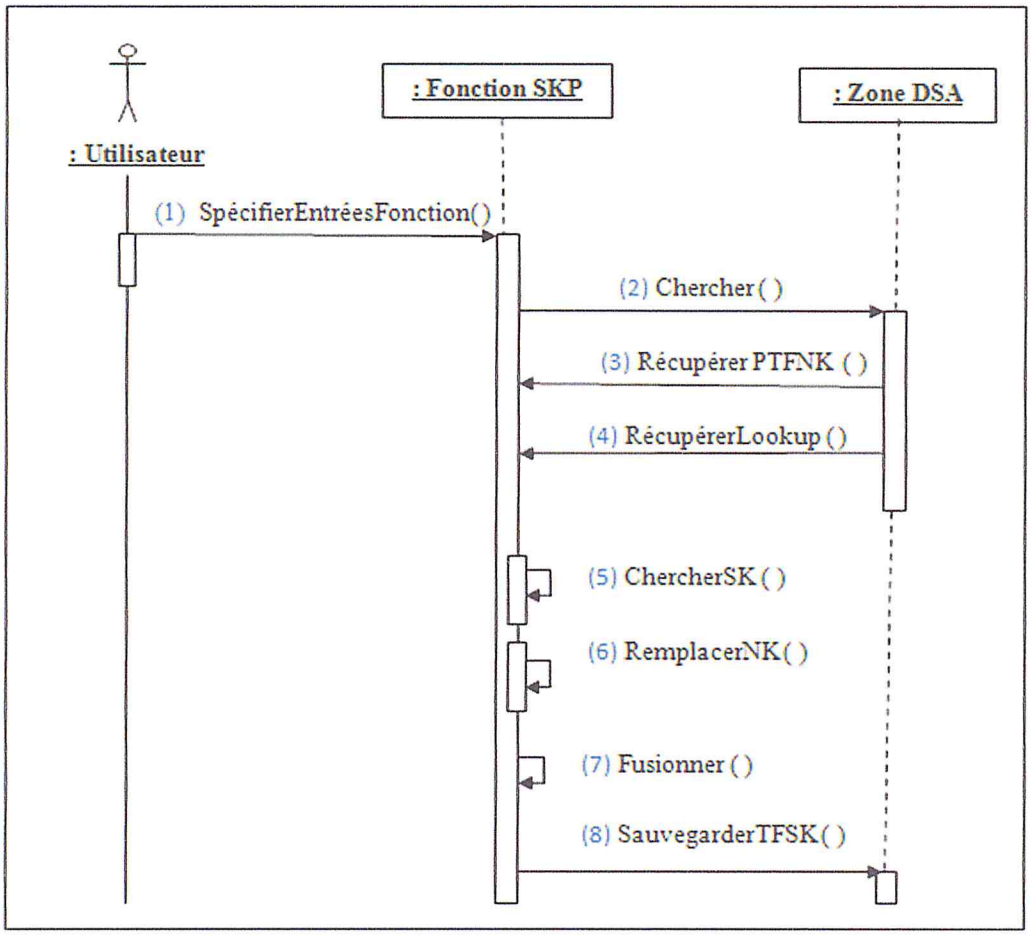

Figure 28 : Diagramme de séquence Remplacer les clés NK par les clés SK « SKP ».

Description du scénario :

Le diagramme décrit les interactions entre les objets dans le cas d'utilisation Remplacer les clés NK par les clés SK « SKP »».

(1)L'utilisateur spécifier les entrées de la fonction SKP y compris l'emplacement de la table de fait TF, de la table Lookup et de la Zone DSA. Ainsi, le nombre des tables de dimension associées.

(2)L'objet Fonction SKP envoie une requête à la Zone DSA pour récupérer les tables PTFNKet Lookup.

(3)L'objet Zone DSA envoie la table PTFNKà l'objet Fonction SKP.

(4)L'objet Zone DSA envoie les tables Lookup à l'objet Fonction SKP.

(5) L'objet Fonction SKP cherche les clés SK correspondantes dans les tables Lookup.

(6) L'objet Fonction SKP remplace les clés NKs par les clés SKs dans la table PTFNK.

(7) Après avoir remplacer toutes les clés NKs par les clés SKs correspondantes, les résultats de toutes les partitions sont fusionnés(Reduce).

(8)La table TF est stockée avec les clés SK dans la Zone DSA.

5.3 Diagramme de séquence Mettre à jour les tables de dimension « SCD » :

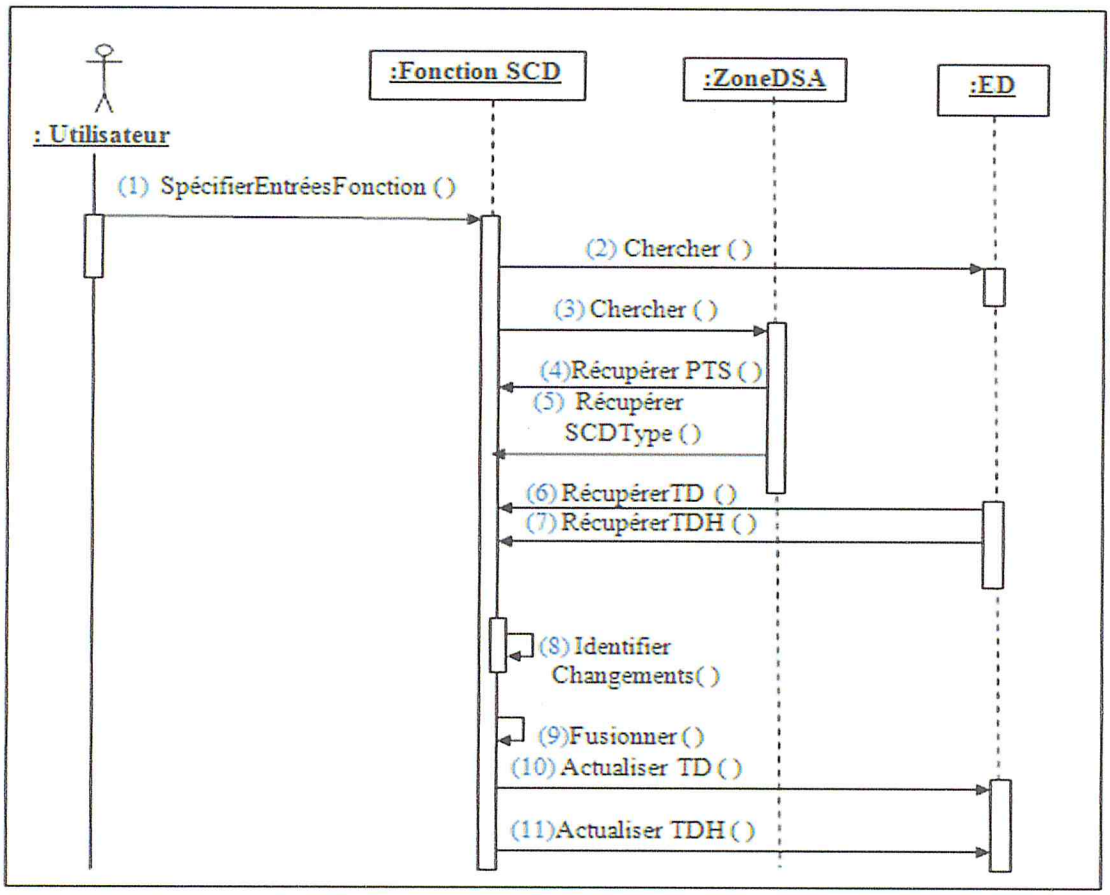

Figure 29 : Diagramme de séquence Mettre à jour les tables de dimension « SCD ».

Description du scénario :

Le diagramme décrit les interactions entre les objets dans le cas d'utilisation « Mettre à jour les tables de dimension « SCD » ».

(1)L'utilisateur spécifier les entrées de la fonction SCD y compris l'emplacement des tables de dimension TD et TDH , de la table sourcePTS et de la table TypeSCD qui contient les types des attributs .

(2)L'objet Fonction SCD envoie une requête à l'entrepôt de données pour récupérer la table TD et la table TDH.

(3)L'objet Fonction SCD envoie une requête à la Zone DSA pour récupérer la table PTS et la table TypeSCD.

(4)L'objet Zone DSA envoie la table PTS à l'objet Fonction SCD.

(5)L'objet Zone DSA envoie la table TypeSCDà l'objet Fonction SCD.

(6)L'objetED envoie la table TD à l'objet Fonction SCD.

(7)L'objet ED envoie la table TDH à l'objet Fonction SCD.

(8)La détection des tuples qui ont subi des modification et les mettre à jour en respectant le type des attributs.

(9)La fusion les résultats des traitements de chaque partitions (Reduce).

(10)La table TD sera mise à jour à la fin de traitement.

(11)La table TDH sera mise à jour à la fin de traitement.

6.Conclusion

Dans ce chapitre, nous avons déployé le diagramme de cas d'utilisation d'UML ainsi que les schémas multidimensionnels pour modéliser notre système ce qui nous facilite la réalisation de son implémentation dans le prochain chapitre.

CHAPITRE V

IMPLEMENTATION ET EXPERIMENTATION

1. Introduction

Une fois la modélisation de notre système a été conçue dans le chapitre précédent nous arrivons à son implémentation, pour le faire nous avons utilisé un ensemble d'outils qui seront présentés par la suite dans ce chapitre. Nous allons ainsi présenter le résultat des expérimentations que nous avons pu faire afin de voir si nous avons aboutit nos objectifs.

2. Implémentation

2.1 Outils utilisés :

2.1.1 Langage Java :

Le langage Java est un langage de programmation orienté objet, développé par *Sun Microsystems*,Outre son orientation objet le langage Java a l'avantage d'être modulaire le faite qu'on peut écrire des portions de code génériques c.à.d. utilisables par plusieurs applications, rigoureux puisque la plupart des erreurs se produisent à la compilation et non à l'exécution et portable dont un même programme compilé peut s'exécuter sur différents environnements[33].

Sa particularité principale est que les logiciels écrits avec ce dernier sont très facilement portables sur plusieurs systèmes d'exploitation tels que Unix, Microsoft Windows, Mac OS ou Linux avec peu ou pas de modifications, c'est pour cette raison que nous avons l'utilisé.

2.1.2 Eclipse IDE :

C'est un environnement de développement intégré libre (le terme Eclipse désigne également le projet correspondant, lancé par IBM) extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation [34]. Nous l'avons choisi parce qu'il convient aux nos besoin tout simplement, de même il est tellement facile à utiliser.

2.1.3 Apache Maven :

Apache Maven est l'outil open-source qui apporte aux développeurs une gestion et une automatisation des principales tâches nécessaires à la mise en œuvre et au déploiement d'un projet java [35]. La configuration d'un projet Maven se fait à l'aide d'un fichier pom.xml situé à la racine du projet dans lequel on peut spécifier les dépendances

(bibliothèques) nécessaires pour le projet.

La figure 30 représente le fichier pom.xml de notre projet dans lequel nous spécifions les dépendances que nous avons besoin qui inclurent essentiellement le framework Apache spark qui est responsable de faire le calcul parallélisé(plus de détail dans ce qui suit) et la bibliothèque openCSV qui gère les opérations sur les fichiers csv.

Après avoir terminer le codage nous avons créé le fichier exécutable myApp.jar de notre application à l'aide du Maven. Cet exécutable sera utilisé pour faire nos expérimentations.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="ht
  <modelVersion>4.0.0</modelVersion>
  <groupId>myapp.com.master</groupId>
  <artifactId>myapp</artifactId>
  <version>0.0.1-SNAPSHOT</version>
   <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                 <artifactId>maven-compiler-plugin</artifactId>
                <version>3.1</version>
                <configuration>
                    <source>1.7</source>
                    <target>1.7</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
 <dependencies>
<dependency>
 <groupId>org.apache.spark</groupId>
 <artifactId>spark-core_2.11</artifactId>
 <version>2.0.0</version>
 <scope>provided</scope>
 </dependency>
<dependency>
    <groupId>net.sf.opencsv</groupId>
    <artifactId>opencsv</artifactId>
    <version>2.3</version>
 </dependency>

 </dependencies>
<properties>
 <java.version>1.7.0_71</java.version>
 </properties>


 </project>
```

Figure 30:Extrait du fichier pom.xml de notre projet.

3. Expérimentations

Pour évaluer notre travail, nous avons alloué 12 nœuds du cluster IBNBADIS sur lesquels nous avons fait les configurations nécessaires qui inclurent essentiellement l'installation de framework Apache Spark chargé de la distribution.

3.1 Données de test :

Pour le test nous avons utilisé un programme qui génère différentes tailles de données relative aux renseignements des étudiants ayant le même schéma des données présenté dans la figure 23 (chapitre IV, section 2.1). Les expérimentations ont été réalisées sur des jeux de données allant de 50 Mo à 5000 Mo.

3.2 Déploiement de l'environnement de test :

3.2.1 Cluster IBNBADIS :

IBNBADIS est une plateforme de calcul haute performance (*HPC: High Performance Computing*) fournie par le Centre de Recherche sur l'Information Scientifique et Technique (*CERIST*), elle est composé d'un nœud d'administration *ibnbadis0*, un nœud de visualisation *ibnbadis10* et 32 nœuds de calcul (*ibnbadis11-ibnbadis42*) dotés chacun de deux processeurs Intel(R) Xeon(R) CPU E5-2650 2.00GHz et de 30 Go de la RAM. Chaque processeur est composé de 8 cœurs, ce qui fait 512 cœurs au total. La puissance théorique du cluster est d'environs 8TFLOPS, pour plus de détail voir le lien [36].

3.2.2 Environnement logiciel :

3.2.2.1 Système BullX :

Le système d'exploitation du cluster *IBNBADIS* est *Bullx Linux.* Ce dernierest basé sur *RedHat Entreprise Linux* qui est une distribution *Linux* fournie par *Bull* et dédiée aux applications *HPC.*

3.2.2.2 Apache Spark :

Apache Spark est un framework de traitements de gros volumes de données de manière distribuée et rapide. Il a été développé par *AMPLab de l'Université UC Berkeley, Californie* en 2009 et passé open source sous forme de projet Apache en 2010.

Apache Spark se caractérise par :

• **La vitesse** : Par rapport à MapReduce, il est 100 fois plus rapide sur les traitements en mémoire, et 10 fois plus rapide sur les traitements sur disque.

• **La facilité d'utilisation** : Spark vous permet d'écrire rapidement des applications en Java, Scala, ou Python. Il offre ainsi un shell pour requêter les données de façon interactive.

• **L'analytics sophistiqués** : en plus des opérations de Map et Reduce, Spark supporte les requêtes SQL, les flux de données (*Streaming*) et propose des fonctionnalités d'apprentissage automatique et de traitements orientés graphe.

L'idée principale du projet Spark était de concevoir un outil pour soutenir le traitement en mémoire, afin que les développeurs puissent travailler sur des algorithmes itératifs sans être obligés à chaque fois de stocker les résultats d'une itération avant de passer à l'itération suivante. Cela permet d'améliorer la vitesse d'exécution des applications *BigData* qui réutilisent les données de manière récurrente.

❖ **ResilientDistributedDatasets (RDD)** : La base de Spark

Un RDD est le concept central de Spark, il s'agit d'une collection d'objets immuables partitionnés sur l'ensemble des machines d'un cluster en partitions logiques, qui peuvent être calculées sur les différents nœuds du cluster. Il est aussi tolérant aux pannes car un RDD sait comment recréer et recalculer son ensemble de données.
Le RDD est simple à créer et peut être obtenu à partir de multiples sources : une collection dans le programme (Liste..) transformée en RDD, un fichier local ou distribué *(HDFS: HadoopDistributed File System),* une base de données ou un autre RDD auquel on aura appliqué une transformation.

Les RDD supportent deux types d'opérations :

- **Les transformations** (map, filter, groupBy….) : ce sont des descriptions du travail qu'on veut faire sur les sources de données, elles retournent un autre RDD.
- **Les actions** (count, collect, et save…) : ce type d'opération permet de fournir un livrable, soit le résultat final soit un autre RDD.

3.2.3 Configuration du cluster Spark en mode Standalone :

Spark utilise l'architecture Master/Slave. Comme le montre la figure 31, le **driver** (*machine maitre*) est le processus qui exécute la méthode principale de l'application. Il convertit le programme en tâches et les affecter aux exécuteurs (*Workers*: *machines esclaves)* viaSparkContext.

Spark offre le gestionnaire de cluster **SparkStandalone** chargé de piloter les nœud esclaves, leur distribuer les tâches équitablement, et arbitrer la quantité de CPU et de mémoire qui sera allouée à chacun des traitements.

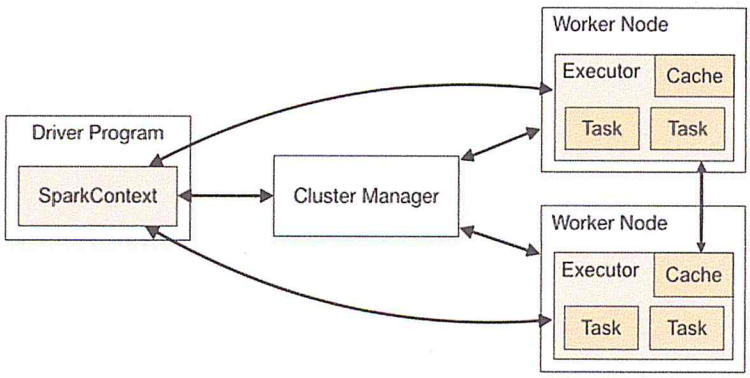

Figure 31: Processus Spark [37].

❖ **Installation du Spark :**

Afin d'installer Spark sur un cluster en mode Standalone il suffit de faire le suivant : dans un premier temps, il faut télécharger le package de la version la plus récente de Spark (dans notre cas est la version 2.0.0), le décompresser et le placer sur chaque nœud. Et pour faire connecter le Master avec les Workers, il faut spécifier les adresses IP de ces dernières

dans un fichier Slaves qui se situe dans la racine du répertoire de Spark installé (voir l'annexe).

Lancement du cluster :

Après l'installation de Spark nous pouvons démarrer le serveur master ainsi les workers via le shell en utilisant la commande : **./sbin/start-all.sh**

Nous pouvons accéder au informations sur l'état du master et de ses workers, des ressources utilsés pour chaque worker (mémoire et cœurs) et les situations des applications via l'explorateur web, en tapant l'adresse URL par défaut **"http: // localhost: 8080"**, ce qui donne la fenêtre suivante (voir figure 32):

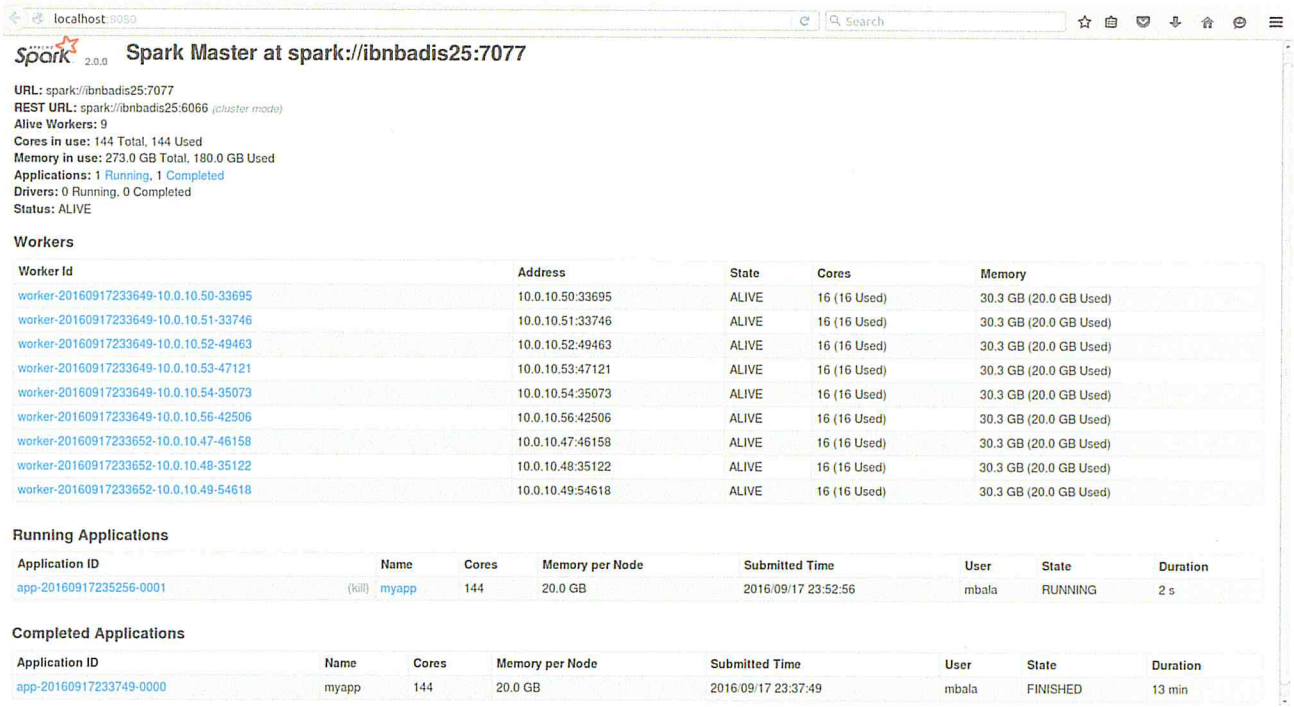

Figure 32: Master et workers.

Une fois notre cluster prêt il ne reste plus qu'à lancer le traitement en utilisant la commande **spark-submit** pour soumettre le traitement au cluster.

```
./bin/spark-submit\
 --class myapp.Main\                    //classe Main de notre programme
 --master spark://ibnbadis25:7077 \  // se connecter au master (machine maitre)
 --driver-memory 20G \          // Spécifier la taille de mémoire fournie au driver
 --executor-memory 20G \        // Spécifier la taille de mémoire fournie au exécuteur
```

```
/home/mbala/myapp.jar \          // Spécifier l'exécutable jar de notre application
// Ici nous ajoutons les entrées de l'application
```

Nous pouvons consulter l'application en cours d'exécution en tapant l'adresse URL par défaut **"http: // localhost: 4040"**, nous pouvons voir tous les détails des exécuteurs, la situation des tâches et l'environnement de l'exécution (voir figure 33).

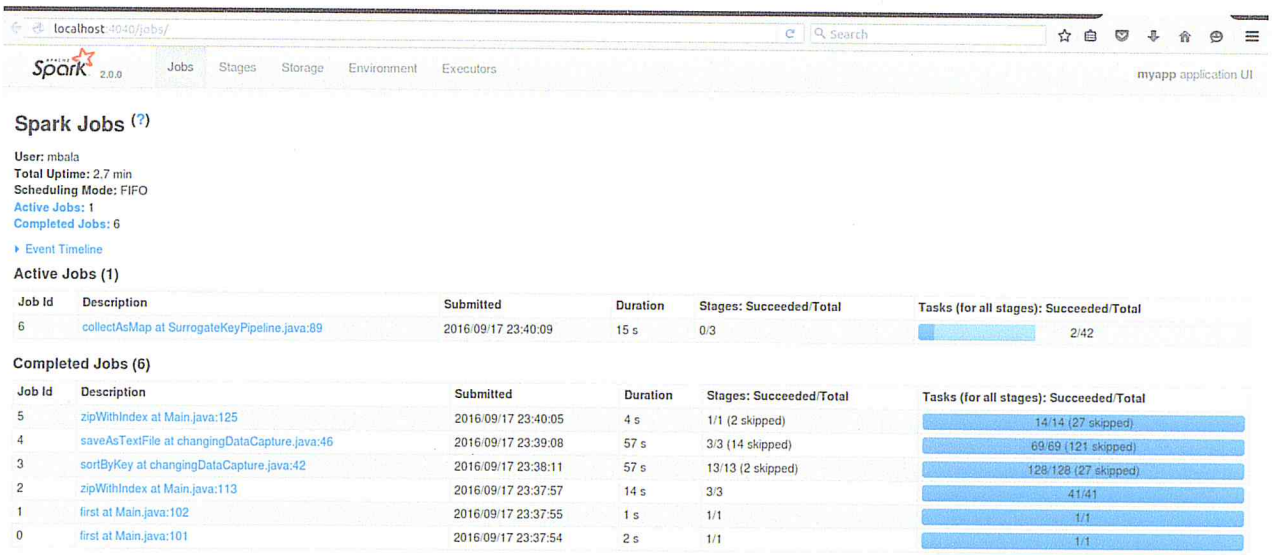

Figure 33: Détail d'exécution de l'application.

3.3 Analyse expérimentale :

3.3.1 Test 1:

Dans un premier test nous avons focalisé sur l'impact de la taille des données en entrée sur le temps d'exécution, pour cela nous avons générer différentes tailles du fichier Etudiant. Le tableau 5montre le temps d'exécution retenu :

Taille de données (Mo)	50	100	300	500	700	900	5000
Temps d'exécution (min)	2,6	3,5	9,6	12	17	26	37

Tableau 5: Temps d'exécution (min) par rapport à la taille des données.

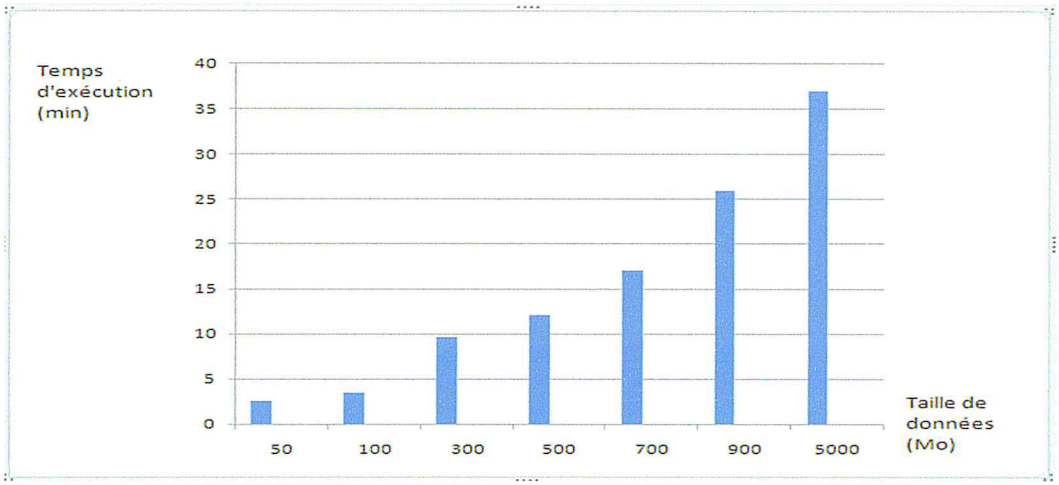

Figure 34: Temps d'exécution (min) par rapport à la taille des données.

D'après la figure 34, nous remarquons que le temps d'exécution s'augmente lorsque nous augmentons la tailledes fichiers. Ce qui implique que la taille des données a un impact direct sur le temps d'exécution d'application, et plus particulièrement sur la vitesse du processus ETL. Néanmoins, nous remarquons que la différence entre le temps d'exécution de 900Mo et 500Mo est 14 min, alors que la différence entre le temps d'exécution de 900Mo et 5000Mo est 11 min, ceci montre la puissance de notre méthodologie lorsque nous manipulerons des fichiers plus volumineux.

3.3.2 Test 2 :

Dans ce test nous avons focalisé sur l'impact de la taille du cluster (nombre des nœuds) sur le temps d'exécution du processus, pour cela nous avons utilisé un fichier ayant une taille de 100 Mo et nous avons varié le nombre des nœuds workers de 2 à10 plus le nœud maitre (master). Le tableau 6 montre le temps d'exécution retenu :

Nombre des nœuds	2	4	6	10
Temps d'exécution (min)	9,8	5,3	3,7	3,5

Tableau 6: Temps d'exécution (min) par rapport à la taille du cluster.

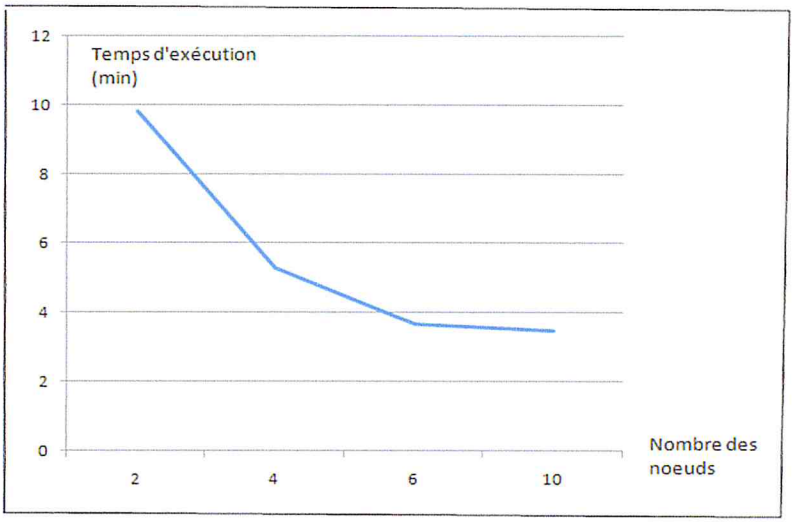

Figure 35: Temps d'exécution (min) par rapport à la taille du cluster.

La figure 35 représente le temps d'exécution de l'application sur le cluster ,nous observons qu'en ajoutant des nœuds au cluster la vitesse du processus s'augmente de manière significative. Néanmoins, nous remarquons que le temps d'exécution se diminue entre 6 et 10 nœuds de façon moins significative. Nous pouvons constater que le nombre des nœuds joue un rôle important lorsqu'il s'agit d'amélioration des performances, mais ils existent d'autres facteurs tels la taille de la mémoire, nombre de taches en parallèle, etc.

4. Conclusion

Dans ce dernier chapitre, nous avons présenté les outils les plus importants que nous avons utilisés pour l'implémentation de notre application. Puis nous avons exposé l'environnement dans lequel nous avons réalisé nos expérimentations afin d'évaluer notre approche qui fait la parallélisassion et la distribution d'ETL (au niveau fonctionnalité). Nous avons constaté d'après leur résultatque notre approche de distribution des données est efficace pour les Big Data, c'est pourquoi le temps d'exécution se diminuait lorsque nous avons ajouté des nœuds (augmentation de nombre des processus parallèles). Mais il nous reste à faire des tests comparatifs avec les approches qui font la parallélisassion au niveau processus seulement pour pouvoir valider.

CONCLUSION GENERALE

Conclusion générale

Les données massives (Big Data) sont présentes aujourd'hui dans la plus part des systèmes d'information. Ces données doivent être passées par une phase d'intégration chargée d'extraire les données à partir de sources hétérogènes, les transformer et enfin les charger dans l'ED. Cependant, leur traitement pose une réelle problématique due aux limites en termes de capacité de traitement et de stockage.

L'objectif de ce mémoire se situe dans la phase d'intégration des données massives, celle-ci est basée sur un processus d'extraction, de transformation et de chargement (ETL). Particulièrement, dans un environnement BigData , le processus ETL est distribué en plusieurs instances qui s'exécutent de façon parallèle sur un cluster d'ordinateurs pour pouvoir traiter des volumes inhabituels de données. La plus part des travaux ont opté pour des solutions fondées sur le paradigme MapReduce pour l'automatisation et la parallélisation du processus. Dans ce contexte, il nous a été demandé de définir le processus ETL en termes de ses fonctionnalités de base, de proposer une architecture et des algorithmes dans un environnement parallèle et distribué et de les implémenter selon la technique « in-memory » sous le framework de parallélisation «Apache Spark». Enfin, le travail devra être évalué grâce à une expérimentation sur un cluster d'ordinateurs doté du Spark.

Comme ce travail se situe dans un contexte décisionnel, nous avons tout d'abord exposé les fondamentaux sur les systèmes décisionnels : les entrepôts de données, les environnements OLAP et OLTP, les différents modèles de modélisation à un niveau conceptuel et logique et plus particulièrement, le processus ETL et ses différentes phases :Extraction (E), Transformation (T) et Chargement (L) ainsi en termes de ses fonctionnalités de base. Dans une deuxième étape nous avons choisi d'étudier les fonctions d'ETL les plus courantes à savoir Changing Data Capture (CDC), SlowlyChanging Dimension (SCD) et Surrogate Key Pipeline (SKP). Puis nous avons proposé des architectures et des algorithmes de ces dernières dans un environnement parallèle et distribué selon le paradigme MapReduce. Enfin, dans une dernière étape, nous avons conçu et mis en œuvre notre système en implémentant les trois fonctions en java selon la technique « in-memory » du Spark, puis nous avons créé le fichier jar de notre application afin de l'utiliser dans nos expérimentations sur le cluster IBNBADIS. Les expérimentations ont été fait afin de voir l'impact de la taille des données ainsi la taille de

cluster sur le temps d'exécution, nous constatons alors que notre approche est efficace pour la manipulation des Big Data lorsque le niveau de parallélisme est important mais il reste à faire des tests comparatifs avec d'autres approches pour pouvoir la valider.

Perspectives

Pour finir cette formidable expérience, nous pouvons dire que ce travail nous a permis d'acquérir une très bonne expérience professionnelle et d'évoluer dans un domaine intéressant et actuel. Comme un projet dans le cadre de PFE n'est jamais complètement terminé, nous pouvons citer les perspectives suivants :

- Proposer des architectures d'autres fonctions d'ETL importantes afin de prendre en charge des processus d'ETL consistants et complexes.

- Appliquer des expérimentations sur des données de taille plus grande (Go, To..) ainsi faire des expérimentations comparatives pour valider notre approche.

- Prendre en charge d'autres formats de données sources telles que les bases de données relationnelles, documents XML, données NoSQL (Not Only SQL), etc.

- Enfin, doter l'application par des interfaces graphiques pour faciliter à un utilisateur non informaticien de l'utiliser.

Annexe

Initiation à Apache Spark

1.Introduction

Spark est un framework qui fait suite à Hadoop et son écosystème gravitant autour du monde du Big Data. Il a été créé initialement par **Matei Zaharia**. Spark est connu pour être 100 fois plus rapide que Hadoop et a prouvé sa scalabilité jusqu'à 8000 nœuds. Il est l'un des projets Apache les plus actifs avec pas moins de 450 contributeurs. Spark est très utilisé dans le domaine du *machine learning* appliqué au *Big Data*.

L'objectif de Spark est de permettre aux développeurs de créer des applications Big Data. Plus en détail, Spark a pour but de faciliter l'écriture et d'améliorer la vitesse d'exécution d'applications Big Data qui ré-utilise des données de manière récurrente (des algorithmes itératifs ou interactifs par exemple). Spark implémente le framework de programmation MapReduce popularisé par Hadoop et donc la programmation fonctionnelle. Il est écrit en Scala et s'exécute sur la machine virtuelle Java (JVM). Grâce aux différentes APIs (Scala, Java et Python), Spark permet d'écrire des programmes puissants et rapides en très peu de lignes de code.

2. L'écosystème de Spark

À côté des APIs principales de Spark, l'écosystème contient des librairies additionnelles qui permettent de travailler dans le domaine des analyses big data et du machine learning. Parmi ces librairies, on trouve :

- **Spark SQL** (Java, Scala etPython) : API permettant de requêter les RDDs en SQL.
- **Spark Streaming** (Java et Scala) : API permettant d'écrire des applications streaming.
- **MLlib** (Java, Scala et Python) : API reprenant certains des algorithmes bien connus du machine learning (kmeans, régression linéaire, analyse en composantes principales, …).
- **GraphX** (Scala) : API permettant de manipuler et de faire des opérations sur les RDDs à la manière d'un graph.

- **SparkR** : est un package dédié au langage de statistique R, et qui permet aux utilisateurs d'utiliser les fonctionnalités de Spark à partir d'une interface R. il traite les RDD comme des listes distribuées.
- **BlinkDB** : est un moteur de recherche «approximative » qui exécute des requêtes SQL sur des données de grand volume. Pourquoi approximative ? L'idée c'est que dans

certaines situations les utilisateurs privilégient le temps de réponse ou des résultats avec un taux d'erreur toléré sur la précision et la justesse des résultats. Donc si vous voulez des réponses rapides *raisonnablement*
précises au lieu de bonnes réponses lentes, BlinkDB est le bien placé.

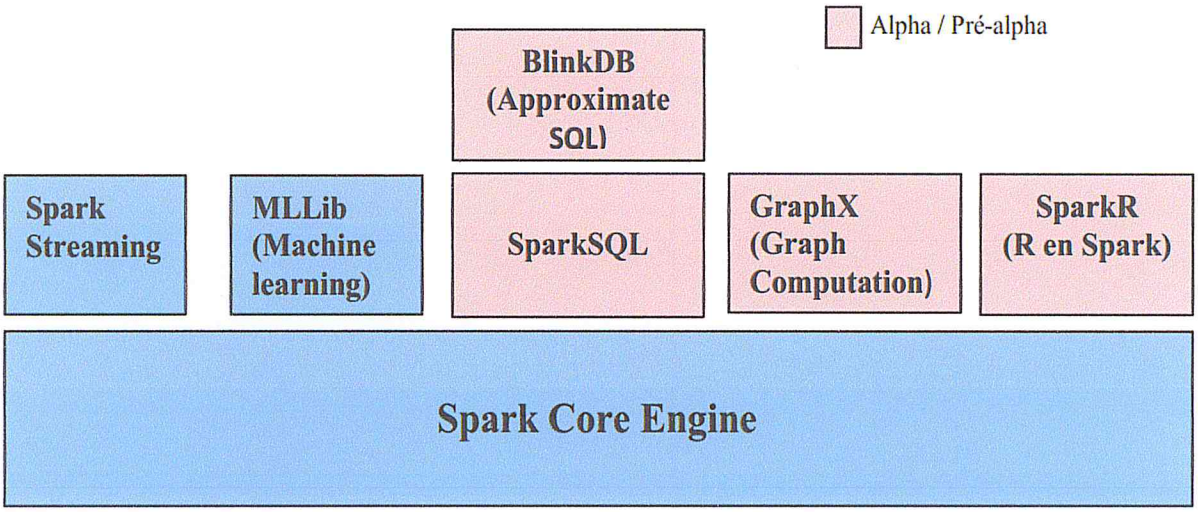

Figure 36 : Ecosystème Spark.

3. Installation de Spark

Puisque Spark est écrit en Scala et s'exécute en JVM, ces derniers doivent être installés sur les machines de cluster. Pour son installation nous devons télécharger son archive à partir du site officiel. Puis l'extraire en utilisant les commandes suivantes :

wget http://d3kbcqa49mib13.cloudfront.net/spark-2.0.0.tgz **//Telecharger**
tar -zxf spark-1.0.0.tgz **// déziper**

Pour vérifier notre installation, nous pouvons lancer l'application Spark Shell depuis le répertoire spark :

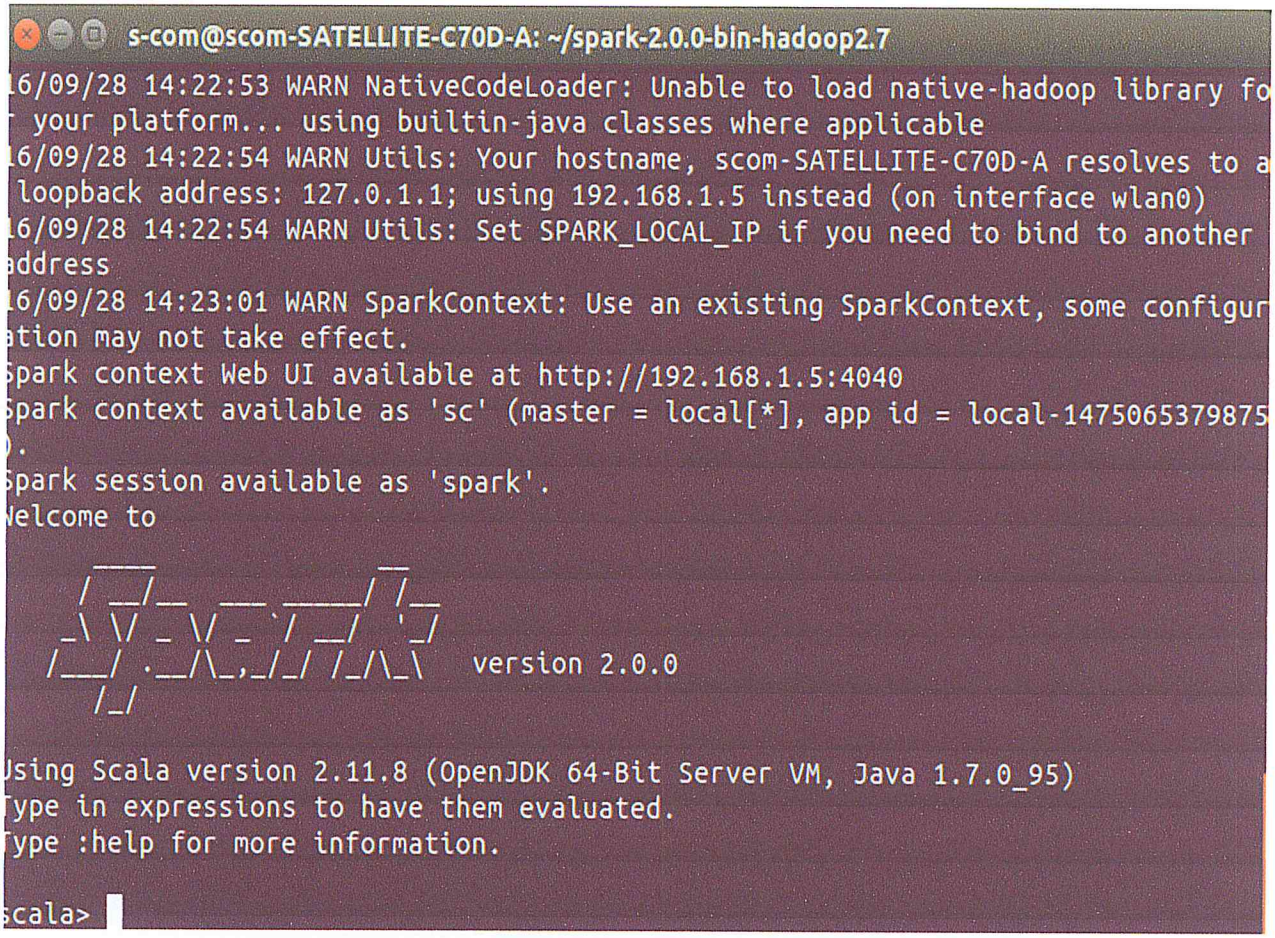

Figure 37 : Scala shell de Spark

4. Exemples d'utilisation des RDDs sur SPARK:

4.1 Exemple1 : Nombre de répétition d'un mot dans un texte.

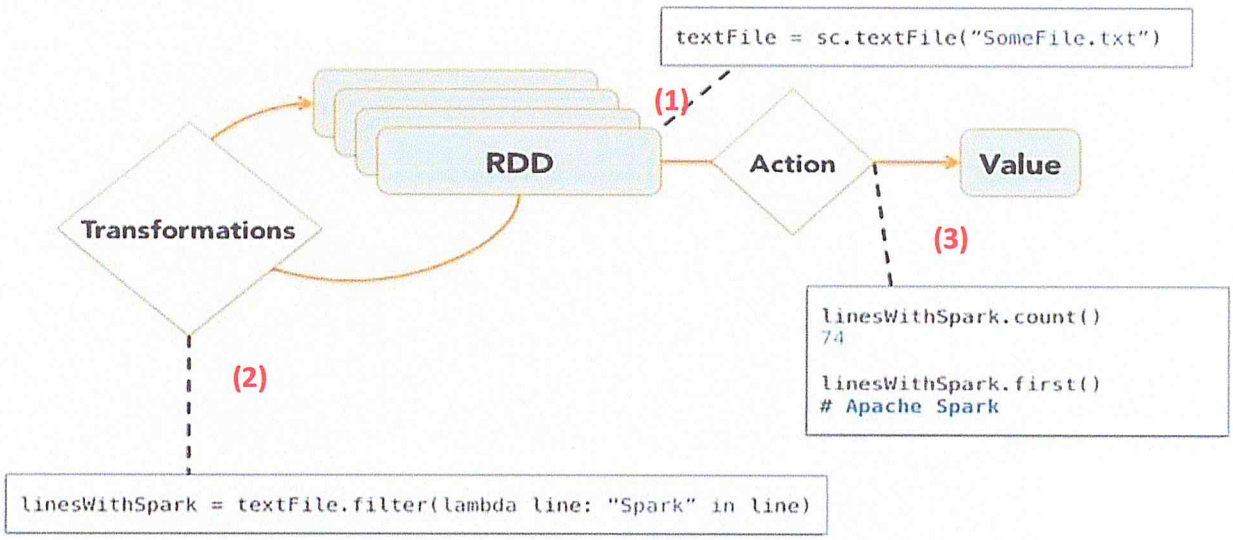

Figure 38: Manipulation des RDDs.

(1) : Le chargement de la source de donné signifie la création d'un RDD ;

(2) : A partir du premier RDD nous créons un nouveau RDD, et ce en appliquant une opération de type de transformation. Ce RDD contient uniquement les lignes ou le mot « Spark » figure.

(3) : Nous appliquons des opérations de types action qui fournit un résultat. Dans notre exemple le résultat est le nombre de ligne qui contient le mot « Spark » ainsi que la première ligne.

4.2 Exemple 2: Ma première application Spark

Il faut d'abord créer un "contexte Spark" pour se connecter au cluster. Puisque nous écrivons du Java, la classe que nous utilisons est JavaSparkContext et nous lui passons un objet de configuration contenant :

- un nom d'application (utile lorsque l'application est déployée en cluster)
- la référence vers un cluster Spark à utiliser,"local" pour exécuter les traitements au sein de la JVM courante.

```
SparkConf conf =new SparkConf()
.setAppName("myapp")
.setMaster("local");
JavaSparkContext sc =new JavaSparkContext(conf);
```

Nous pouvons ensuite écrire la suite de traitements et récupérer le résultat :

```
long count = sc.textFile("etudiant.csv")
.filter(line -> line.contains("licence"))
.count();
System.out.println(count);
```

Détaillons ce code :

- Nous commençons par demander à Spark de lire le fichier CSV. Spark sait nativement lire un fichier texte et le découper en lignes.

  La méthode utilisée est textFile() et le type retourné est JavaRDD<String> (un RDD de Strings).

  ```
  sc.textFile("etudiant.csv")
  ```

- Nous essayons de garder seulement les étudiants du cycle licence. La méthode utilisée est filter() et elle ne modifie par le type retourné qui reste donc JavaRDD<String>.

  ```
  .filter(line -> line.contains("licence"))
  ```

- Enfin, nous comptons les éléments du RDD (c.à.d. le nombre des étudiants qui font licence).L'opération finale count() est utilisée et un long est retourné.

```
.count()
```

Voici un extrait de ce qui est produit sur la console :

```
...
14/10/29 17:09:54 INFO FileInputFormat: Total input paths to process : 1
14/10/29 17:09:54 INFO SparkContext: Starting job: count at FirstSteps.java:26
14/10/29 17:09:54 INFO DAGScheduler: Got job 0 (count at FirstSteps.java:26) with 1 output partitions (allowLocal=false)
...
14/10/29 17:09:54 INFO TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, localhost, PROCESS_LOCAL, 1242 bytes)
14/10/29 17:09:54 INFO Executor: Running task 0.0 in stage 0.0 (TID 0)
14/10/29 17:09:54 INFO HadoopRDD: Input split: file:/Users/aseigneurin/dev/spark-samples/arbresalignementparis2010.csv:0+8911344
...
14/10/29 17:09:55 INFO SparkContext: Job finished: count at FirstSteps.java:26, took 0.475835815 s
32112
```

Spark a exécuté les traitements en local, au sein de la JVM.

Le fichier a été lu en un seul bloc. En effet, celui-ci fait 8,5 Mo et, par défaut, Spark découpe les fichiers en blocs de 32 Mo.

Le résultat (32112) est obtenu en moins d'une demi-seconde. Ce temps d'exécution n'est pas, en soi, impressionnant, mais nous verrons la puissance du framework Spark lorsque nous manipulerons des fichiers plus volumineux.

Conclusion

Apache spark possède des APIs qui permettent de faire pas mal des transformations et d'options sur les RDDs, ainsi pour la configuration de cluster. Pour plus de détail visiter le site officiel, il décrit le tout nettement.

Site officiel de Spark :

https://spark.apache.org/docs/2.0.0-preview/

Web

[06] LEBORGNE, Mathieu SANTEL. Entrepot de Donnees. [En ligne] [Citation : 17/03/ 2016]URL :

http://igm.univmlv.fr/~dr/XPOSE2005/entrepot/index.html.

[11]Yazid Grim et Fleur-Anne.Blain. Conception d'un entrepôt de données (Data Warehouse). www.developpez.com. [En ligne] [Citation : 19/ 03/ 2016.]URL :http://grim.developpez.com/cours/businessintelligence/concepts/conception-datawarehouse/#LII-B.

[15] Grim, Yazid. OLAP, les fondamentaux. www.developpez.com. [En ligne] 07/ 05/ 2008. [Citation : 24/ 03/ 2016.] URL :http://grim.developpez.com/articles/concepts/olap/#LIV-B

[23] ELOMARI, A. Surrogate Key (Clé de substitution).[En ligne]2013 [Citation : 20/ 05/ 2016.]URL :

http://systemeetl.blogspot.com/2013/01/surrogate-key-cle-de-substitution.html

[24]Hkancel. Mettre en place le Change Data Capture (CDC) dans Integration Services (SSIS) 2012. mcnext.com. [En ligne]2007.[Citation:20/5/2016]URL :

https://mcnextpost.com/2013/11/26/mettre-en-place-le-change-data-capture-cdc-dans-integration-services-ssis-2012/.

[25] Fernandez, Alian. Piloter la Performance : Méthodes, Outils et Techniques. www.piloter.org. [En ligne] [Citation : 12/ 06/ 2016.]URL :

http://www.piloter.org/.

[26] ETL,Modélisation dimensionelle et datawarehowsing. www.dwfacile.com. [En ligne] 2004. [Citation : 20 /07/ 2016.] URL :

http://www.dwfacile.com/stag_or_not.htm.

[29] Le Big Data et le data mining. www.astrosurf.com. [En ligne] [Citation : 09/ 07/ 2016.]URL :

http://www.astrosurf.com/luxorion/big-data-mining.htm.

[32] A.Saliha, Modélisation objet avec UML : le diagramme de séquence. [En ligne]2011. [Citation 10/07/2016.]URL : http://salihayacoub.com/420Ke2/Semaine%209/DiagrammeDesequences.pdf.

[34] Eclipse (logiciel). Techno-Science.net. [En ligne] 06 /06/ 2004. [Citation : 06/ 07/ 2016.]URL :

http://www.techno-science.net/?onglet=glossaire&definition=517.

[35]Tropardy, Thibaut. Utiliser Maven pour un projet Web. [En ligne] [Citation :06/08/2016] URL :

http://w3blog.fr/2013/05/18/utiliser-maven-pour-un-projet-web/.

[36] BENDJOUDI, DR. AHCÈNE. Manuel d'utilisation du cluster IBNBADIS. http://www.rx-racim.cerist.dz/. [En ligne] [Citation : 06/ 08/ 2016.]URL : http://www.rx-racim.cerist.dz/?page_id=26%20cluster%20IBNBADIS.