الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي والبحث العلمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

جـامعة سعـد دحلب البليـــدة 1
Université Saad Dahleb, Blida 1
Faculté des sciences
Département d'informatique

Mémoire présenté par :
BOUHAMADOU Rahma

Proposé par :
Mr. BALA Mahfoud

Pour l'obtention du diplôme de MASTER

Spécialité : Informatique
Option : Ingénierie des logiciels

Thème :

Mise en œuvre d'une plateforme d'intégration basée sur une librairie de fonctionnalités d'ETL

Soutenue en: 2015, devant le jury composé de :

- Mme L.Ouahrani, maître assistante, USDB - Présidente
- Mme Toubaline, maître assistante, USDB - Examinatrice
- M. H.Derrar, maître assistant, USDB - Examinateur
- M. M.Bala, maître assistant, USDB - Promoteur

Promotion : 2014/2015

« Face au réel, ce qu'on croit savoir clairement offusque ce qu'on devrait savoir. Quand il se présente à la culture scientifique, l'esprit n'est jamais jeune. Il est même très vieux, car il a l'âge de ses préjugés. Accéder à la science, c'est, spirituellement, rajeunir, c'est accepter une mutation brusque qui doit contredire un passé. »

Gaston Bachelard

Résumé

Dans le cadre de ce travail, nous nous intéressons à la phase d'intégration des données d'un système décisionnel. Celle-ci est basée sur un processus ETL (Extracting, Transforming, Loading), chargé de préparer les données issues de diverses sources dans une zone intermédiaire appelée Data Staging Area (DSA) avant de les charger dans un ED. Dans ce contexte, notre contribution est de définir le processus ETL à un niveau de granularité fin grâce à un ensemble de fonctionnalités d'ETL implémentées séparément et pouvant être intégrées et synchronisées pour répondre aux objectifs d'intégration de données. Chacune de ces fonctions, prenant en charge un aspect particulier dans le processus, est implémentée en se basant sur une architecture et un algorithme proposés. Le système se compose de deux parties : l'ensemble des fonctions constituant une librairie d'ETL et une plateforme basée sur une interface unique interactive pour la définition, le paramétrage ainsi que l'exécution de processus ETL.

Mots clés :

Systèmes décisionnels, entrepôt de données (ED), intégration de données, processus ETL, fonction ETL.

Abstract

In this work, we are interested in the data integration phase of a decision-support system (DSS). It is based on an ETL (Extracting, Transforming, Loading) process, which prepares the data from various sources in a Data Staging Area (DSA) before loading them in the data warehouse (DW). In this context, our contribution is to de ine the ETL processes in a ine-grained level thanks to a set of ETL functionalities implemented separately and that can be integrated and synchronized to meet the data integration objectives. Each of these functions, supporting a particular aspect in the process, is implemented according to a proposed architecture and algorithm. The system consists of two parts: the ETL functions library and a platform based on a single interactive interface for setting, parameterization and execution of ETL processes.

Keywords:

Decision-support systems, data warehouse (DW), data integration, ETL process, ETL functionality.

ملخص

في هذا العمل, نهتم بمرحلة تكامل البيانات من نظام صنع القرار. لأنه يقوم على عملية ETL (استخراج، تحويل، تحميل)، لإعداد البيانات من مصادر مختلفة في منطقة ثانوية المعروفة باسم منطقة تدريج البيانات (DSA) قبل التحميل في مستودع البيانات (ED). في هذا السياق، مساهمتنا هي تحديد عمليات ETL إلى مستوى مفصل من خلال تنفيذ مجموعة من وظائف الـ ETL بشكل منفصل ويمكن أن تكون متكاملة ومتزامنة لتحقيق أهداف تكامل البيانات. كل من هذه الوظائف، تدعم جانب معين في هذه العملية، يتم تنفيذها على أساس هندسة وخوارزمية مقترحة. ويتكون النظام من جزأين: مجموعة من الوظائف التي تشكل مكتبة ETL ومنصة تستند إلى واجهة تفاعلية واحدة لتعريف، تحديد المعايير وتنفيذ عمليات الـــ ETL.

كلمات مفتاحية:

أنظمة صنع القرار, مستودع البيانات, تكامل المعلومات, عملية الـ ETL, وظيفة ETL.

Á ma famille,

mes amis

et ceux que j'aime

Remerciements

Au terme de ce travail, je tiens à exprimer ma profonde gratitude et mes sincères remerciements à mon promoteur, **Mr. Bala Mahfoud,** pour la confiance qu'il m'a accordé, son soutien ainsi que pour ses conseils instructifs et sa disponibilité durant toute la période de réalisation de ce travail.

Tous mes sentiments de reconnaissance à tous mes amis que j'ai eu une énorme chance de les avoir connu. Des personnes aussi différentes, mais ayant toutes un grand cœur.

Aussi, je tiens à remercier infiniment toute ma famille, particulièrement mon père et ma mère, pour tous leurs sacri ices et leurs encouragements qu'ils m'ont prodigué tout au long de mes années d'étude.

Et enfin, pour n'oublier personne, je remercie tous ceux qui ont contribué de près ou de loin à l'aboutissement de ce travail.

Table des matières

Liste des tableaux

Liste des algorithmes

Liste des figures

Introduction générale

Contexte général

Nous sommes dans « l'ère de la société d'information ». Cette dernière occupe, particulièrement au sein de l'entreprise, une place de plus en plus stratégique. Collecter, organiser et analyser de nouvelles informations sont devenus des défis auxquels les gestionnaires sont confrontés quotidiennement pour se maintenir face à une concurrence accrue et espérer devenir compétitifs. Il existe aujourd'hui une panoplie d'outils sophistiqués dédiés à l'analyse de données et l'aide à la décision. Ils permettent essentiellement la sélection et l'extraction des données à partir de diverses sources traitant sur l'activité de l'entreprise, leur nettoyage et leur intégration, le stockage des données préparées dans un entrepôt de données et enfin des applications de reporting, d'analyse des données en ligne, de statistiques et de fouille de données.

L'industrie de l'entreposage de données a mûri avec les recherches du groupe Kimball, depuis qu'il a publié, en 1996, la première édition du *Data Warehouse Toolkit* [Kimball, 2002]. Depuis lors, la voie a été ouverte pour les entreprises, et l'entreposage de données a été adopté par des organisations de toutes les tailles.

Les données pertinentes pour la prise de décision sont collectées à partir des sources et intégrées dans l'ED. L'entreposage de ces données se fait par le biais des processus d'extraction, de transformation et de chargement ou processus ETL [Vassiliadis, 2009]. Le processus ETL (Extract - Transform - Load) constitue une phase très importante dans un projet décisionnel. Il est très complexe en termes de flux de données et une grande variété de tâches de transformation.

Problématique

Les applications décisionnelles permettent à l'entreprise d'avoir une synthèse de son activité et de faciliter la prise de décisions. Cependant, elles peuvent se révéler dangereuses si les données sources ne sont pas fiables. Ce qui engendre des résultats catastrophiques pour l'entreprise. C'est pourquoi, compte tenu de la diversité des sources de données, l'intégration de données (ETL) représente la partie critique et complexe d'un projet décisionnel.

A présent, les efforts d'amélioration des outils d'ETL et d'intégration de données sont concentrés sur l'ajout de fonctionnalités, aux dépens de deux facteurs essentiels : la facilité d'utilisation et l'évolutivité sans perte de performance.

Objectif

Face à la complexité du processus ETL, nous nous intéressons dans ce travail à définir, concevoir et implémenter les fonctionnalités de base de l'ETL. Dans une seconde étape, il s'agit d'intégrer et de synchroniser ces fonctionnalités afin de constituer un processus. La définition d'un processus ETL à un niveau de granularité fin permettra plus de robustesse, de facilité de maintenance et de performance. A titre de tests, nous avons choisi de travailler sur quelques fonctions courantes à savoir Changing Data Capture (CDC), Slowly Changing Dimension (SCD), Surrogate Key Pipeline (SKP).

L'objectif est de mettre en place une plateforme qui dispose d'un ensemble d'éléments permettant de configurer et d'exécuter des processus ETL. Elle doit se baser sur une bibliothèque de fonctionnalités d'ETL qui représente le noyau du système.

L'outil à développer devra présenter des interfaces souples et simples qui facilitent à l'utilisateur la construction graphique du processus ETL à partir des sources de données qui doivent passer par un ensemble de fonctions d'ETL en vue de leur nettoyage, filtrage, conversion, agrégations et enfin leur chargement dans un entrepôt de données.

Organisation du mémoire

Pour présenter nos contributions, le présent mémoire est organisé de la manière suivante :

- **Partie 1 : Etat de l'art sur les systèmes décisionnels et le processus ETL**

Cette partie présente les généralités sur le domaine décisionnel et, plus particulièrement, sur le processus ETL. Elle est composée de deux chapitres :

- **Chapitre I :** généralités sur les systèmes d'informations décisionnels.

Il expose les systèmes d'information décisionnels ainsi que la notion d'entreposage de données.

- **Chapitre II :** L'intégration de données (Processus ETL).

Ce chapitre présente la définition du processus ETL et ses approches.

- **Partie 2 : Le processus ETL**

Dans cette partie, nous détaillons les étapes du processus ETL, en deux chapitres :

- **Chapitre III :** Flux de données dans un processus ETL.

Il présente les tâches ETL et le parcours des données entre ses étapes, de l'extraction au chargement passant par la transformation.

- **Chapitre IV :** Définition d'un processus ETL selon ses fonctionnalités de base.

On aborde ici la notion de fonctionnalité et son intérêt dans l'approche ETL.

- **Partie 3 : Mise en œuvre du système**

Cette dernière partie présente la démarche suivie pour la mise en place de la plateforme d'intégration basée sur une librairie de fonctionnalités d'ETL. Elle est composée aussi de deux chapitres :

- **Chapitre V :** Spécification des besoins et conception du système.

Ce chapitre présente la spécification des besoins du système et sa conception.

- **Chapitre VI :** Implémentation.

Il expose l'architecture du système, les méthodes, les outils de développement et les résultats des tests.

PARTIE 1

CHAPITRE I

GENERALITES SUR LES SYSTEMES D'INFORMATION DECISIONNELS

1. Introduction

Le système d'information (SI) a pour objectif de faciliter l'établissement, la mise en œuvre de la stratégie et supporter la réalisation des activités de l'entreprise. Il est construit à partir des exigences des métiers, des processus définis par l'entreprise et il est constitué de l'ensemble des moyens (humains, logiciels, matériels) utilisés pour collecter, stocker, traiter et communiquer les informations.

Les systèmes d'information ont été structurés en deux sous systèmes : l'un dit « **opérationnel** » ou de « **gestion** », qui couvre toutes les activités de gestion du fonctionnement de l'entreprise (marketing, vente, achat, production, logistique, finance, ressources humaines), et qui prend en charge la réalisation des opérations quotidiennes, et l'autre dit « **décisionnel** » qui fournit des informations pour définir la stratégie, piloter les opérations et analyser les résultats.

Dans ce chapitre, nous présentons la différence entre les systèmes opérationnels et les systèmes décisionnels et nous décrivons les concepts principaux des entrepôts de données et de l'analyse multidimensionnelle.

2. SI opérationnel versus SI décisionnel

Un système décisionnel est 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. …) [Bruley, 2010].

Dans un système opérationnel, les données ne sont conservées que sur une courte période. Elles sont détaillées, personnelles, identifiées et représentent généralement en volume quelques centaines de mégaoctets, voire quelques gigaoctets.

Dans un système décisionnel, les données stockées dans les entrepôts de données sont historisées et peuvent être agrégées. Il présente différents niveaux de granularité et doit être capable de livrer des informations aussi détaillées qu'agrégées. La base peut atteindre des volumes considérables, de l'ordre du téraoctet, voire plus.

Données opérationnelles	**Données décisionnelles**
Orientées application (verticales), précieuse au moment de l'accès	Orientées activité (transversales), condensées, représentent des données historiques
Mise à jour interactive possible	Pas de mise à jour interactive
Accès par une personne à la fois	Utilisées par l'ensemble des analystes
Haute disponibilité en continu	Haute disponibilité ponctuelle
Unique (pas de redondance)	Redondances rares
Petite quantité utilisées par une requête	Grande quantité de données utilisées par les requêtes
Réalisation de l'opération au jour le jour	Cycle de vie différent
Forte probabilité d'accès	Faible probabilité d'accès
Utilisées de façon répétitive	Utilisées de façon aléatoire

Tableau 1: Données opérationnelles versus données décisionnelles [J-S, 2010]

Les requêtes des bases de données opérationnelles s'effectuent sous forme de transactions qui lisent et écrivent un nombre réduit de lignes dans différentes tables liées par des liens référentiels. Ce type de requêtes est dit « On line Transaction Processing » (OLTP). Au contraire, le type de requêtes effectuées sur un entrepôt de données est appelé « On Line Analytical Processing » (OLAP).

Les requêtes OLAP nécessitent la lecture d'une énorme quantité de données pour produire un ensemble de valeurs numériques.

Les différences principales entre les systèmes OLTP et OLAP résumées par Kelly [Kelly, 1997] sont soulignées dans le tableau suivant :

	OLTP	**OLAP**
Utilisation	Gestion des transactions	Aide à la Décision
Conception	Orientée applications	Orientée utilisateurs
Fréquence	Quotidienne	Sporadique
Données	Récentes, détaillées	Historiques, multidimensionnelles, agrégées
Source	BD unique	Plusieurs BDs
Utilisation	Répétitive	Ad-hoc
Accès	Programmes précompilés	Outils d'Analyse
Nb lignes accédées	Dizaines	Milliers
Type d'utilisateur	Opérateurs	Décideurs
Nb utilisateurs	Milliers	Centaines
Performance	Elevée	Basse
Dimension BD	Giga-octets	Téraoctets

Tableau 2: Différences principales entre les systèmes OLTP et OLAP

3. Les entrepôts de données (Data Warehouse)

3.1 Définition :

Un entrepôt de données (ED) est vu comme un ensemble de données thématiques, intégrées, non volatiles et historisées pour des fins décisionnelles [Inmon, 1996]. IL regroupe les données pertinentes utiles aux analyses et la prise de décision.

- **Thématiques :** Les données sont organisées par thèmes ou par sujets majeurs de l'entreprise.

- **Intégrées :** Les données proviennent de diverses sources de données. L'hétérogénéité des ces dernières donne lieu à des conflits syntaxiques et

sémantiques. L'intégration des données permet de résoudre le problème et d'avoir une représentation uniforme et cohérente des données lors de leur chargement au niveau de l'ED.

- **Non volatiles :** Les données stockées dans l'ED n'ont pas vocation à être supprimées. Ceci permet de garder la traçabilité des informations à long terme.

- **Historisées :** Les données sont datées, ce qui permet d'effectuer des analyses comparatives dans le temps.

3.2 Modélisation d'un entrepôt de données (ED):

Les modèles de données d'un ED ont été conçus pour faciliter les analyses décisionnelles [Kimball, 1996].

Typiquement, la modélisation d'un ED repose sur trois niveaux d'abstraction :

- Le niveau conceptuel qui prend en compte les besoins des décideurs en faisant abstraction des aspects techniques.
- Le niveau logique où l'on fait le choix d'un modèle de stockage.
- Le niveau physique où l'ED est implanté sur un Système de Gestion de Base de Données (SGBD) particulier. [Atigui, 2013]

3.2.1 Niveau conceptuel :

Les modèles de conception des systèmes transactionnels OLTP ne sont pas adaptés aux systèmes OLAP dont les requêtes sont souvent très complexes. Pour cela, Ralph Kimball a proposé une nouvelle approche de modélisation: la modélisation multidimensionnelle [Kimball, 1996]. Cette modélisation est aujourd'hui reconnue comme la modélisation la plus appropriée aux besoins d'analyse et de prise de décision. Elle offre une structuration et une organisation des données facilitant leur analyse et a pour principal objectif d'avoir une vision multidimensionnelle des données.

Les données de l'entrepôt sont représentées par des schémas qui organisent les données en termes **de sujets** et **d'axes d'analyses** [Kimball, 1996].

Le modèle multidimensionnel renferme les deux concepts fondamentaux de **fait** et de **dimension**.

- **Fait :** représente la donnée observable ou le sujet d'analyse. Il est composé d'un ou plusieurs indicateurs appelés « **mesures** » qui correspondent aux informations liées au sujet d'analyse.

- **Dimension :** représente l'axe d'analyse selon lequel l'entreprise analyse le fait. Une dimension est composée de **Paramètres** en fonction desquels les mesures sont étudiées. Ils sont organisés selon de hiérarchies, de la granularité la plus faible (paramètre racine) à la granularité la plus haute.

- **Cube de données :** un cube de données est une structure multidimensionnelle. Il est constitué d'une table de faits avec des indicateurs et leurs dimensions (les axes par rapport auxquels ceux-ci sont analysés). La figure 1 décrit un cube destiné pour l'analyse des ventes par rapport aux dimensions « Produit », « Région » et «Temps ».

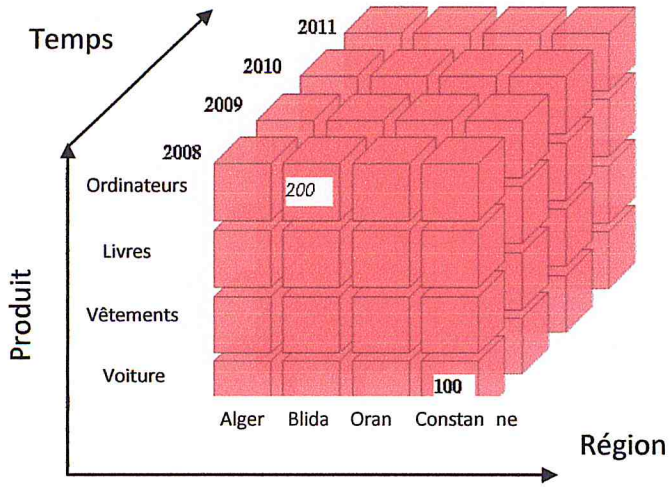


Figure 1 : Cube de données ventes

3.2.1.1 Schéma en étoile :

Le modèle de données en étoile doit son nom à sa forme. Il se présente comme une étoile dont le centre est la table des faits et les branches sont les tables de dimension. Le schéma en étoile est illustré par la figure 2.

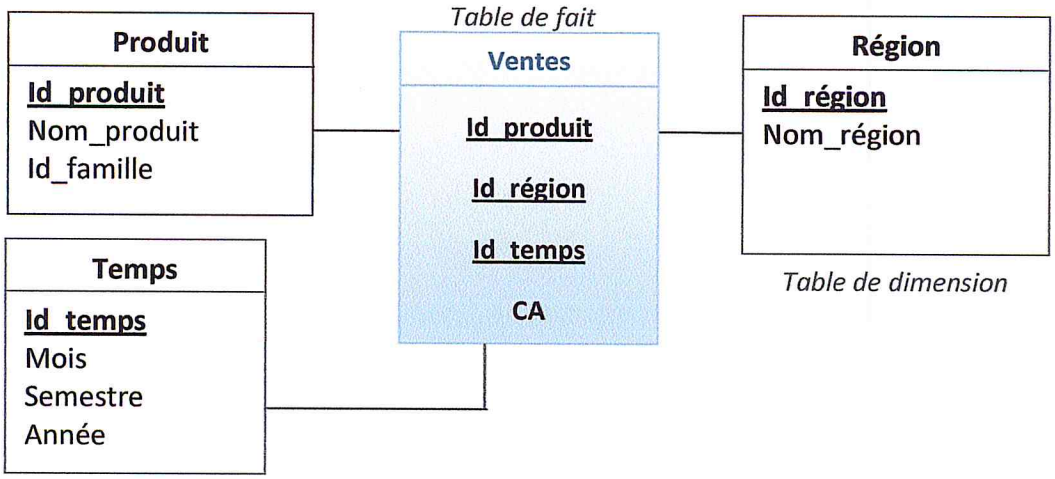


Figure 2 : Schéma en étoile

3.2.1.2 Schéma en constellation :

Ce n'est rien d'autre que plusieurs modèles en étoiles liés entre eux par les dimensions communes. La figure suivante illustre le schéma en constellation :

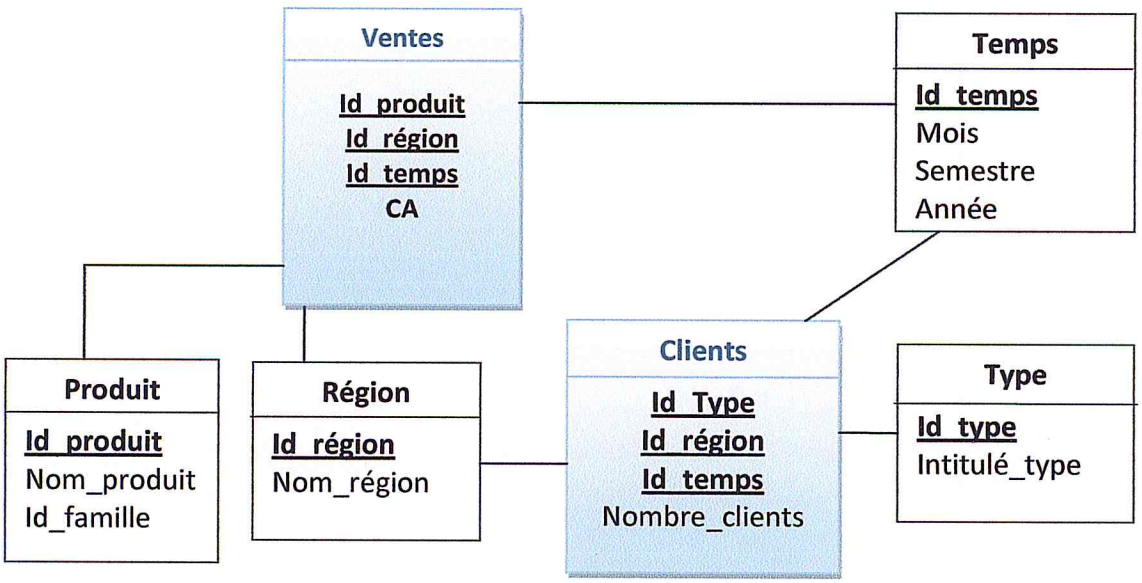


Figure 3: Schéma en constellation

3.2.1.3 Le schéma en flocon :

Le modèle de données en flocon est identique au modèle en étoile sauf que ses branches sont éclatées en hiérarchie. Il est plus adapté pour des usages bien spécifiques et est généralement justifié par l'économie d'espace de stockage. Cependant, il peut s'avérer moins compréhensible pour l'utilisateur final et très couteux en termes de performances. La figure 4 illustre le schéma en flocon de neige :

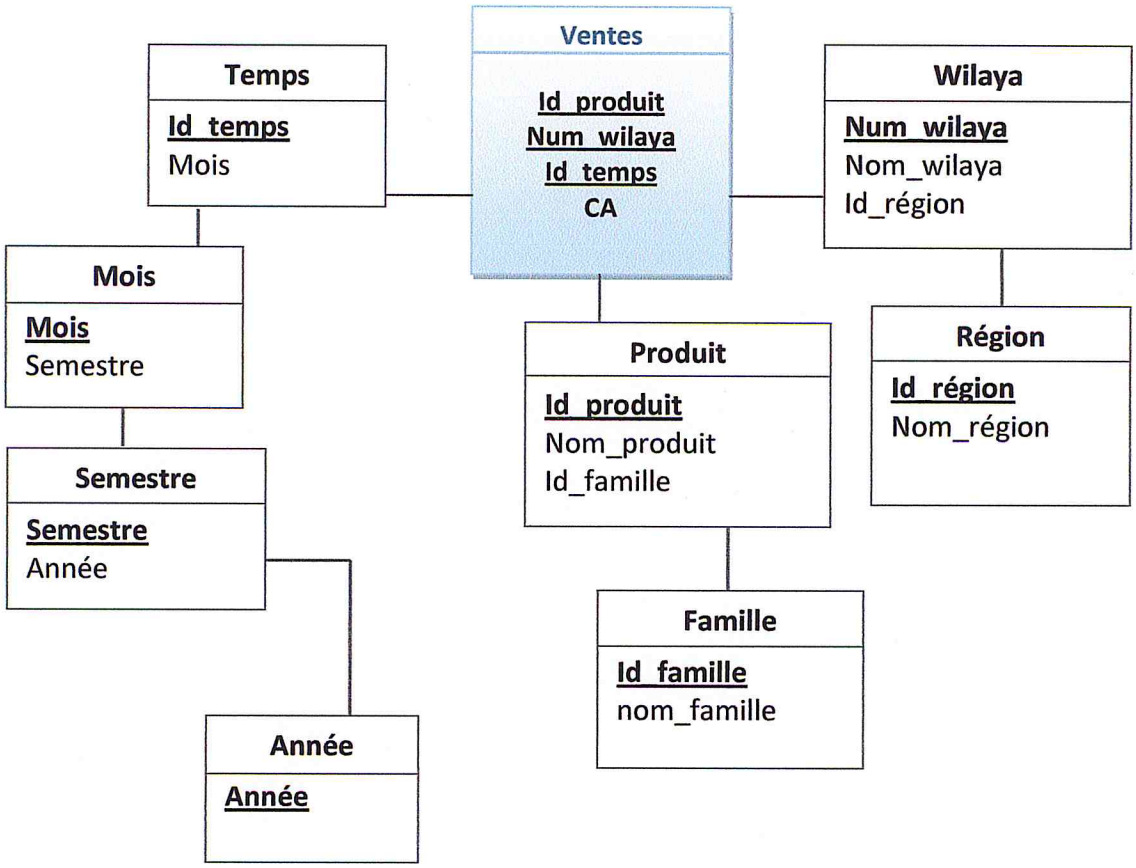


Figure 4 : Schéma en flocon de neige

	Avantages	Inconvénients
Schéma en étoile	• Facilité de navigation. • Performances : nombre de jointures limité.	• Toutes les dimensions ne concernent pas les mesures. • Redondances dans les dimensions.
Schéma en flocon	• Réduction du volume (Economie d'espace) • Permettre des analyses par pallier (drill down/Roll up) sur la dimension hiérarchisée.	• Navigation difficile. • Nombreuses jointures. • Requêtes moins performantes.
Schéma en constellation	• Navigation facile. • Dimensions réutilisées (vue orientée entreprise)	• Dimensions redondantes. • Augmentation du volume.

Tableau 3: Avantages et inconvénients des schémas en étoile et en flocon de neige

3.2.2 Niveau logique :

A ce niveau, plusieurs possibilités sont envisageables pour modéliser la structure d'un entrepôt. Le plus courant est d'utiliser un modèle spécifique à l'environnement tels que MOLAP (Multidimensional - OLAP) ou HOLAP (Hybrid - OLAP) ou un modèle relationnel dit **ROLAP (Relational OnLine Analytical Processing)**.

3.2.2.1 OLAP :

OLAP, acronyme de On-Line Analytical Processing, désigne le système permettant la prise de décision stratégique rapide et fiable sur des données modélisées en multidimensionnel. [Grim, 2008]

CODD, père des bases relationnelles, a énuméré, en 1993, les 12 règles qu'un environnement OLAP doit respecter :

- **Conception et vue multidimensionnelle :** un outil OLAP doit se baser sur un modèle multidimensionnel pour faire de l'analyse.
- **Transparence :** la technologie utilisée, la conception ainsi que toutes les spécifications techniques doivent être invisibles à l'utilisateur final.
- **Accessibilité :** les outils OLAP doivent permettre d'accéder les données de façon à produire de la connaissance rapide. Une information pertinente et on-time doit être fournie en tout temps.
- **Rapidité :** les montées de charges ne doivent pas freiner l'analyste. L'outil doit pouvoir supporter de grosses requêtes (c'est la caractéristique la plus difficile à satisfaire).
- **Architecture Client Serveur :** pour un accès uniforme et des traitements plus rapides et plus sophistiqués.
- **Dimensions génériques :** essayer, autant que possible, d'avoir une unicité dans la définition des dimensions.
- **Gestion des matrices creuses :** en mathématiques, les matrices creuses sont des matrices qui contiennent beaucoup de zéros. En informatique, il existe des algorithmes qui utilisent cette spécificité pour optimiser le stockage de ce type de matrices. Les performances sont, en général, au rendez vous. Les outils OLAP doivent avoir cette capacité d'optimisation d'espace de stockage par la gestion des matrices creuses.
- **Multi-utilisateurs :** les outils OLAP sont destinés à un accès concurrent.
- **Croisement inter dimensions illimité :** l'utilisateur ne doit avoir aucune restriction quand au nombre de croisements qu'il fait entre les dimensions.
- **Intuitifs :** les utilisateurs d'outils OLAP ne sont pas forcément informaticiens. Il est donc nécessaire d'offrir des solutions adaptées à leur style cognitif.
- **Affichage flexible :** l'utilisateur doit pouvoir aisément " arranger " son résultat au format désiré.
- Nombre illimité de dimensions et de niveaux d'agrégation.

3.2.2.2 ROLAP :

ROLAP (Relational OLAP). Comme son nom l'indique, il utilise le concept relationnel pour stocker des données modélisées dans le format multidimensionnel. Les analyses sont transformées en requêtes SQL classiques qui sont exécutées sur les données du cube en utilisant un SGBD relationnel. ROLAP 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.

Les outils modernes permettent aussi la gestion du cache, l'optimisation des requêtes et la création de tables d'agrégats à la demande.

La technologie ROLAP perd beaucoup de terrain face à ces concurrents car son principal inconvénient est qu'elle implique beaucoup de lourdeur et d'émulation pour son implémentation. On simule des opérations sur des matrices avec du SQL, et le fait de simuler deux conceptions apparemment différentes apporte son lot de gestion lourde et de manque de performances.

Leurs principaux avantages sont :
- Exploitation des capacités du relationnel qui est un standard bien établi et maîtrisé, ce qui facilite leur intégration dans les SGBD relationnels existants.
- Stockage de grandes quantités de données. [Grim, 2008]

3.2.2.3 MOLAP (Multidimensional OLAP) :

Dans un système MOLAP, les données sont stockées dans une base de données multidimensionnelles le plus souvent propriétaires avec un problème des limitations de volume de données, mais il fournit un temps de réponse de requêtes très rapide. [Grim, 2008]

3.2.2.4 HOLAP (Hybrid OLAP) :

HOLAP, pour Hybride OLAP est une architecture hétérogène composée de tout ou partie des architectures précitées : du MOLAP pour les données sommaires et du ROLAP pour les données détaillées. [Grim, 2008]

4. Conclusion

Dans ce chapitre nous avons présenté les concepts principaux des systèmes d'entrepôts de données et nous avons décrit les concepts à base de l'analyse multidimensionnelle, structures de données et les niveaux de modélisation d'entrepôts de données. L'intégration des données hétérogènes, issues de différentes applications de production, alimentant un data warehouse représente une tâche très importante dans un projet décisionnel. Pour cela on va présenter dans le chapitre suivant l'état de l'art du processus d'intégration de données ETL.

PARTIE 1

CHAPITRE II

L'INTEGRATION DE DONNEES

« PROCESSUS ETL »

1. Introduction

Les entreprises avaient mis beaucoup d'emphase sur la présentation et l'utilisation finale d'un ED. Avec l'accroissement du volume de données l'accent est plutôt mis sur la phase d'intégration du système décisionnel qui est basée sur le processus d'ETL.

L'ETL n'est pas un simple programme d'alimentation de l'ED et ne doit pas être traité de la sorte. Il s'agit plutôt d'un système complexe chargé de se connecter sur diverses sources, d'extraire les données considérées pertinentes par rapport aux objectifs d'analyse, de les transformer en vue de les homogénéiser et les intégrer. Le processus d'ETL sera chargé, enfin, de charger les données dans l'ED où celles-ci seront publiées pour des applications d'analyse et d'aide à la décision. En effet, Kimball a décomposé l'ETL en 38 sous-systèmes et l'a estimé à 70% en termes de coût et de temps d'un projet décisionnel [Kimball, 2004].

Dans ce chapitre, nous allons voir les différentes approches d'intégration de données et les processus d'ETL.

2. Les approches d'intégration de données :

2.1 Approche matérialisée :

Cette approche consiste à centraliser physiquement, dans un entrepôt de données (ED), l'ensemble de données consolidées à partir de diverses sources (catalogues électroniques, bases de données relationnelles, etc.)

La conception d'un système d'intégration selon une architecture matérialisée est similaire à celle des entrepôts de données. Le processus de construction d'un système d'intégration matérialisé se compose d'étapes correspondant aux processus d'ETL (Extract, Transform, Load) [Giraldo, 2003].

L'avantage principal d'une telle approche est que l'interrogation d'un ED se fait directement sur les données de l'entrepôt et non pas sur les sources originelles. On peut donc utiliser les techniques d'interrogation et d'optimisation des bases de données traditionnelles. Par contre cette approche exige un coût de stockage supplémentaire et surtout un coût de maintenance causé par les opérations de mises à

jour au niveau de sources de données (toute modification dans les sources locales doit être répercutée sur l'ED).

2.2 Approche virtuelle :

Cette architecture consiste à développer une application chargée de jouer le rôle d'interface entre les sources de données locales et les applications d'utilisateurs. Ce type d'approches a été utilisé par un nombre important de projets. Il repose sur deux composants essentiels : le *médiateur* et l'*adaptateur* [Xuan, 2006].

- **Le médiateur : chargé de la localisation des sources de données et des données** pertinentes par rapport à une requête, il résout de manière transparente les conflits de données. Un ensemble de connaissances sur les sources permet au médiateur de générer un plan d'exécution pour traiter les requêtes d'utilisateurs.
- **L'adaptateur :** un outil permettant à un (ou plusieurs) médiateur(s) d'accéder au contenu des sources d'informations dans un langage uniforme. Il fait le lien entre la représentation locale des informations et leur représentation dans le modèle de médiation.

L'approche médiateur présente l'intérêt de pouvoir construire un système d'interrogation de sources de données sans toucher aux données qui restent stockées dans leur source d'origine. Par contre, le médiateur ne peut pas évaluer directement les requêtes qui lui sont posées car il ne contient pas de données, ces dernières étant stockées de façon distribuée dans des sources indépendantes.

Dans ce travail, nous nous intéressons à l'approche matérialisée, et le tableau suivant présente les avantages et les inconvénients des deux approches :

	Avantages	Inconvénients
Approche virtuelle	• Données toujours fraiches • Facilité d'ajout de nouvelles sources	• Traduction de requêtes • Capacités différentes de sources
Approche matérialisée	• Personnalisation des données (nettoyage, filtrage) • Versions • Performances	• Données pas toujours fraiches • Gestion des mise-à-jour • Gestion des gros volumes de données

Tableau 4: Avantages et inconvénients des approches d'intégration [Doucet, 2007]

3. Définition du processus ETL :

Afin d'alimenter l'entrepôt de données à partir des différentes applications d'entreprise, on utilise une gamme d'outils appelés ETL, pour « Extract, Transform, Load ». Comme le nom l'indique, ces outils permettent d'extraire des données à partir de différentes sources, les transformer et les charger dans la base de données cible, ici l'entrepôt de données.

Le processus de chargement des données à partir des sources vers l'entrepôt comporte trois étapes principales :

- **Extraction de données :** définit les sources de données (généralement hétérogènes) à utiliser pour alimenter l'ED.
- **Transformation de données :** 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. Durant cette phase, les éléments cible de l'ED à charger sont sélectionnées. Il est également indispensable d'établir les relations de correspondance entre les attributs sources et les attributs cibles appelées schéma de mappage.
- **Chargement de données :** c'est la dernière phase de l'alimentation d'un ED. Il s'agit d'insérer les données dans l'ED. Les tâches de chargement de l'entrepôt ont généralement lieu de façon périodique.

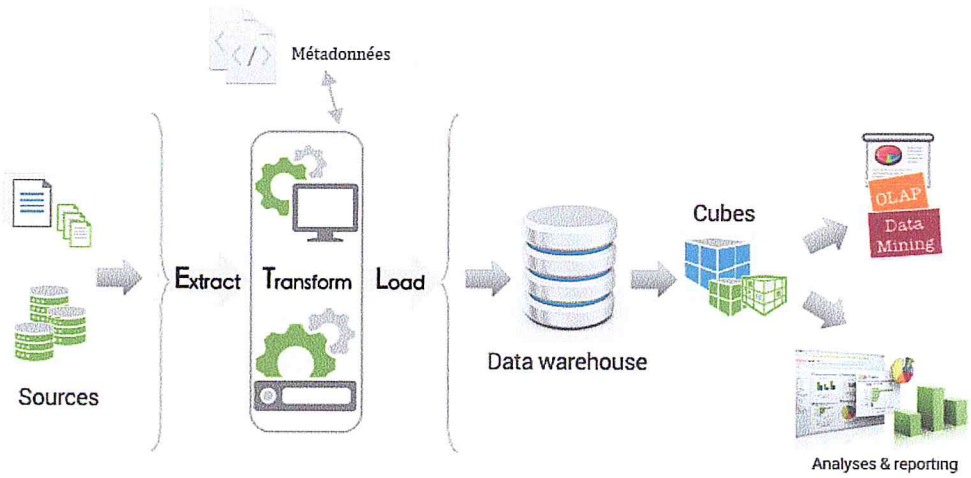


Figure 5: Processus ETL

4. Les structures de données ETL

4.1 Sources de données :

L'ETL peut prendre en charge différentes natures de sources de données, tant en entrée qu'en sortie, les principales étant bien sûr les SGBD relationnels, les flux XML et il peut s'agir également de fichiers à formats fixes ou avec séparateurs (CSV), de fichiers Excel, etc.

- **Bases de données relationnelles :**

Les bases de données sont des données structurées. Ces dernières sont des informations organisées et classées en vue de faciliter leur lecture et leur traitement grâce aux SGBDs.

- **Fichiers plats :**

➢ **CSV :**

Le format CSV (Comma Separated Values) en français, « valeurs séparées par des virgules ». Il s'agit d'un format de fichiers ouvert qui permet de stocker les données d'un tableau. Chaque ligne du fichier correspond à une ligne du tableau, les colonnes sont en général séparées par des virgules. On peut très bien remplacer les virgules par des tabulations ou tout autre caractère. Ce format est utilisé pour échanger de manière interopérable des données de tableurs, bases de données, annuaires, *etc.* entre logiciels différents et/ou plateformes différentes.

➢ **XML :**

Le langage XML, l'acronyme de *eXtended Markup Language*, est un format textuel qui permet de créer des documents contenant des données semi-structurées.

Les données semi-structurées peuvent se voir comme une relaxation du modèle relationnel classique, un des fondements des bases de données traditionnelles, dans lequel on autorise une structure moins rigide et homogène des « champs de données ». Ce modèle de données c'est révélés très utile dans la représentation de familles de documents variés: multimédia, hypertexte, données scientifiques, etc.

- **Les problèmes des sources de données:**
  - Sources diverses et disparates.
  - Sources sur différentes plateformes et OS.
  - Historique de changement non-préservé dans les sources.
  - Qualité de données douteuse et changeante dans le temps.
  - Structure des systèmes sources changeante dans le temps.
  - Incohérence entre les différentes sources.
  - Données dans un format difficilement interprétable ou ambigu.
- **Exemple :**

Pour de nombreux projets (web ou application mobile), il est nécessaire d'avoir une base de données des pays du monde. Cela permet par exemple d'enregistrer le pays de résidence des utilisateurs ou de proposer un contenu éditorial par pays. Nous proposons une base de données SQL, CSV ou XML des pays du monde :

```xml
<database name="test">
    <!-- Table pays -->
    <table name="pays">
        <column name="id">1</column>
        <column name="code">4</column>
        <column name="alpha2">AF</column>
        <column name="alpha3">AFG</column>
        <column name="nom_fr_fr">Afghanistan</column>
        <column name="nom_en_gb">Afghanistan</column>
    </table>
    <table name="pays">
        <column name="id">2</column>
        <column name="code">8</column>
        <column name="alpha2">AL</column>
        <column name="alpha3">ALB</column>
        <column name="nom_fr_fr">Albanie</column>
        <column name="nom_en_gb">Albania</column>
    </table>
</database>
```

Figure 6: Données en format XML

	A	B	C	D	E	F	G
1	1,"4","AF","AFG","Afghanistan","Afghanistan"						
2	2,"8","AL","ALB","Albanie","Albania"						
3	3,"10","AQ","ATA","Antarctique","Antarctica"						
4	4,"12","DZ","DZA","AlgÃ©rie","Algeria"						
5	5,"16","AS","ASM","Samoa AmÃ©ricaines","American Samoa"						
6	6,"20","AD","AND","Andorre","Andorra"						
7	7,"24","AO","AGO","Angola","Angola"						
8	8,"28","AG","ATG","Antigua-et-Barbuda","Antigua and Barbuda"						
9	9,"31","AZ","AZE","AzerbaÃ¯djan","Azerbaijan"						
10	10,"32","AR","ARG","Argentine","Argentina"						
11	11,"36","AU","AUS","Australie","Australia"						
12	12,"40","AT","AUT","Autriche","Austria"						
13	13,"44","BS","BHS","Bahamas","Bahamas"						
14	14,"48","BH","BHR","BahreÃ¯n","Bahrain"						
15	15,"50","BD","BGD","Bangladesh","Bangladesh"						

Figure 7: Données en format CSV

id	code	alpha2	alpha3	nom_fr_fr	nom_en_gb
1	4	AF	AFG	Afghanistan	Afghanistan
2	8	AL	ALB	Albanie	Albania
3	10	AQ	ATA	Antarctique	Antarctica
4	12	DZ	DZA	Algérie	Algeria
5	16	AS	ASM	Samoa Américaines	American Samoa
6	20	AD	AND	Andorre	Andorra
7	24	AO	AGO	Angola	Angola
8	28	AG	ATG	Antigua-et-Barbuda	Antigua and Barbuda
9	31	AZ	AZE	Azerbaïdjan	Azerbaijan

Figure 8: Données en format relationnel SQL

4.2 Zone de stockage « DSA » :

La zone de stockage « Data Staging Area (DSA) » ou la zone de préparation est une zone temporaire qui sert à stocker les données extraites des systèmes sources. C'est là que s'effectuent les différentes transformations des données. Les données du DSA sont détruites une fois le chargement de l'entrepôt est terminé.

En général le modèle physique du DSA est identique à celui des systèmes sources, nous pouvons ajouter une colonne pour savoir la provenance des données (système source). Avec un peu de recul, le DSA ne doit pas forcément être une base de données, des fichiers plats peuvent suffire à condition que l'outil ETL permette le traitement de ces fichiers comme des tables de données.

- **Avantages :**

Voici quelques raisons pour lesquelles il est préférable d'utiliser la zone de stockage temporaire:

- *Extract Once, transform many* : une seule extraction des systèmes sources, et autant de transformations et de chargement dans le DSA.
- Ne pas impacter le fonctionnement des systèmes sources, surtout dans les cas des systèmes 24/7. Le temps nécessaire à extraire les données, devrait être donc minimal. Pour ce faire, la transformation ne doit pas se faire en même temps que l'opération d'extraction : on extrait les données le plus rapidement possible et on les mets ensuite dans un le DSA où s'effectuent les transformations.

- En cas de problèmes de transformation (plantage lors de la transformation, erreur BD...), on ne sera pas obligé de refaire l'extraction, du moment où la source de la transformation est le DSA.

4.3 Entrepôts de données « DW » :

Après avoir opérer toutes les transformations prévues dans le processus ETL, les données du DSA sont transférées vers l'ED. Ce dernier est central et devrait contenir toutes les données de l'entreprise. Une fois l'entrepôt construit, le décideur peut alors utiliser les outils de fouille de données « *Data Mining* » pour construire les représentations adaptées à ses besoins. Dans certains cas, les données de l'entrepôt sont restructurées pour construire les magasins de données[1] adaptés à un domaine particulier au sein de l'entreprise.

- La **construction d'un entrepôt** de données vise à élaborer une structure, constituée d'une grande quantité de données, apte à assister l'usager en lui facilitant l'exploration et en lui offrant la possibilité d'effectuer des analyses rapides.
- La **facilité d'emploi** est liée à la possibilité pour l'usager d'interroger directement les données en interagissant avec elles, sans avoir à maîtriser des langages d'interrogation ou des interfaces complexes.
- La **rapidité** est obtenue en exploitant une dénormalisation du modèle de données et en mémorisant les agrégations de données.

L'architecture d'un entrepôt de données peut donc être schématisée comme suit :

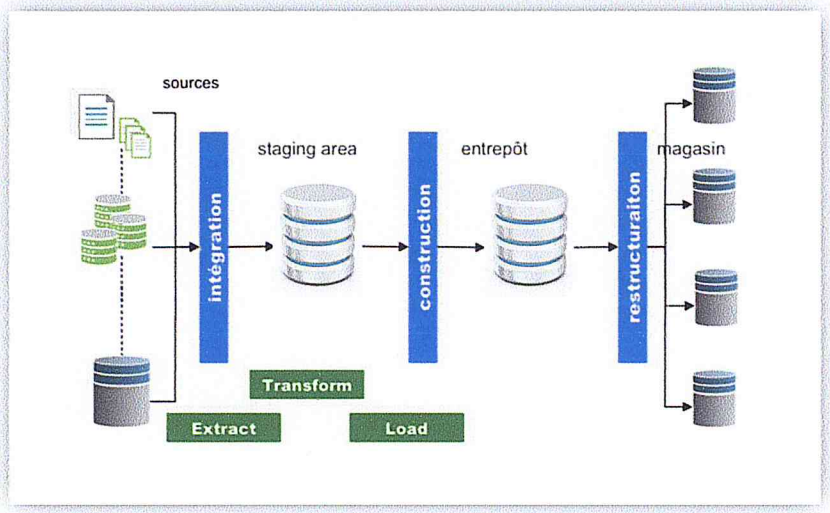


Figure 9: Architecture de l'entrepôt de données [Braesch, 2003]

5. Conclusion

Dans ce chapitre, nous avons vu les différentes approches d'intégration de données dont celle des entrepôts de données (l'approche matérialisée) qui se base sur le processus ETL ainsi que la définition de ce dernier. Nous détaillerons par la suite les phases du processus ETL.

[1]Magasin de données ou « datamart » : Entrepôt de données spécialisé, destiné à ne contenir que les informations élaborées pour un objectif particulier.

PARTIE 2

CHAPITRE III

FLUX DE DONNEES DANS UN PROCESSUS ETL

1. Introduction

Dans ce chapitre, nous exposons les détails sur les étapes du processus d'intégration de données (ETL) ainsi que la nature des données qui y circulent.

2. Extraction de données

La première partie du processus ETL consiste à extraire les données des systèmes sources. L'objectif de cette phase est de convertir les données dans un format unique approprié pour permettre, par la suite, d'opérer les transformations nécessaires.

2.1 Analyse et identification des sources :

Toutes les données sources ne sont forcément pas utiles. Pour cela, il faut une étape d'analyse et d'identification des données sources nécessaires au processus d'intégration.

- **Données de référence (DR) :**

Parmi les données du système d'information, certaines sont plus critiques pour l'activité métier car structurantes et largement partagées entre plusieurs applications transactionnelles, chacune en charge d'une partie selon sa position dans le processus métier attenant. Ce sont **les données de référence**.

Exemples : Référentiel client (commande, facturation), référentiel des fournisseurs (centralisation des achats pour un groupe, …), etc.

Les DR peuvent être classifiées en plusieurs types:

- **DR Maîtres** : objets métiers principaux d'un domaine fonctionnel et structurant pour l'ensemble des applications du domaine (client, produit, fournisseur, …).
- **DR Constitutives**: entrent dans la composition de plusieurs données maîtres (adresses).
- **DR Paramètres**: tables de valeurs ou nomenclature (code postaux, code devises, taux des taxes, …) partagées. [Ploix, 2012]

L'analyse des données et leur constitution est particulièrement importante dans l'intégration de données dans un ED :

- Les objets métiers (et par extension les données de référence) sont les sources autant des **dimensions** (ex. client) que des faits (ex. ventes).
- La distinction fait/dimension est une vue décisionnelle sur les objets métiers en circulation.

2.2 Méthodes d'extraction :

Une fois les données pertinentes identifiées, il faut commencer l'extraction qui doit prendre en considération l'hétérogénéité des données sources. L'outil d'extraction doit être capable d'extraire les données de diverses sources d'où la nécessité de développer des extracteurs (wrappers) pour pouvoir alimenter l'ED.

Un extracteur est associé à chaque source de données. Il est capable de traduire les requêtes et les données depuis le modèle d'une source (sélection et extraction) vers le modèle de l'entrepôt et vice versa.

- **Inconvénient :**

L'inconvénient de cette approche est le risque de faire des extractions erronées, incomplètes et qui peuvent biaiser l'ED. Alors, Il faut aussi gérer les anomalies en les traitant et en gardant une trace.

2.3 Techniques d'extraction :

Pour extraire les données sources, il y a plusieurs technologies utilisables :

- **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 utilitaires de réplication :**

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

- **Des outils spécifiques d'extraction :**

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

3. Transformation et nettoyage de données

3.1 Le nettoyage des données :

Le nettoyage des données est une discipline sur laquelle de nombreux éditeurs travaillent actuellement. Il a pour but de résoudre les problèmes de **consistance** des données au sein de chaque source. Donc, il s'agit d'appliquer des filtres prédéfinis sur les données afin d'attribuer des valeurs cohérentes aux variables mal ou non renseignées ou encore d'harmoniser les formats.

3.1.1 Types d'inconsistances :

Une centaine de types d'inconsistances ont été répertoriées comprenant :

- Les fautes de saisie.
- Les formats différents dans le même attribut (colonne).
- Le manque d'information (texte manquant).
- Les valeurs nulles.
- L'incompatibilité entre la valeur et la description de l'attribut.
- Les redondances : duplication de l'information.
- La persistance de données obsolètes.
- La confrontation de données sémantiquement équivalentes mais syntaxiquement différentes.

3.1.2 Tâches de nettoyage :

- **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.

- **La conversion :**

Elle sert à convertir les données provenant des sources dans un format cible approprié.

- **Les dictionnaires de synonymes ou d'abréviations :**

Il est possible également de réaliser des analyses lexicales des champs sources. Cela permet de comprendre les champs qui signifient la même chose. Par exemple, les champs suivants signifient la même valeur « Alg », « Algr », « Alger ». Ainsi, on peut définir des tables de règles (abréviations, synonymes) afin de pouvoir remplacer les valeurs sources par des valeurs cibles selon les besoins. Exemple : on utilise l'abréviation « Mr » pour désigner « Monsieur », « 1 » pour « Vrai » et « 0 » pour « Faux » respectivement.

3.2 La transformation des données:

Basée sur une série de règles et de fonctions, l'étape de transformation s'applique aux données extraites de la source afin de dériver les données pour le chargement. Certaines sources de données ne nécessitent qu'une manipulation très légère voire aucune. Dans d'autres cas, un ou plusieurs types de transformations peuvent être nécessaires pour répondre aux besoins de la base de données cible.

La phase de transformation consiste à supprimer les incohérences sémantiques entre les sources pouvant survenir lors de l'intégration. Elle regroupe les opérations de mise en format nécessaires aux données, de calcul des données secondaires et de fusion ou d'éclatement des informations composites. Comme elle peut comprendre aussi une phase d'agrégation des données.

3.2.1 Quelques types d'incohérences sémantiques :

- **Incohérences des schémas :**

- *Problèmes de modélisation* : différents modèles de données sont utilisés.
- *Problèmes de terminologie* : un objet est désigné par deux ou plusieurs noms différents. En revanche, un même nom peut désigner deux ou plusieurs objets différents.

- *Conflits sémantiques* : choix de différents niveaux d'abstraction pour un même concept.
- *Conflits de structure* : choix de différentes propriétés pour un même concept.
- *Conflits de représentation* : deux représentations différentes choisies pour les mêmes propriétés d'un même objet.

3.2.2 Tâches de transformation :

- **Sélection** : Sélectionner uniquement les données nécessaires.
- **Jointure :** Joindre les données provenant de sources multiples.
- **Eclatement :**

Fractionnement d'une colonne en plusieurs colonnes (ex. éclater une colonne qui contient une chaine de valeurs séparées par des virgules spécifiées sous forme de valeurs individuelles dans différentes colonnes).

- **Traduction des valeurs codées.**
- **Agrégation :**

L'agrégation (par exemple, cumul - résumant plusieurs lignes de données - total des ventes pour chaque magasin, et pour chaque région, etc.). Le niveau d'agrégation est choisi au moment de la construction de l'ED et les données initiales seront perdues. C'est effectivement ce qui permet d'avoir des temps de réponse très courts.

- **Génération de clé de substitution :**

La clé de substitution (SK) est utilisée afin de substituer la clé naturelle (NK) qui provient des systèmes opérationnels pour rendre un élément unique dans la dimension. Ce n'est plus la NK qui sera utilisée pour faire les jointures avec les faits ou les autres dimensions.

Avantages :

- *Performance :* Accélère l'accès aux données du moment où l'on va utiliser un index numérique (le type de données de SK est numérique).
- *Indépendance du système source :* On ne peut garantir que la clé naturelle ne change pas dans les systèmes sources.
- *Historique des changements:* Si l'on désire garder l'historique des changements de la dimension selon certains critères, on doit gérer la SK. On se retrouve facilement avec plusieurs enregistrements de la même NK dans la dimension.

- **Transposition :** pivotant ou tournant sur plusieurs colonnes en plusieurs lignes ou vice versa.
- **Désagrégation :** éliminer la répétition des colonnes dans une table séparée (ex. le déplacement d'une série d'adresses dans une colonne en adresses individuelles dans plusieurs colonnes dans une autre table d'adresses liées).
- **Validation :**

Rechercher et valider les données pertinentes à partir des tables ou des fichiers référentiels pour les dimensions à variation lente.

4. Chargement de données

Le chargement est la dernière phase d'alimentation de l'ED. C'est une phase délicate notamment lorsque les volumes sont importants. Pour obtenir de bonnes performances en chargement, il est impératif de maîtriser les structures du SGBD (tables et index) associées aux données à charger afin d'optimiser au mieux ces processus. Les techniques de parallélisation optimisent les chargements lourds. Pour les mettre en œuvre, des utilitaires particuliers existent chez la majorité des éditeurs de bases de données.

L'opération principale dans le processus est le transfert des tables de dimensions puis les tables de faits de manière efficace.

4.1 Chargement des tables de dimensions :

Les dimensions fournissent le contexte pour les tables de faits et donc pour toutes les mesures présentées dans l'ED. Bien que les tables de dimensions soient généralement beaucoup plus petites que les tables de faits, elles représentent le cœur de l'ED, car elles fournissent des points d'entrée des données.

4.1.1 Les données dimensionnelles :

Toutes les dimensions doivent être physiquement construites pour avoir le minimum de composants. On appelle une clé de substitution (SK) la clé primaire de la dimension contenant une valeur unique. Le processus ETL de l'entrepôt de données

doit toujours créer et insérer les clés de substitutions. Elles ne sont jamais assignées par d'autres entités.

La SK d'une dimension est utilisée pour opérer des jointures à des tables de faits. 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. Ces dernières sont beaucoup plus compactes lorsque les clés étrangères sont de type entier.

Quelques dimensions sont créées par le système ETL et n'ont pas une véritable source extérieure. Celles-ci sont généralement de petites dimensions de consultation (lookup) qui sont créées directement comme des tables relationnelles dans leur forme finale. Mais le cas important est la dimension extraite d'une ou plusieurs sources extérieures.

Les données dimensionnelles sont, en général, complexes telles que les tables CLIENT, PRODUIT, qui sont alimentées à partir de plusieurs sources et à des moments différents. Cela nécessite la résolution des conflits et l'application des mises à jour aux entités et ce, à divers points.

4.1.2 Etapes de chargement:

- Le nettoyage des données comporte toutes les opérations nécessaires pour nettoyer, valider les données alimentant une dimension et les rendre cohérentes. Pour certaines dimensions plus simples, ce module peut être presque inexistant. Mais pour les grandes dimensions importantes comme client ou produit, le module de nettoyage est un système très important avec de nombreux sous-composants, dont la validation des attributs et la vérification des valeurs.

- La normalisation de données se compose de toutes les étapes nécessaires pour aligner le contenu de certains ou de tous les champs de la dimension. Par exemple, si nous avons des tables de faits décrivant les opérations de facturation et des appels de soutien aux clients, elles sont probablement associées à une dimension CLIENT. Dans les grandes entreprises, les sources pour ces deux dimensions relatives aux clients pourraient être très différentes. Au pire des cas, il pourrait ne pas y avoir de cohérence garantie entre les champs des deux tables de dimension. Dans les cas où l'entreprise est contrainte à faire face aux combinaisons d'informations de sources

multiples ; telle que la facturation et le support client, l'étape de normalisation est nécessaire pour qu'une partie ou la totalité des champs dans les deux dimensions puissent partager les mêmes domaines de valeurs. Une fois l'étape de normalisation ait modifié plusieurs attributs descriptifs importants dans la dimension, les données conformes sont alignées à nouveau.

- Enfin, le module de chargement de données se compose de toutes les étapes nécessaires à l'administration du Slowly Chaging Dimensions (SCD, décrite dans le chapitre suivant) et écrire les dimensions comme des tables physiques dans un format dimensionnel adéquat avec des clés primaires et naturelles correctes ainsi que des attributs descriptifs finaux. La création et l'affectation des clés de substitution (SK) interviennent dans ce module. [Kimball, 2004]

4.2 Chargement de tables de faits :

Dans la modélisation dimensionnelle, nous construisons délibérément nos bases de données autour des mesures numériques de l'entreprise. Les tables de faits contiennent des mesures et les tables de dimensions contiennent le contexte des mesures environnantes.

Cette vision simple du monde a prouvé encore une fois à être intuitive et compréhensible envers les utilisateurs finaux de nos entrepôts de données. Voilà pourquoi nous alimentons les entrepôts à travers les modèles dimensionnelles. [Kimball, 2004]

4.2.1 L'intégrité référentielle :

Dans la modélisation dimensionnelle, l'intégrité référentielle signifie que chaque table de fait est remplie de clés étrangères légitimes. D'une autre manière, un tuple de table de fait ne doit pas contenir des références de clés étrangères corrompues ou inconnues.

Il n'y a que deux façons pour violer l'intégrité référentielle dans un schéma dimensionnel :

- Charger un tuple de la table de fait avec une ou plusieurs mauvaises clés étrangères.
- Supprimer un tuple de dimension dont la clé primaire est utilisée dans la table de fait.

Un tuple d'une table de fait qui viole l'intégrité référentielle (parce qu'il contient une ou plusieurs mauvaises clés étrangères) n'est pas seulement une nuisance mais peut être aussi dangereux. Vraisemblablement, le tuple a une certaine légitimité car il représente, sans doute, un véritable évènement de mesure, mais il est stocké dans la base de données de façon incorrecte. Toute requête qui fait référence à la mauvaise dimension dans le tuple de la table de fait ne parviendra pas à inclure ce tuple. Par définition, la jointure entre la dimension et ce tuple ne peut pas avoir lieu.

L'étape précédant le chargement de la table de fait consiste à rechercher des clés naturelles (NK) dans le tuple de la table de fait et les remplacer par les bonnes valeurs des SK de dimension. Ce processus est expliqué en détail dans le chapitre suivant et porte sur le nom de Surrogate Key Pipeline (SKP). Le cœur de cette procédure est une table lookup contenant le couple (NK, SK). Si cette table est correctement entretenue, les tuples de la table de fait obéiront à l'intégrité référentielle. De même, pour pouvoir supprimer un tuple de la table de dimension, une requête doit être faite afin de joindre le tuple à la table de fait. Si la requête renvoie la valeur « null », le tuple est supprimé. [Kimball, 2004].

5. Métadonnées

Tout système d'information utilise non seulement des données, mais aussi des *données sur les données*. Ces dernières, désignées par le terme générique de métadonnées, ont deux rôles distincts :

- La définition sémantique des données dans des termes appropriés à la vision conceptuelle qu'en a l'utilisateur.
- La description des structures techniques dans lesquelles elles sont enregistrées et des chemins qui permettent d'y accéder.

Dans un environnement décisionnel, l'utilisateur a besoin d'une carte et un manuel de navigation. En d'autres termes, il lui faut une documentation *informationnelle* complète sur les données. Pour cela, on utilise souvent les métadonnées pour stocker les données qui décrivent les caractéristiques de l'ED, sa création, sa gestion et son usage. Mais, il est possible aussi de se servir des métadonnées pour décrire les informations relatives au processus d'alimentation de l'entrepôt, les sources dont on va extraire les données, les fonctions à appliquer afin de les transformer et la cible à charger ainsi que les informations administratives relatives (date d'extraction, les bases temporaires à utiliser, l'utilisateur, etc.).

5.1 Les types des métadonnées :

Il existe deux types de métadonnées :

5.1.1 Les métadonnées conceptuelles :

Elles décrivent la signification informationnelle des données et s'appliquent aux Modèles Conceptuels de Données.

Les métadonnées conceptuelles dans la plupart des applications de gestion sont rares, absentes ou même erronées. Ceci est dû à la manière dont le système est conçu. Malgré l'influence des méthodes de conception destinées à permettre aux programmes de fonctionner, des défaillances conceptuelles affectant les données peuvent se présenter. Ces défaillances vont se traduire par des coûts de maintenance importants qui ne sont pas pris en compte par les développeurs dont la préoccupation est focalisée beaucoup plus sur l'aspect technique et sur la réponse des pressions à court terme.

5.1.2 Les métadonnées logiques et physiques:

Elles décrivent les modalités physiques de stockage et les chemins d'accès et aux Modèles Logiques et Physiques.

Les métadonnées relatives à l'implémentation physique sont complètes et précises, parce qu'elles seules sont imposées par la technique. Les métadonnées de cette catégorie revêtent des formes extrêmement variées et hétérogènes. Ce sont notamment :

- Des descriptions de formats d'enregistrement dans le code des programmes d'application.
- Des descriptions de structures de fichiers dans les registres de contrôle des systèmes d'exploitation.
- Des catalogues produits et utilisés par les systèmes de gestion de bases de données (SGBD), ainsi que des scripts ayant servi à créer les modèles physiques.

6. Conclusion

Dans ce chapitre, nous avons décrit les étapes du flux de données ETL en détail. Ainsi que quelques pensées relatives aux dimensions et aux faits plus particulièrement. Nous verrons par la suite quelques fonctionnalités, les plus utilisées dans les processus d'intégration de données.

PARTIE 2

CHAPITRE IV

DEFINITION D'UN PROCESSUS ETL SELON SES FONCTIONNALITES DE BASE

1. Introduction

Aujourd'hui, la plupart des grandes entreprises disposent d'un ED capable d'analyser l'information depuis des systèmes de production sur une base de données quotidiennes ou hebdomadaires. Cependant, la plupart jugent le processus ETL trop long et complexe, vu qu'il se fait sur des bases de données critiques et importantes. En même temps, le besoin d'accéder à l'information à jour a augmenté au point que les mises à jour quotidiennes ou hebdomadaires ne suffisent plus.

Dans ce chapitre, nous présentons une approche permettant une meilleure maitrise du processus ETL, une robustesse, une meilleure performance et une facilité de maintenance.

2. Fonctionnalité d'ETL :

Une fonctionnalité d'ETL est une fonction de base qui prend en charge un traitement particulier dans un processus ETL telles que Changing Data Capture (CDC), Surrogate Key (SK), Slowly Changing Dimension (SCD), Surrogate Key Pipeline (SKP). La fonctionnalité CDC par exemple permet dans la phase d'extraction d'un processus ETL d'identifier, dans les systèmes sources, les tuples affectés par les changements afin de les capturer et les considérer dans le rafraichissement de l'ED.

3. Fonctions courantes :

3.1 Changing Data Capture « CDC » :

La phase d'extraction d'un processus ETL commence par identifier les données affectées par des changements (insertions, modifications, suppressions) pour les extraire et les stocker dans la zone de préparation (DSA). Ces données seront traitées ultérieurement dans la phase de transformation. Dans presque tous les ED, nous devons transférer uniquement les changements pertinents ayant affecté les données sources depuis le dernier transfert. Le rafraichissement complet de nos tables cibles (faits et dimensions) est généralement indésirable.

Capturer les données sources récentes est appelée « Changing Data Capture», CDC en abrégé. L'idée derrière la capture de données modifiées semble assez simple: transférer uniquement les données qui ont été modifiées depuis le dernier chargement. Mais la construction d'un bon système de capture de changement de données n'est pas aussi facile tel qu'il paraît.

En général, Le CDC est une fonctionnalité qui permet de suivre et de capturer les différentes modifications ayant eu lieu sur une table. Les modifications capturées par le « CDC » sont toutes celles effectuées sur la table via les instructions d'insertion (INSERT), de mise à jour (UPDATE) ou de suppression (DELETE).

3.1.1 Objectifs :

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.

Voici, plus précisément, les objectifs de CDC:

- Isoler les données sources modifiées pour permettre le traitement sélectif plutôt que de rafraîchissement complet.
- Capturer tous les changements (suppressions, modifications et insertions) apportées à la source de données.
- Suivi de la conformité avec des métadonnées supplémentaires.
- Effectuer l'étape de capture de données modifiées le plus tôt possible, de préférence, avant le transfert de données en grands volumes à l'entrepôt de données.

3.1.2 Méthodes de capture :

Kimball [Kimball, 2004] a décrit les quatre principales façons de détection des changements:

- **Les colonnes de vérification :**

Dans la plupart des cas, le système source contient des colonnes d'audit. Les colonnes d'audit sont ajoutées à la fin de chaque table pour stocker la date et l'heure où un tuple a été ajouté ou modifié. Les colonnes de vérification sont généralement peuplées par des déclencheurs de base de données qui sont déclenchés automatiquement quand des tuples sont insérées ou mises à jour. Parfois, pour des raisons de performance, les colonnes sont peuplées par d'autres moyens que les déclencheurs tels que des applications front-end. Dans ces cas, il faut analyser et tester chacune des colonnes pour indiquer les données modifiées. Si des valeurs NULL existent, il faut trouver une autre approche pour détecter les changements.

- **Capture des changements à partir des journaux de transactions des bases de données :**

Consiste à prendre une capture de la base de données à un certain temps et la parcourir pour les transactions qui affectent les tables participant au processus ETL.

- **Extraits chronométrés :**

Sélectionner généralement toutes les lignes où la date ou dans les champs de la date de création ou modification est égale à SYSDATE-1, ce qui signifie qu'on aura tous les tuples du jour d'avant. Le chargement des tuples, basé uniquement sur le temps est une erreur commune faite par la plupart des développeurs ETL débutants. Ce processus est horriblement peu fiable. La sélection de données basée sur le temps charge des tuples dupliqués quand elle est redémarrée à partir du début en cas d'échec. Cela signifie que le nettoyage manuel des données est nécessaire si le processus échoue pour une raison quelconque.

- **Base de données complète « comparaison des différences » :**

Cette technique assure 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) pour détecter les changements. En outre, l'utilisation des algorithmes CRC (contrôle de redondance cyclique) peut déterminer rapidement si un tuple complexe a changé. C'est cette approche que nous retenons pour l'implémentation de la fonctionnalité CDC.

3.1.3 Exemple :

Prenons l'exemple d'une table source PRODUIT caractérisé par son identificateur (Id_produit), son nom (Nom_produit) et sa famille (Id_famille) .Grace à une version précédente de celle-ci (snapshot), nous pourrons identifier les tuples ayant été affecté par des mises à jour (insertion, modification, suppression) :

- Table source (version actuelle):

Id_produit	Nom_produit	Id_famille
P01	HP	1
P03	Canon	2
P04	ASUS	3
P05	Toshiba	1

- Snapshot (version précédente de la table):

Id_produit	Nom_produit	Id_famille
P01	HP	1
P02	Samsung	4
P03	Canon PowerShot	2
P04	ASUS	3

- Les tuples identifiés (ayant été affectés par des mises à jour) :

Id_produit	Nom_produit	Id_famille	Résultat de CDC
P02	Samsung	4	**suppression**
P03	Canon PowerShot	2	**modification**
P05	Toshiba	1	**insertion**

3.1.4 Algorithme:

La technique présentée dans cet algorithme est celle basée sur les snapshots:

- La table source notée TS contient des données dans leur version récente.
- La table du DSA notée TSvp contient les mêmes données dans leur version précédente.
- La partie (1) charge TS dans DSA et trie TS et TSvp selon la clé « Clé » afin de détecter, dans la partie (2) les insertions (TSvp.Clé > TS.Clé), suppressions (TSvp.Clé < TS.Clé) et modifications (TSvp.Clé = TS.Clé avec une différence de au moins un attribut). Les autres cas seront rejetés (copie similaire du tuple dans TS et TSvp) puisqu'aucun changement n'a eu lieu depuis le dernier rafraichissement.
- Pour détecter les cas de modifications, nous appliquons une fonction de hachage connue sous le nom de CRC (Cyclic Redundancy Check) sur les tuples de TS et de TSvp ayant la même valeur de « Clé ».

```
CDC SNAPSHOTS
Entrées :
TS : Version récente de la table source,
TSvp : Table DSA qui contient la version précédente de la table source,
tuple1 : Enregistrement de type TS,
tuple2 : Enregistrement de type TSvp,
Sorties :
L'ensemble des tuples affectés par insertion, modification ou suppression
Début
1 { TS(DSA) ← TS(Système source)
  { Trier TS et TSvp par ordre croissant sur Clé
  { Lire(TS, tuple1)
  { Lire(TSvp, tuple2)
  { Tant que (non fin(TS) et non fin(TSvp)) faire
  {     Si (tuple1.Clé = tuple2.Clé) Alors
2 {         Si (CRC(tuple1) ≠ CRC(tuple2))Alors
  {             Capturer tuple1 comme modification
  {         Fin Si
  {         Lire(TS, tuple1)
  {         Lire(TSvp, tuple2)
  {     Sinon
  {         Si (tuple1.Clé > tuple2.Clé) Alors
  {             Capturer tuple2 comme suppression
  {             Lire(TSvp, tuple2)
  {         Sinon
  {             Capturer tuple1 comme insertion
  {             Lire(TS, tuple1)
  {         Fin Si
  {     Fin Si
  { Fait
  { Tant que (non fin(TS)) faire
3 {     Capturer tuple1 comme insertion
  {     Lire(TS, tuple1)
  { Fait
  { Tant que (non fin(TSvp)) faire
  {     Capturer tuple2 comme suppression
  {     Lire(TSvp, tuple2)
  { Fait
Fin.
```

Algorithme 1: Changing Data Capture (CDC)

3.2 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. Une table de dimension contient des attributs qui décrivent la sémantique et le contexte du contenu de la table de fait. Dans ce sens, la table de dimension mérite une importance particulière lors des mises à jour de façon à préserver la sémantique de ses membres. Lors du chargement des données issues des systèmes sources vers une table de dimension, des approches de versionnement s'appliquant sur les membres de la dimension existent. Elles ont pour rôle 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).

3.2.1 Les types SCD:

Nous distinguons différentes approches de versionnement relatives à la mise à jour des membres d'une dimension. Selon ces approches, un attribut d'une dimension pourra ne jamais être concerné par une mise à jour comme il pourra être concerné mais aucune trace ne sera conservée sur ses anciennes valeurs. Dans d'autres approches, un attribut est concerné par des mises à jour mais un archivage de ses anciennes valeurs est nécessaire d'où une gestion des versions de l'attribut (versionnement). Nous décrivons dans ce qui suit, les approches les plus pertinentes que nous avons retenu.

- **SCD type 0:**

La valeur de l'attribut de dimension ne change jamais, donc les faits sont toujours regroupés par cette valeur d'origine. Le type 0 est approprié pour tout attribut marqué «original», tel le score de crédit d'origine d'un client ou d'un identifiant durable.

- **SCD type 1:**

L'ancienne valeur de l'attribut dans la ligne de la dimension est remplacé par la nouvelle valeur lors la mise à jour. Les attributs de type 1 reflètent toujours l'affectation la plus récente, et donc cette technique ne nécessite 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:**

Le changement de type 2 ajoute un nouveau tuple dans la dimension avec les valeurs d'attributs mises à jour. Cela nécessite de généraliser la clé primaire de la dimension au-delà de la clé naturelle car il y aura potentiellement plusieurs lignes décrivant chaque membre. Quand une nouvelle ligne est créée pour un membre de dimension, une nouvelle clé de substitution (SK) est attribuée et utilisée comme une clé étrangère dans toutes les tables de fait lors la mise à jour jusqu'à ce qu'un changement ultérieur crée une nouvelle clé de dimension et mis à jour la ligne de la dimension. Dans ce cas, la table de dimension contient, au fur et à mesure des mises à jour sur des attributs SCD type 2, des tuples avec leurs différentes versions, ce qui augmente le volume de la table de manière importante surtout si celle-ci est une grande dimension (big dim) comme le cas des dimensions clients, produits, ...

- **SCD type 4 :**

Le principe de ce type de versionnement est le même que celui de type 2 puisqu'il nécessite l'historisation. La seule différence est au lieu de préserver les anciennes versions des tuples dans la table de base, celles-ci sont transférées dans une mimi-dimension.

3.2.2 Exemple :

Prenons l'exemple d'une dimension PRODUIT :

- **SCD type 1 :**

Source :

Avant

Id_produit	Nom _produit	Id_famille
P401	HP	1

Après

Id_produit	Nom _produit	Id_famille
P401	**HP Pavilion**	**1**

Dimension :

SK_P	Id_produit	Nom _produit	Id_famille
SP1	401	**HP Pavilion**	**1**

- **SCD type 2 :**

Source :

Avant

Id_produit	Nom _produit	Id_famille
P401	HP	1

Après

Id_produit	Nom _produit	Id_famille
P401	**HP Pavilion**	**1**

Dimension :

SK_P	Id_produit	Nom _produit	Id_famille
SP1	P401	HP	**1**
SP2	P401	**HP Pavilion**	**1**

- **SCD type 4 :**

Source :

Avant

Id_produit	Nom _produit	Id_famille
P401	HP	1

Après

Id_produit	Nom _produit	Id_famille
P401	**HP Pavilion**	**1**

Dimension :

SK_P	Id_produit	Nom _produit	Id_famille
SP2	P401	**HP Pavilion**	**1**

Table d'historisation :

SK_P	Id_produit	Nom_produit	Id_famille
SP1	P401	HP	**1**

3.2.3 Algorithme :

Nous proposons les algorithmes 2 et 3 qui décrivent respectivement :

- Le fonctionnement de SCD pour les stratégies SCD type 1 et SCD type 4.
- La fonction SK() de génération d'une nouvelle valeur pour la clé de substitution.

Pour pouvoir traiter les approches SCD sur les tables de dimension d'un ED, le processus SCD doit disposer de métadonnées sur les attributs de la dimension pouvant être concernés par les mises à jour ainsi que l'approche SCD correspondante. Pour se faire, nous proposons une table nommée TSCD dont la structure est décrite par le tableau 5.

```
SCD
Entrées :
TS : Table source dans DSA après transformation,
TD : table de dimension,
TDH : Table Historique de TD,
VT1, VT2, VT3 : enregistrements de type respectivement TS, TD et TD ,
VSK : Integer,
VTYPE, VT : entier,
début
    VTYPE ← 1
    OUVRIR (TS) ;
    tantque non fin (TS) faire
        LIRE (TS, VT1) ;
        si EXISTS (SELECT * FROM TD WHERE TD.NK=VT1.NK) alors
            VT2 ← SELECT * FROM TD WHERE (TD.NK=VT1.NK)
            VT3 ← VT2 // VT3 est une variable de travail
            Pour CHAQUE attribut de VT1 et VT2 Faire
                si VT1.Attribut ≠ VT2.Attribut alors
                    VT ← SELECT TypeSCD FROM TSCD WHERE SCD.Dimension='TD'
                    and SCD.attribut=VT2.attribut
                    si VT = 1 alors
                        UPDATE TD SET TD.attribut=VT1.attribut where TD.NK=VT1.NK ;
                        UPDATE TDH SET TDH.attribut=VT1.attribut where
                        TDH.NK=VT1.NK ;
                        sinon
                            VTYPE ← VT ; //VTYPE contiendra le plus grand type SCD
                        fin
                    fin
                    VT3.attribut ← VT1.attribut ;
                fin
                suivant (VT1, attribut) ; suivant (VT2, attribut) ;
            FinPour
            si VTYPE=4 alors
                INSERT (TDH, VT2) ; // insérer la version précédente du tuple dans TDH
                DELETE (TD, VT2) ; // supprimer la version précédente du tuple à partir de TD
                VT3.SK ← SK (TD) ; INSERT (TD, VT3) ; // insérer la version récente du tuple dans TD
            fin
            sinon
                VT3 ← VT1 ; VT3.SK ← SK (TD) ; INSERT (TD, VT3) ; // Il s'agit d'un nouveau
                membre à insérer dans TD
            fin
        fin
    fait
fin
```

Algorithme 2: Slowly Changing Dimension (SCD)

SK
Entrées :
TD : table de dimension
Sortie : $SK()$: nouvelle valeur de SK pour le tuple à insérer dans TD
début
SK ← SELECT MAX(SK) FROM TD ;
SK ← SK+1 ;
Return (SK) ;
fin

Algorithme 3 : Surrogate Key (SK)

Exemple : table TSDC, approches SCD appliquées aux attributs des dimensions Produit et Région.

Dimension	Attribut	TypeSCD
Région	Nom_région	4
Produit	Nom_produit	4
Produit	Designation_produit	1
Produit	Famille	4

Tableau 5: table TSDC

3.3 Surrogate Key Pipeline « SKP »:

Surrogate Key Pipeline (SKP) est l'une des fonctionnalités de base d'un processus ETL. A la fin de la phase de transformation du processus ETL, le chargement des données dans les tables de dimension précède celui des tables de fait. La différence entre les deux est que le chargement dans la table de fait doit respecter l'intégrité référentielle envers les dimensions associées. Pour ce faire, il faut remplacer les clés naturelles (NK) entrant dans la ligne de la table de faits avec les clés de substitution (SK) appropriées. L'appellation SK Pipeline vient du fait que les tuples sources à destination de la table de fait passent tous par un processus consistant, pour chacun et pour chacune de ses NK, de rechercher la clef SK correspondante et de remplacer NK par SK. 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. Le tuple ayant subi un remplacement de toutes ses NK pourra être inséré dans la table de fait.

3.3.1 Le processus SK (Surrogate Key):

L'approche directe de la recherche de la valeur la plus récente de la clé de substitution (SK) correspondant à chaque clé naturelle (NK) est d'utiliser une table lookup générée à partir de la dimension contenant toutes les valeurs de NK avec les valeurs SK les plus récentes. Pour obtenir la clé de substitution (SK) actuelle, il suffit d'identifier la clé naturelle (NK) dans la table lookup, puis sélectionnez la clé de substitution (SK) correspondante. Les environnements matériels actuels offrent presque une mémoire adressable illimitée, ce qui rend cette approche pratique.

3.3.2 Exemple:

Prenons un exemple traitant sur le chiffre d'affaire CA réalisé par Produit, Région et Temps (voir chapitre I - section 3.2). La figure 10 illustre le schéma en étoile approprié.

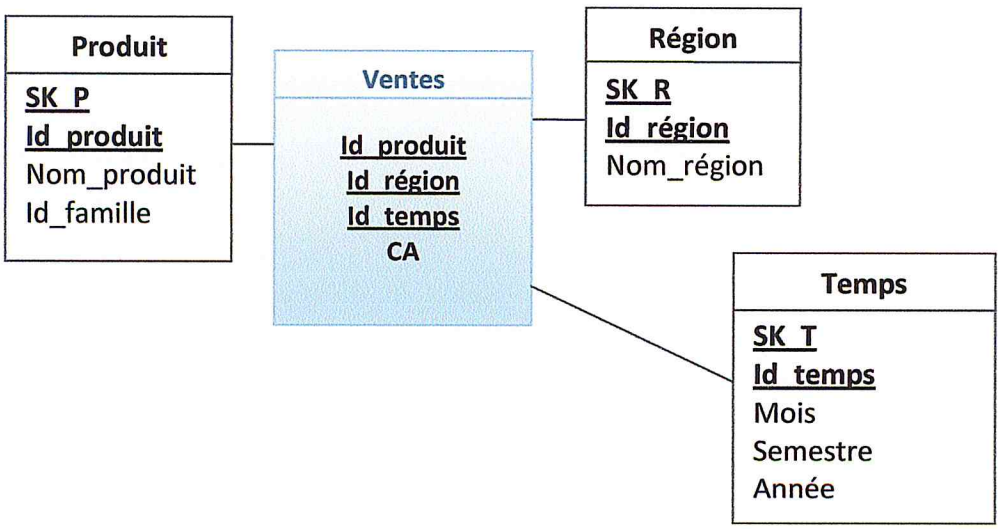
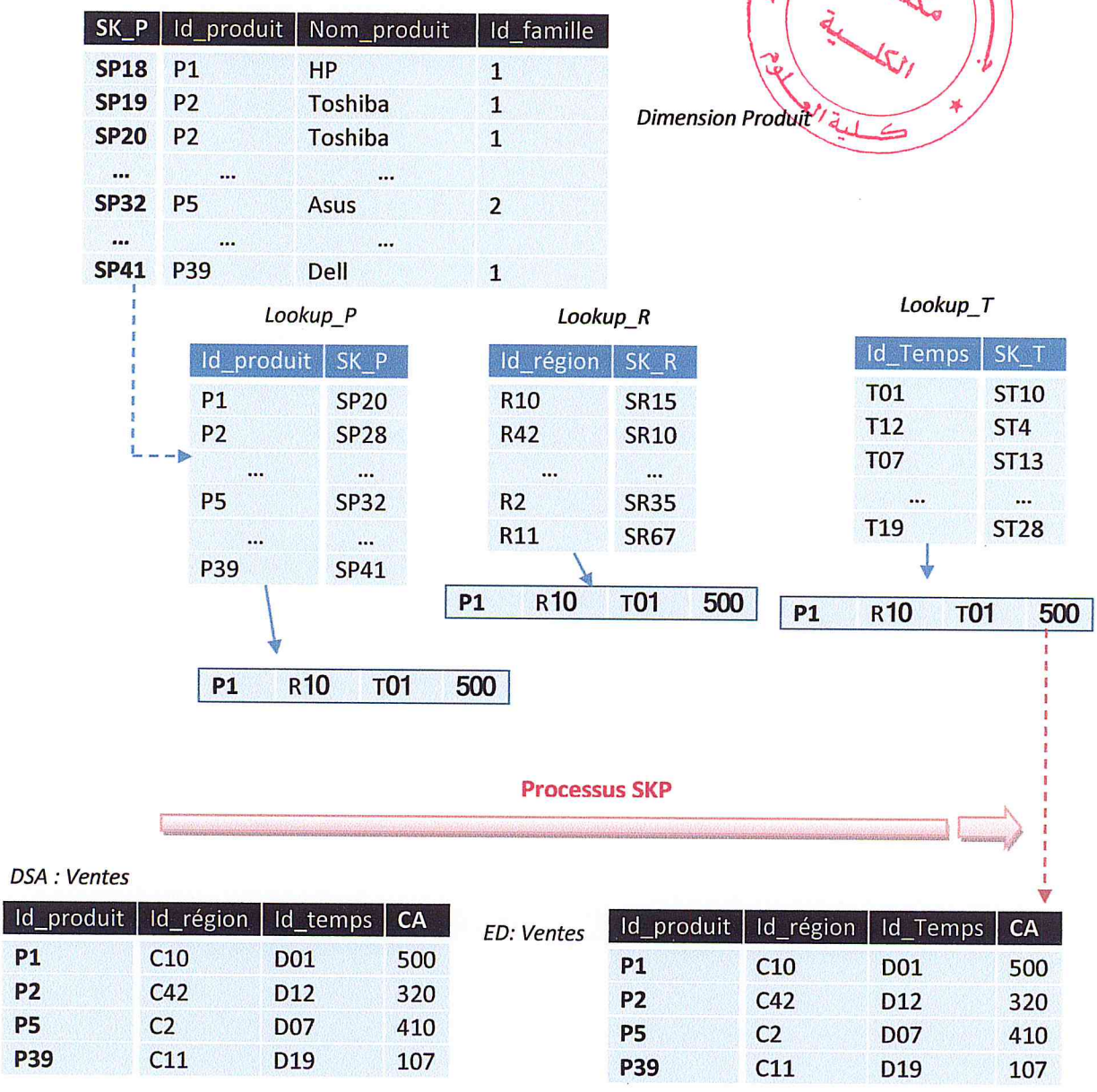


Figure 10 : Schéma en étoile pour l'analyse du chiffre d'affaire

Nous présentons dans la figure 11 le traitement d'un ensemble de tuples (Id_produit, Id_région, id_Temps, CA) de la table source Ventes par le processus SKP.



SK_P	Id_produit	Nom_produit	Id_famille
SP18	P1	HP	1
SP19	P2	Toshiba	1
SP20	P2	Toshiba	1
...	...	...	
SP32	P5	Asus	2
...	...	...	
SP41	P39	Dell	1

Id_produit	SK_P
P1	SP20
P2	SP28
...	...
P5	SP32
...	...
P39	SP41

Id_région	SK_R
R10	SR15
R42	SR10
...	...
R2	SR35
R11	SR67

Id_Temps	SK_T
T01	ST10
T12	ST4
T07	ST13
...	...
T19	ST28

DSA : Ventes

Id_produit	Id_région	Id_temps	CA
P1	C10	D01	500
P2	C42	D12	320
P5	C2	D07	410
P39	C11	D19	107

ED: Ventes

Id_produit	Id_région	Id_Temps	CA
P1	C10	D01	500
P2	C42	D12	320
P5	C2	D07	410
P39	C11	D19	107

Figure 11: Exemple du processus SKP pour le traitement des tuples de Ventes

3.3.3 Algorithme:

L'algorithme 4 décrit le programme principal de SKP. Il consiste à :

- Charger les tables TLookup1, TLookup2, TLookup3 et TLookup4 en mémoire cache pour un accès rapide.
- Lire à partir de la table TFNK en bloc pour accélérer le processus SKP.
- Déclencher le processus SKP en appelant la première procédure Lookup1 pour le remplacement de NK1 par SK1.

```
Entrées : TFNK : Table de fait avec les clefs NK.
début
    Variables VT1, VT2, VT3, VT4 : enregistrements de type TFNK
    // charger les tables d'index Lookup en cache pour accélérer SKP
    mise en cache de TLookup1, TLookup2, TLookup3 et Tlookup4
    Ouvrir (TFNK) ;
    tantque non fin (TFNK) faire
        // charger les tuples de TFNK par bloc dans buffer pour accélérer SKP
        Buffer ← bloc(TFNK) ; // charger dans buffer un nombre de tuples multiple de 4
        tantque non fin (Buffer) faire
            Lire (Buffer, VT1) ;
            Lire (Buffer, VT2) ;
            Lire (Buffer, VT3) ;
            Lire (Buffer, VT4) ;
            Lookup1 (VT1) ;
            Lookup1 (VT2) ;
            Lookup1 (VT3) ;
            Lookup1 (VT4) ;
        fait
    fait
fin
```

Algorithme 4: Surrogate Key Pipeline (SKP)

L'algorithme 5 décrit la première étape du processus SKP. Il consiste, pour un tuple soumis par le programme principal, de remplacer la valeur de sa clé NK1 par la valeur de SK1, récupérée à partir de la table d'index Tlookup1. Après remplacement, le tuple sera soumis à Lookup2 qui est la deuxième étape du processus SKP.

Lorsqu'un tuple parcourra les quatres étapes Lookup1 (algorithme 5), Lookup2 (algorithme 6), Lookup3 (algorithme 7) et Lookup4 (algorithme 8), ses quatre clés NK1, NK2, NK3 et NK4 auront été remplacées, respectivement, par les valeurs de SK1, SK2, SK3 et SK4. Ainsi, le tuple est prêt pour être inséré dans la table de fait au niveau du DW.

Entrées : $TLookup1$: table $Lookup$ correspondant à $dimension_1$
VT : Tuple de la table $TFNK$
début
| $VT.NK1 \leftarrow$ SELECT $SK1$ FROM $TLookup1$ WHERE $TLookup1.NK1$=$VT.NK1$;
| Lookup2 (VT) ;
fin

Algorithme 5: Lookup1 (VT)

Entrées : $TLookup2$: table $Lookup$ correspondant à $dimension_2$
VT : Tuple de la table $TFNK$ dont seule $NK1$ est remplacée par $SK1$
début
| $VT.NK2 \leftarrow$ SELECT $SK2$ FROM $TLookup2$ WHERE $TLookup2.NK2$=$VT.NK2$;
| Lookup3 (VT) ;
fin

Algorithme 6: Lookup2 (VT)

Entrées : $TLookup3$: table $Lookup$ correspondant à $dimension_3$
VT : Tuple de la table $TFNK$ dont $NK1$ et $NK2$ ont été remplacées par $SK1$ et $SK2$
début
| $VT.NK3 \leftarrow$ SELECT $SK3$ FROM $TLookup3$ WHERE $TLookup3.NK3$=$VT.NK3$;
| Lookup4 (VT) ;
fin

Algorithme 7: Lookup3 (VT)

Entrées : $TLookup4$: table $Lookup$ correspondant à $dimension_4$
VT : Tuple de $TFNK$ dont $NK1$, $NK2$ et $NK3$ ont été remplacées par $SK1$, $SK2$ et $SK3$
Sortie : $TFSK$: table de fait avec SK
début
| $VT.NK4 \leftarrow$ SELECT $SK4$ FROM $TLookup4$ WHERE $TLookup4.NK4$=$VT.NK4$;
| Ecrire ($TFSK$, VT) ; // insérer le tuple dans la table de fait avec SK ($TFSK$)
fin

Algorithme 8: Lookup4 (VT)

Dans un environnement classique (sur une seule machine), SKP fonctionne avec un parallélisme qui permet de traiter plusieurs tuples en même temps. Le nombre de tuples pouvant être traité en même temps est égal au nombre des clefs NK dans chaque tuple (égal au nombre de SK et également au nombre de tables Lookup). Pour assurer ce principe, la machine ainsi que l'application SKP doivent être multithreads, surtout si le nombre de tables Lookup est important.

4. Conclusion

Répondant à la complexité des traitements dans un processus ETL, nous avons présenté, dans ce chapitre, les fonctionnalités d'ETL les plus courantes telles que Changing Data Capture (CDC), Slowly Changing Dimension (SCD) et Surrogate Key Pipeline (SKP) avec des exemples illustrant chaque fonction.

L'étape suivante est la spécification des besoins et la conception de notre sytème.

PARTIE 3

CHAPITRE V

SPECIFICATION DES BESOINS ET CONCEPTION DU SYSTEME

1. Introduction

Afin de garantir le développement efficace des applications de qualité, les bonnes pratiques doivent toujours être adoptées. Le processus unifié semble être la solution idéale pour les développeurs. Il regroupe les activités à mener pour transformer **les besoins** d'un utilisateur en **un système logiciel** quelque soit la complexité du projet, la taille et le domaine d'application du système.

Pour le développement de notre système, nous nous sommes basés sur le processus unifié qui utilise le langage UML comme langage de modélisation. Dans une première phase nous avons effectué la spécification des besoins, suivie de celle portant sur la conception et enfin la dernière phase consacrée à la réalisation du système suivie par les tests de l'application.

Dans ce chapitre, nous présenterons la spécification et l'analyse des besoins ainsi que la conception de l'application. La réalisation sera présentée dans le chapitre suivant.

2. Langage de modélisation UML

UML qui est l'acronyme d'*Unified Modeling Language* est aujourd'hui indissociable de la conception objet. UML est une représentation standardisée d'un système orienté objet. Il ne représente pas une méthode de conception mais une notation graphique normalisée de présentation de certains concepts pour modéliser des systèmes objets. L'usage d'une représentation graphique est un excellent complément par rapport aux représentions textuelles. En effet, l'une comme l'autre sont ambigües mais leur utilisation simultanée permet de diminuer ces ambigüités. Un dessin permet bien souvent de visualiser clairement ce qu'un texte peut exprimer difficilement et un bon commentaire permet d'enrichir une figure.

En général, UML est un standard de présentation des concepts qui permet de faciliter le dialogue entre les différents acteurs du projet : les autres analystes, les développeurs, et même les utilisateurs. [Doudoux, 2014]

UML est composé de plusieurs diagrammes. Dans notre projet, nous présentons les diagrammes des cas d'utilisations, de séquence et de classes.

3. Spécification des besoins

La détermination et la compréhension des besoins sont souvent difficiles car les intervenants sont noyés sous de grandes quantités d'informations. Les besoins sont souvent exprimés de manière non structurée, sans forte cohérence. En conséquence, le cahier des charges initial est flou et en constante évolution. Lorsque les besoins se précisent ou évoluent – ce qui est toujours le cas – il devient très difficile d'apprécier l'impact et le coût d'une modification. Les cas d'utilisation recentrent l'expression des besoins sur les utilisateurs, en partant du point de vue très simple qui veut qu'un système est avant tout construit pour ses utilisateurs.

Les cas d'utilisation permettent une approche complète pour l'ensemble du cycle de vie, depuis le cahier des charges jusqu'à la réalisation. Un cas d'utilisation est une manière spécifique d'utiliser un système. C'est l'image d'une fonctionnalité du système, déclenchée en réponse à la stimulation d'un acteur externe.
Le modèle des cas d'utilisation comprend les acteurs, le système, les cas d'utilisation eux-mêmes ainsi que des notes servant à décrire des contraintes ou faire des commentaires.

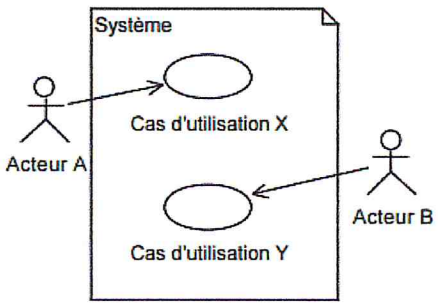


Figure 12: Représentation d'un modèle des cas d'utilisation

Les acteurs du système :

Notre système interagit avec deux types d'utilisateurs :

- Un **utilisateur** : il peut créer des processus ETL, les paramétrer et les exécuter.
- Un **administrateur** : Il prend en charge la création, le paramétrage des projets d'ETL et la gestion des comptes d'utilisateurs.

3.1 Diagramme global du système :

Le système permet aux différents utilisateurs de gérer, configurer ainsi qu'exécuter des processus ETL pour le rafraichissement périodique d'un entrepôt de données. Le diagramme suivant regroupe les fonctionnalités globales qui assurent le fonctionnement du système. Chacun des cas apparaissant sur ce diagramme sera développé ultérieurement afin de le décrire plus en détails.

3.1.1 Diagramme :

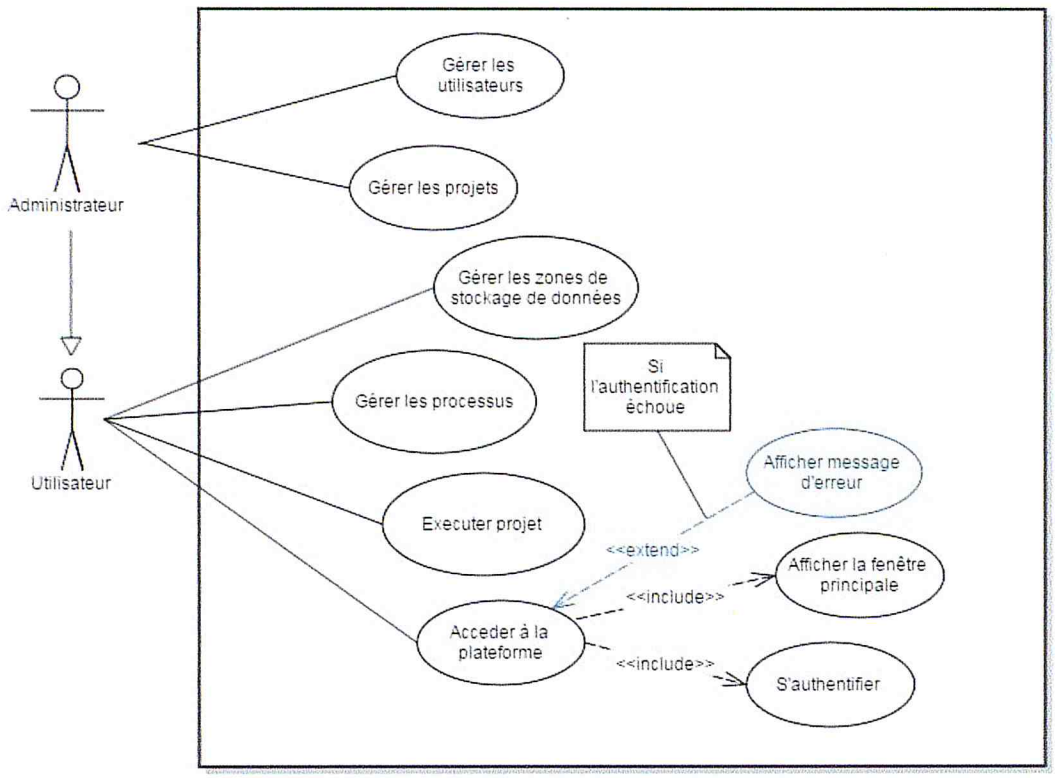


Figure 13: Diagramme de cas d'utilisation global du système

3.1.2 Description du diagramme :

- **Gérer les utilisateurs :** ce cas permet à l'administrateur de gérer les comptes d'utilisateurs pouvant accéder à la plateforme.

- **Gérer les projets :** ce cas permet à un acteur de gérer la totalité des projets de la plateforme.
- **Gérer les zones de stockage de données:** assure la gestion des métadonnées sur la zone de données (source de données, schéma des tables, format des attributs ...) qui contiennent les données constituant l'élément essentiel dans l'élaboration des processus ETL.
- **Gérer les processus :** comporte la création et la mise à jour processus.
- **Exécuter projet :** permet à l'acteur d'exécuter de bout à bout de l'ensemble des processus ETL d'un projet.
- **Accéder à la plateforme :** la connexion au système permet à un acteur l'accès à la plateforme après une authentification réussite.
- **S'authentifier :** avant de se connecter au système, chaque acteur doit être identifié par un login et un mot de passe afin d'avoir les permissions d'accès au système.
- **Afficher la fenêtre principale :** l'affichage de la fenêtre principale de la plateforme.
- **Afficher message d'erreur :** En cas d'erreur d'authentification, l'acteur sera notifié par un message d'erreur.

3.2 Diagramme « Gestion de projet » :

L'administration complète de la plateforme est assurée par un administrateur. Celui-ci se charge de contrôler l'accès des utilisateurs et gérer l'ensemble des projets ETL. Ceci est modélisé par le diagramme de cas d'utilisation suivant :

3.2.1 Diagramme :

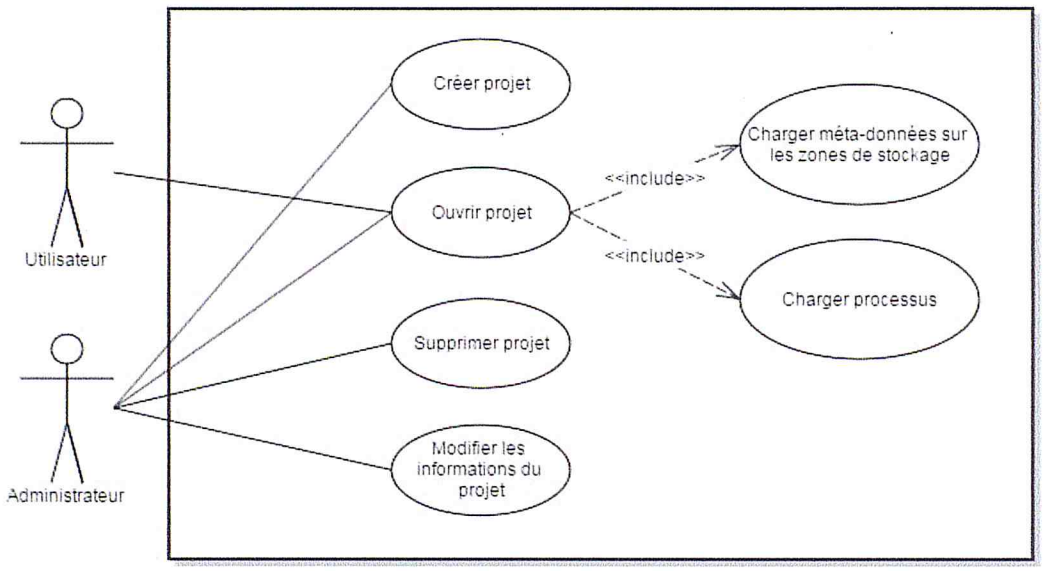


Figure 14: Diagramme du cas d'utilisation « Gestion de projet »

3.2.2 Description du diagramme :

- **Créer projet :** permet la création d'un nouveau projet (nom, description, emplacement de stockage des processus, emplacement DSA).
- **Ouvrir projet :** ce cas permet d'ouvrir un projet existant.
- **Supprimer projet :** la suppression d'un projet du système.
- **Modifier informations du projet :** mettre à jour les informations d'un projet.
- **Charger métadonnées sur les zones de stockage de données :** à l'ouverture d'un projet existant, les métadonnées constituant ce dernier devront être chargées sur les différentes zones de stockage.
- **Charger processus :** consiste à charger les processus d'ETL reliés au projet.

3.3 Diagramme « Gestion des utilisateurs » :

La gestion des comptes d'utilisateurs consiste à mettre à jour une liste d'utilisateurs pouvant accéder au système selon deux niveaux de privilèges: administrateur et utilisateur final.

3.3.1 Diagramme :

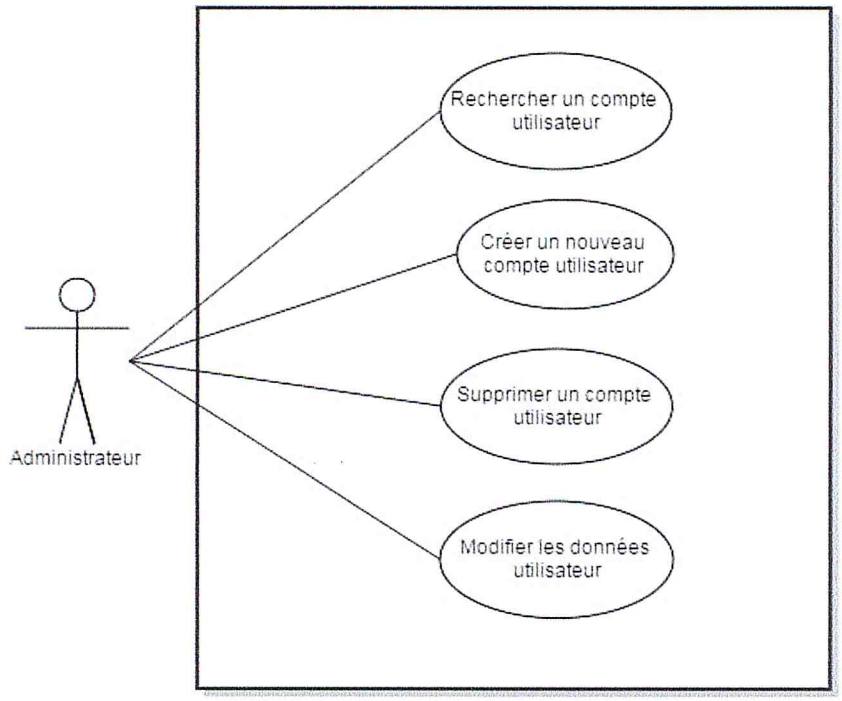


Figure 15: Diagramme de cas d'utilisation « Gestion des utilisateur »

3.3.2 Description du diagramme :

- **Rechercher un compte d'utilisateur :** ce cas permet à un administrateur de rechercher un compte d'utilisateur parmi l'ensemble des utilisateurs de la plateforme.
- **Créer un nouveau compte d'utilisateur :** l'administrateur peut créer un nouveau compte. Chaque personne accédant au système ne devrait posséder que son propre et unique compte caractérisé par : nom, pseudo, mot de passe et rôle.
- **Supprimer un compte utilisateur :** supprimer un compte utilisateur existant, donc il ne pourra pas y accéder à la plateforme.
- **Modifier les données utilisateurs :** mettre à jour les données de l'utilisateur (nom, mot de passe).

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

3.4.1 Diagramme :

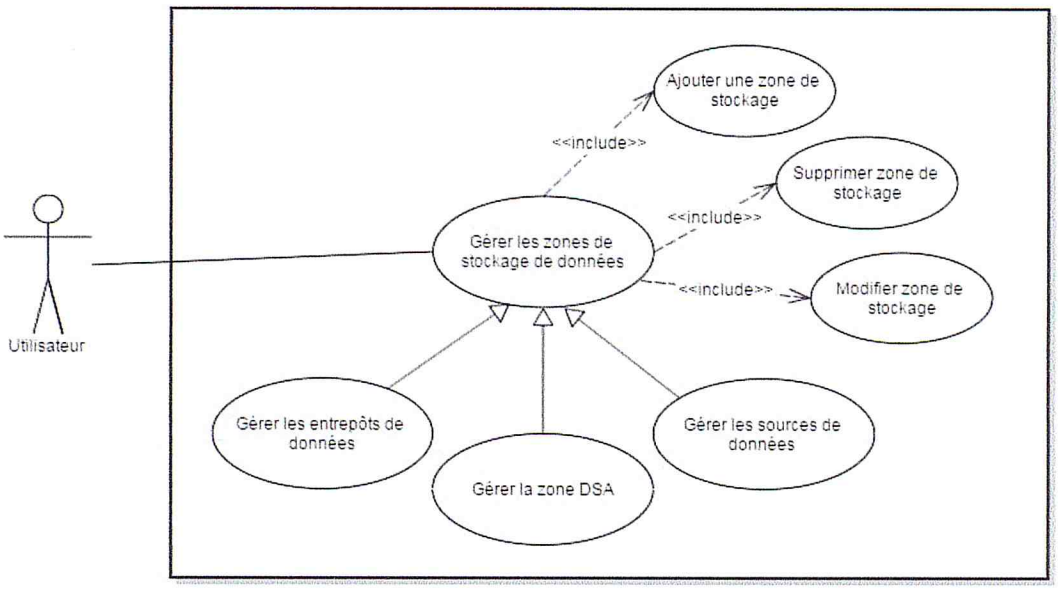


Figure 16: Diagramme du cas d'utilisation « Gestion de zones de stockage de données »

3.4.2 Description du diagramme :

- **Gérer les zones de stockages de données:** la gestion des zones de stockages de données est assurée
- **Ajouter une zone de stockage :** ajouter une nouvelle zone de stockage en désignant son emplacement physique (cas d'un fichier) ou la chaine de connexion (cas d'une base de données).
- **Supprimer zone de stockage :** supprimer une zone de stockage et son contenu par conséquent.
- **Modifier zone de stockage :** modifier l'emplacement d'une zone de stockage.
- **Gérer les entrepôts de données :** consiste à gérer et mettre à jour les entrepôts de données à charger périodiquement dans un projet ETL.
- **Gérer la zone DSA :** désigner la zone de stockage temporaire des données d'un projet.

- **Gérer les sources de données :** la gestion des sources dont on v a extraire les données à traiter dans un projet.

3.5 Diagramme « Gestion des processus » :

3.5.1 Diagramme :

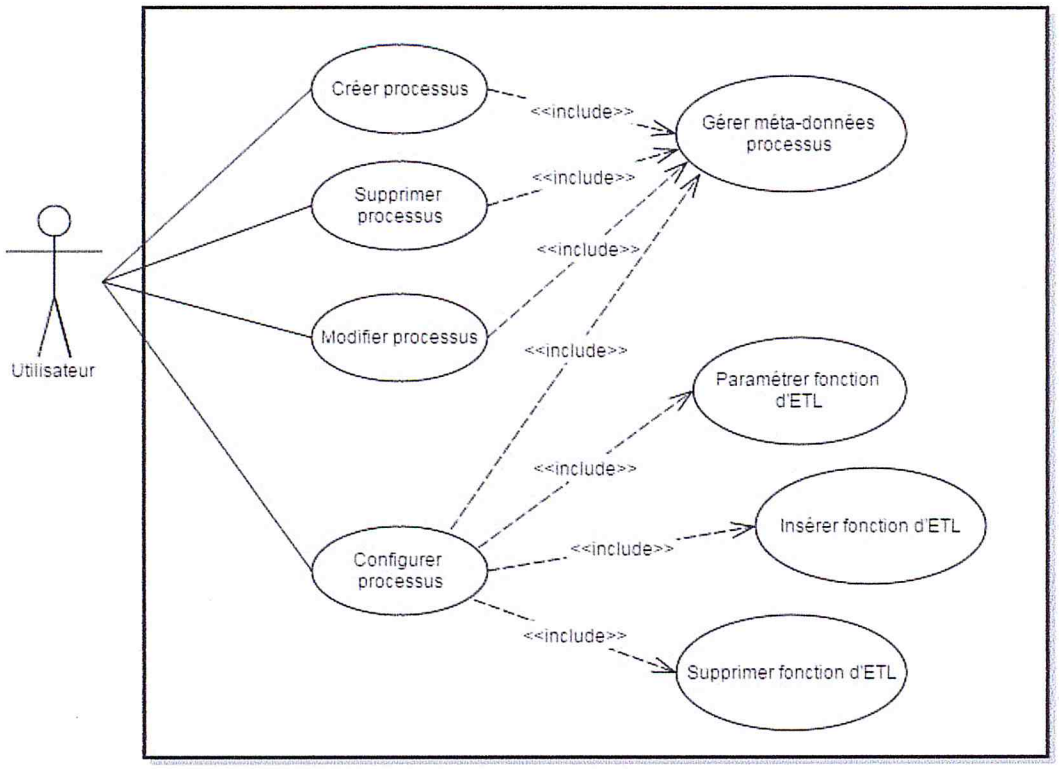


Figure 17: Diagramme du cas d'utilisation « Gestion des processus »

3.5.2 Description du diagramme :

- **Créer processus :** il permet à un acteur de créer un nouveau processus ETL, on spécifiant son nom et sa description.
- **Supprimer processus :** un acteur peut supprimer un processus existant.
- **Modifier processus :** un acteur a la possibilité de modifier les données d'un processus, son nom ou sa description ainsi que sa configuration (les paramètres de ses fonctions).
- **Configurer processus :** la configuration d'un processus ETL consiste à désigner l'ensemble des sources de données à partir lesquelles les données vont être extraire et transformées passant par un ensemble de fonctionnalités

ETL insérées et enfin les charger vers une destination (ED) déjà configurée aussi.

- **Gérer méta-données processus :** la gestion des méta-données des processus ETL consiste à mettre à jour les fichiers XML contenant ces données à chaque opération de création, suppression, modification ou configuration. (plus de détails dans le chapitre VI).
- **Paramétrer fonction d'ETL :** cela consiste à spécifier les données d'entrés/sorties de la fonction ainsi que son ordre d'exécution dans le processus.
- **Insérer fonction d'ETL :** un acteur insère une fonction d'ETL parmi l'ensemble des fonctions implémentées et stockés dans une librairie système.
- **Supprimer fonction d'ETL :** supprimer une fonction du modèle de processus.

3.6 Diagramme « Exécution de projet » :

3.6.1 Diagramme :

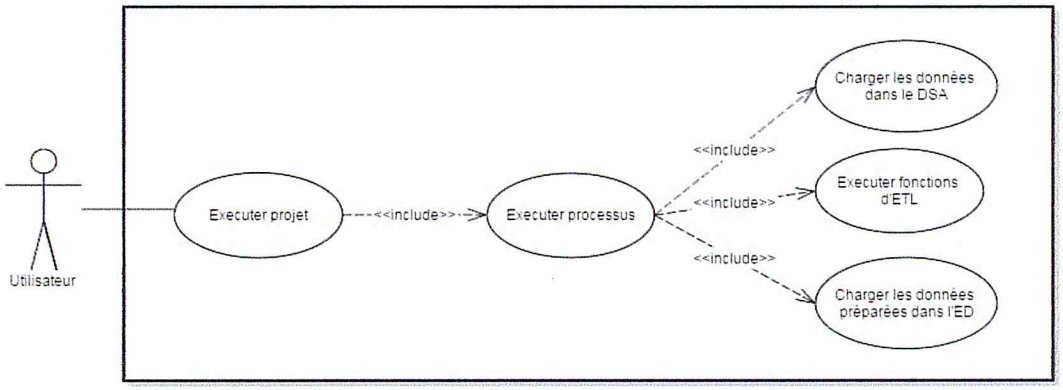


Figure 18: Diagramme du cas d'utilisation « Exécution de projet »

3.6.2 Description du diagramme :

- **Exécuter projet :** Ce cas représente le résultat final du système visé. Il permet à un administrateur d'exécuter l'ensemble des processus ETL crées et configurés afin d'atteindre un objectif souhaité. Dans une première version de la plateforme en se contente sur l'exécution d'un seul processus ETL.

- **Exécuter processus :** Lorsqu'un processus est configuré, ce cas permet d'accéder aux sources de données paramétrées, les chargé dans la zone DSA correspondante et exécuter les fonctions selon l'ordre spécifié jusqu'à charger la destination (ED) par les données transformées.
- **Charger les données dans le DSA :** afin de préparer les données extraites des sources paramétrées d'un processus ETL, il est nécessaire de les charger dans la zone DSA spécifiée et effectuer les différentes opérations de préparation de données.
- **Exécuter fonctions d'ETL :** l'exécution d'une fonction ETL fait appel à son algorithme implémenté dans la librairie de fonctions la plateforme. Les données paramétrées en entrés seront transformées et passées en sortie. Tout cela un ordre d'exécution spécifié.
- **Charger les données préparées dans l'ED :** une fois les données sont passées par une suite de fonctions ETL, c.-à-d. transformées, elles vont être chargées dans l'ED paramétré d'où le résultat final de l'exécution du processus ETL sera atteint.

4. Conception du système

4.1 Diagramme de classes :

Alors que les diagrammes de cas d'utilisation nous ont montré le système du point de vue des acteurs, les diagrammes de classes en montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation.

Un diagramme de classe représente la structure statique du système, car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application. Chaque langage de Programmation orienté objet donne un moyen spécifique d'implémenter le paradigme objet, mais le diagramme de classes permet de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier.

Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation et plusieurs types de dépendances, telles que la réalisation et l'utilisation.

Le diagramme de classes suivant modélise le schéma de la zone de stockage des données relatives aux objets du système à savoir : projet, processus ETL, fonction ETL, zones de stockage (sources, DSA, DW) et utilisateur. Lors de l'utilisation du système, ce dernier insère de manière automatique et transparente toutes les données caractérisant les aspects manipulés par l'utilisateur.

- **Diagramme :**

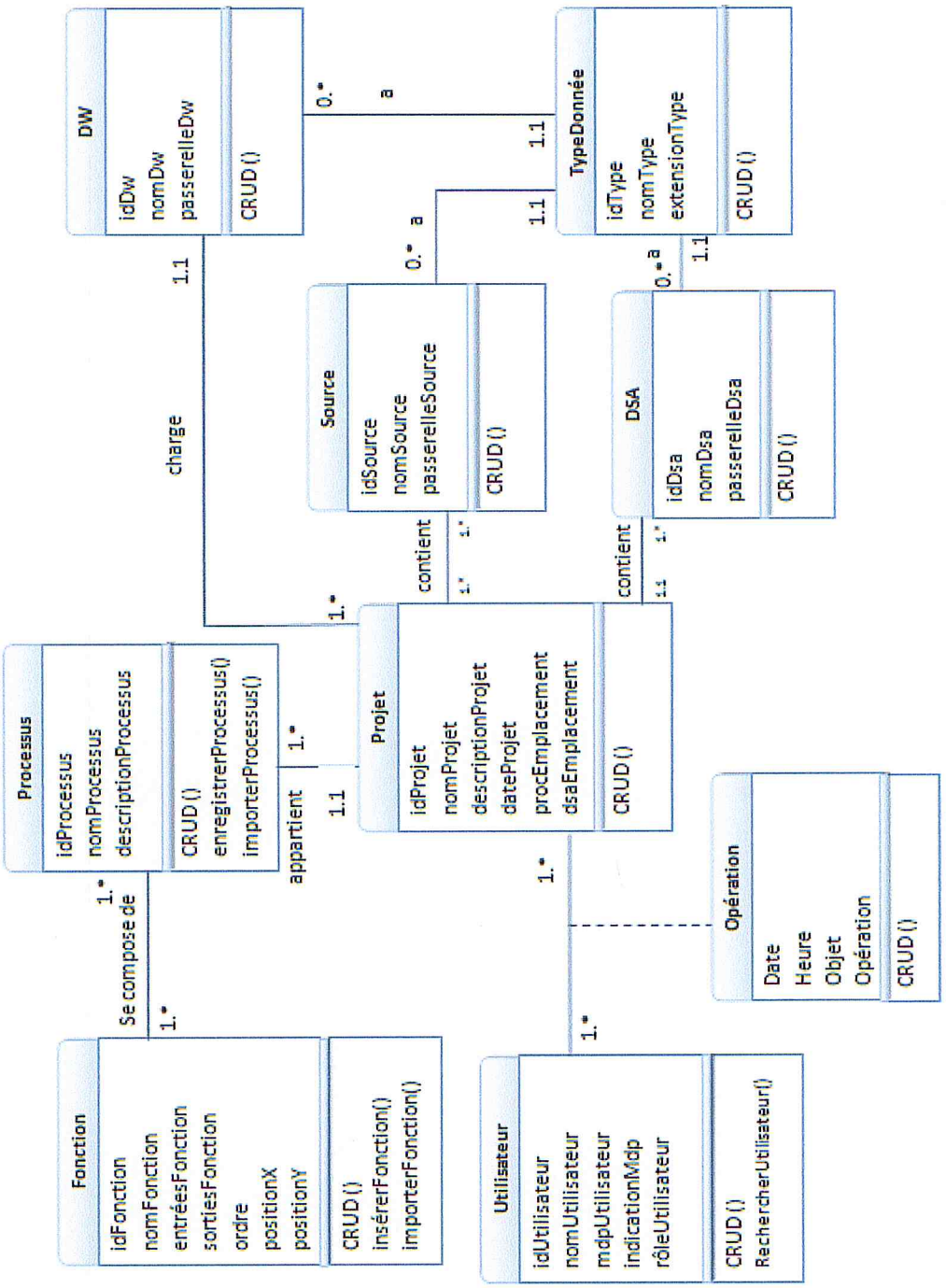


Figure 19: diagramme de classes

- **Description du diagramme :**

Nom de classe	Attribut	Description
Projet : Cette classe regroupe les propriétés d'un projet	idProjet : Entier	La clé primaire identifiant un projet
	nomProjet : Texte	Nom du projet
	emplacementProjet	L'emplacement où le projet est enregistré
	descriptionProjet : Texte	Description du projet
	dateProjet : Date	Date de création du projet
	procEmplacement : Texte	Emplacement de stockage des processus
	dsaEmplacement : Texte	Emplacement de la zone de stockage DSA du projet
Processus : Cette classe modélise un processus ETL	idProcessus: Entier	Identifiant du processus
	nomProcessus : Texte	Nom spécifique au processus
	descriptionProcessus : Texte	Descriptif du processus
Utilisateur : Représente les informations concernant un compte d'utilisateur	idUtilisateur: Entier	Identifiant d'un compte
	nomUtilisateur : Texte	Nom / pseudo utilisateur
	mdpUtilisateur : Texte	Mot de passe de connexion
	indicationMdp : Texte	Indication du mot de passe, en cas de l'oublie du mot de passe
	rôleUtilisateur : Caractère	Désigne le rôle d'un acteur: administrateur ou utilisateur.

Source : regroupe les informations d'une source de données	idSource: Entier	Identifiant de la source
	nomSource : Texte	Nom de la source de données
	passerelleSource : Texte	Emplacement source de données
DSA : regroupe les attributs d'un fichier DSA	idDsa: Entier	Identifiant d'un fichier DSA
	nomDsa : Texte	Nom du fichier
	passerelleDsa : Texte	Le chemin/ passerelle vers le fichier DSA
DW : Décrit les propriétés d'un ED	idDw: Entier	Identifiant de l'ED
	nomDw : Texte	Nom de l'ED
	passerelleDw : Texte	Emplacement de l'ED ou peut être le nom du serveur (cas d'une base de données).
TypeDonnées: Définit un type de source de données	idType: Entier	Identifiant de type de données
	nomType : Texte	Nom spécifié au type : Base de données, fichier xml, Excel etc.
	extensionType : Texte	L'extension du type : sql, xml, txt, xlsx etc.
Opération : Décrit les opérations effectuée sur un objet d'un projet par un utilisateur	Date : Date	La date d'opération
	Heure : Heure	L'heure d'opération
	Objet : Texte	L'objet sur lequel l'utilisateur a effectué une opération.
	Opération : Texte	L'opération effectuée par un utilisateur dans une date et une heure données, sur un objet: insertion, modification ou suppression.

Méthodes :

Nom de classe	Méthode	Désignation
Processus	EnregistrerProcessus()	Permet de générer le modèle graphique du processus sous forme d'un fichier XML stocké dans l'emplacement indiqué dans la colonne procEmplacement de la classe Projet.
	ImporterProcessus()	Permet d'importer/récupérer le modèle graphique du processus à partir du fichier XML afin de l'afficher dans un espace graphique.
Fonction	InsérerFonction()	Permet d'ajouter un composant graphique d'une fonction ETL au processus ETL (ajouter sa position (X ,Y))
	ImporterFonction()	Consiste à récupérer les paramètres d'une fonction lors de l'importation d'un processus.
Utilisateur	RechercherUtilisateur()	Permet à un administrateur de rechercher un compte d'utilisateur
Projet, Source, DW, DSA, TypeDonnée, Utilisateur	CRUD()	Create-Read-Update-Delete permet d'effectuer une des opérations de mise çà jour.

4.2 Diagrammes de séquences :

4.2.1 Diagramme de séquence « Créer projet » :

- **Diagramme :**

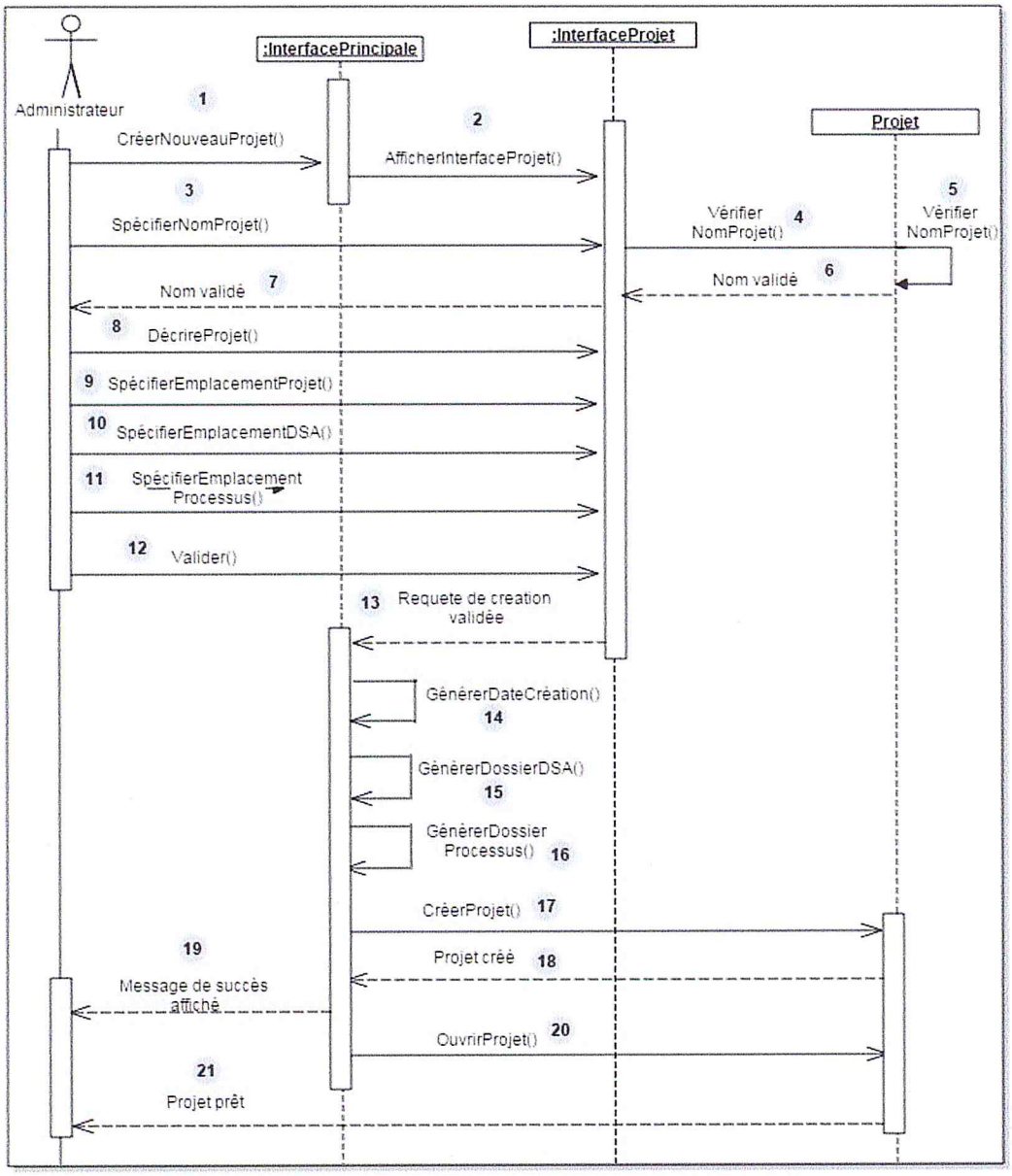


Figure 20: Diagramme de séquence « Créer un projet »

- **Description du scénario :**

Le diagramme décrit les interactions entre les objets dans le cas d'utilisation « créer un projet ».

(1) L'utilisateur s'authentifie et accède à l'object « InterfacePrincipale » pour demander la création d'un nouveau projet.

(2) Un nouveau objet de type « InterfaceProjet » est crée. Il contient un ensemble de champs à remplir.

(3) L'utilisateur spécifie un nom de projet.

(4) L'objet « InterfaceProjet » fait appel à un objet « Projet » afin de vérifier la syntaxe du nom saisi (le nom doit être unique et ne contient pas de caractères spéciaux comme ', / etc.).

(5) L'objet « Projet » vérifie le nom.

(6) Si le nom est valide, l'objet « Projet » notifie l'objet « InterfaceProjet ».

(7) L'objet « InterfaceProjet » informe l'utilisateur que le nom est valide.

(8), (9), (10), (11) L'utilisateur rempli le reste des champs en spécifiant :

- La description du projet.
- L'emplacement de stockage du projet.
- L'emplacement de la zone de stockage des processus.
- L'emplacement de la zone de stockage DSA.

(12) Une fois tous les champs sont bien remplis, l'utilisateur valide la création du projet.

(13) L'objet « InterfaceProjet » envoie la requête de création à l'objet « InterfacePrincipale ».

(14), (15), (16) Quand l'objet « InterfacePrincipale » reçoit la requête de création contenant les informations du projet, il génère la date de création du projet ainsi que les dossiers des zones de stockage de données spécifiés (processus, DSA).

(17) Il exécute la requête de création du projet, donc un objet de type « Projet » est crée.

(18) Il reçoit le résultat de création du projet.

(19) Il notifie l'utilisateur que le projet a été bien en affichant un message de succès.

(20) Il demande l'ouverture du nouveau projet automatiquement.

(21) L'objet « Projet » est prêt pour l'utilisateur.

4.2.2 Diagramme de séquence « Configurer un processus » :

- **Diagramme :**

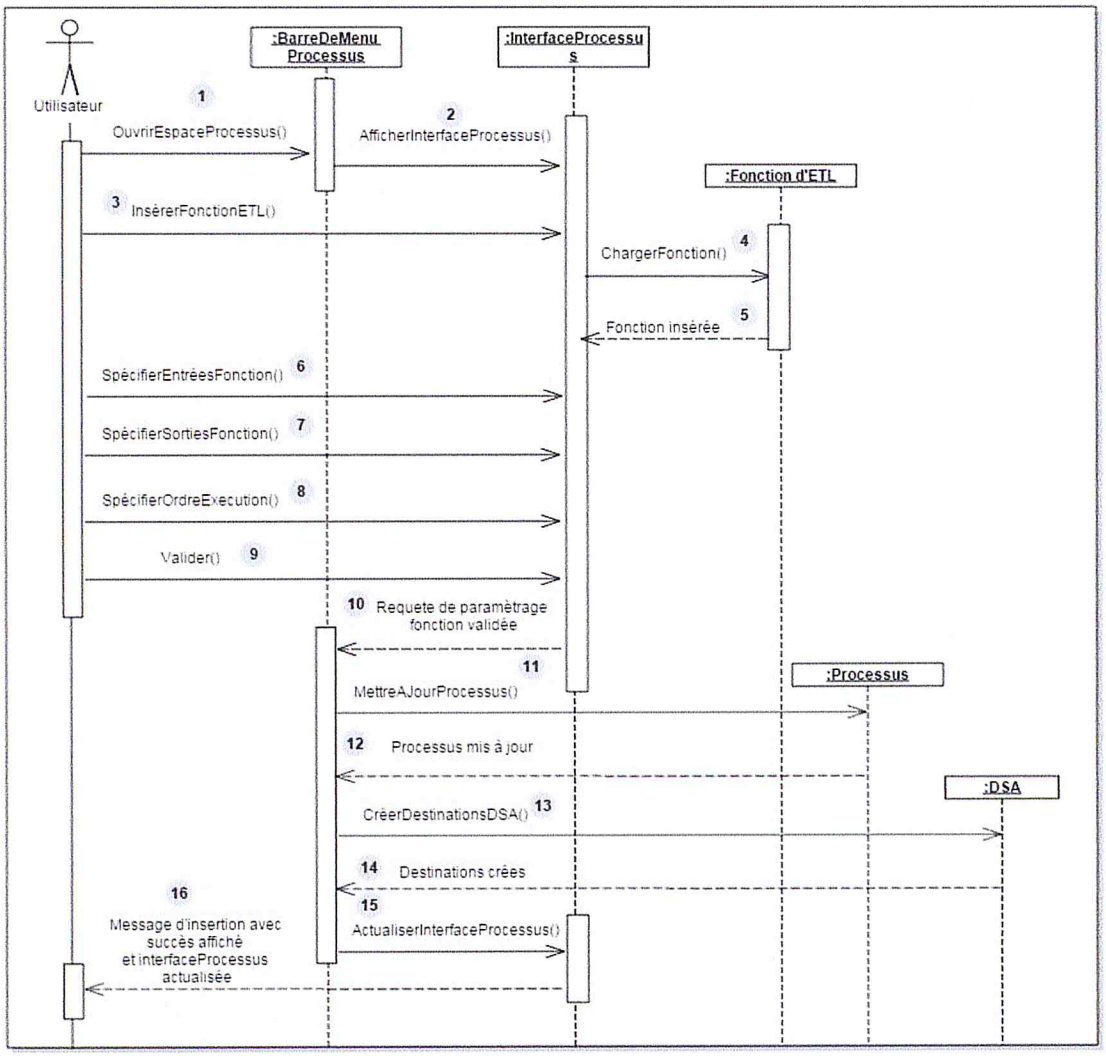


Figure 21: Diagramme de séquence « Configurer un processus »

- **Description du scénario :**

Le diagramme décrit les interactions entre les objets intervenants dans le cas d'utilisation « configuration d'un processus » :

(1) L'utilisateur ouvre un processus (en cliquant sur la barre de menu).

(2) Le système affiche l'espace du processus, donc un objet de type «InterfaceProcessus » est crée.

(3) L'utilisateur choisit une fonction de la librairie des fonctionnalités d'ETL afin de l'insérer dans l'espace processus.

(4) L'objet « InterfaceProcessus » charge le modèle de la fonction (entés/sorties, ordre) choisi par l'utilisateur.

(5) Quand le modèle est bien chargé, le composant graphique de la fonction est inséré dans l'objet « InterfaceProcessus ».

(6), (7), (8) L'utilisateur paramètre la fonction insérée en spécifiant :

- Les entrés de la fonction.
- Les sorties.
- L'ordre d'exécution.

(9) Une fois la fonction est bien paramétrée, l'utilisateur valide sa configuration.

(10) L'objet « InterfaceProcessus » envoie la requête de paramétrage de la fonction à « BarreDeMenuProcessus ».

(11) Quand l'objet « BarreDeMenuProcessus » reçoit la requête paramétrage de la fonction contenant ses paramètres, il mit à jour le modèle de l'objet « Processus » dont l'utilisateur est entrain de configurer.

(12) Le modèle de l'objet « Processus » est mis à jour.

(13) l'objet « BarreDeMenuProcessus » crée les destinations intermédiaires dans la zone de stockage DSA afin de les utiliser dans l'étape de l'exécution pour préparer les données, donc l'objet « DSA » du projet courant est mis à jour.

(14) Les destinations intermédiaires sont crées avec succès.

(15) l'objet « BarreDeMenuProcessus » actualise l'objet « InterfaceProcessus » pour afficher les nouveaux composants du processus (les fonctions et les flèches déterminant l'ordre d'exécution).

(16) L'utilisateur est notifié par un message de succès et l'interface du processus est actualisée.

4.2.3 Diagramme de séquence « Exécuter un processus » :

- **Diagramme :**

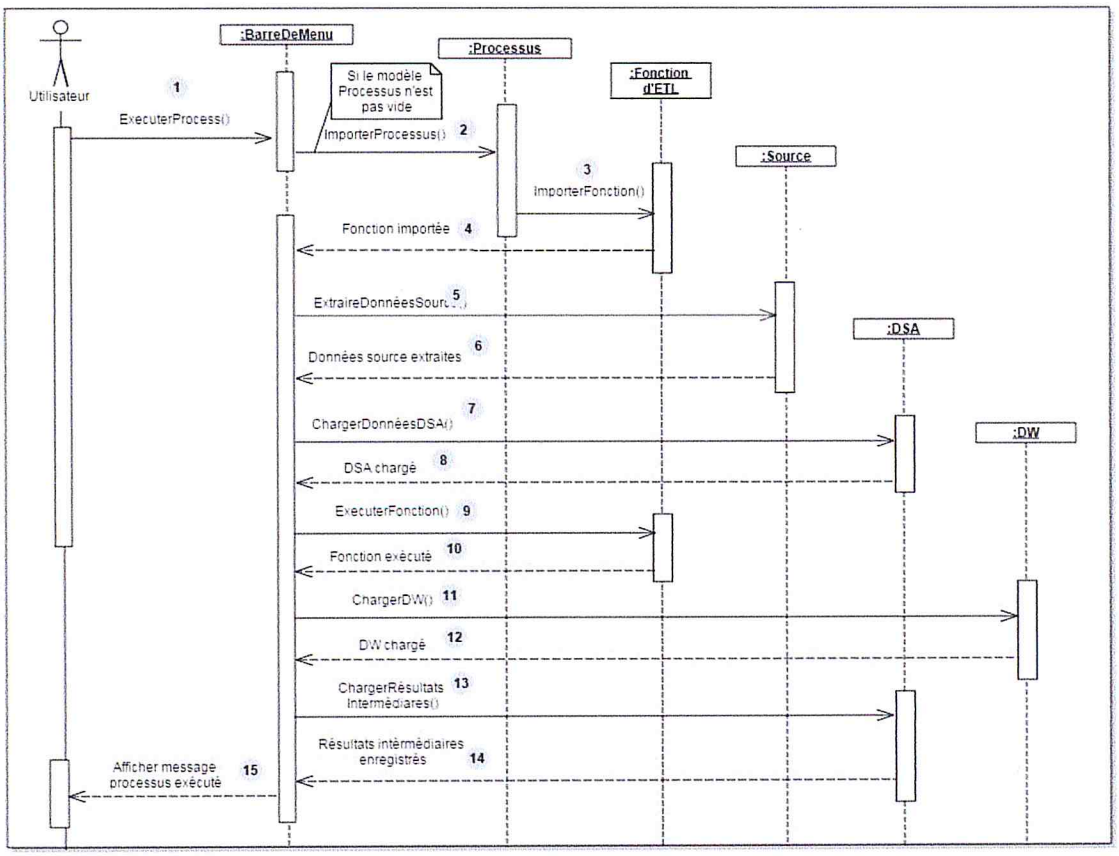


Figure 22: Diagramme de séquence « Exécuter processus »

- **Description du scénario :**

Le diagramme décrit les interactions entre les objets intervenants dans le cas d'utilisation « exécuter processus » :

(1) L'utilisateur exécute un processus (en cliquant sur la barre de menu).

(2) L'objet « BarreDeMenu » accède à l'objet de type « Processus » pour importer le modèle du processus s'il existe (s'il est configuré).

(3) L'objet « Processus » importe l'ensemble des fonctions configurées afin de les exécuter dans l'ordre spécifié.

(4) Les objets de type « Fonctions d'ETL » sont importés.

Pour chaque fonction d'ETL, l'objet « BarreDeMenu » :

(5) extrait les données sources paramétrées (entrées).

(6) charge les données extraites (7) dans la zone de stockage temporaire DSA.

(9) après le chargement des données préparées (8), il accède à la classe correspondante à la fonction dans la librairie des fonctionnalités d'ETL et exécute l'algorithme approprié.

(11) Les résultats obtenus de l'exécution (10) sont ensuite chargés dans la destination ED (DW) paramétré.

(13) Une fois l'entrepôt de données cible chargé (12), s'il existe des données intermédiaires, il les charge dans la zone DSA.

(14) Les résultats intermédiaires enregistrés sur la zone DSA.

(15) L'utilisateur est notifié par un message après la fin de l'exécution du processus.

5. Conclusion

Dans ce chapitre, nous avons modélisé d'une vue statique et dynamique notre système afin d'entamer, dans le prochain chapitre, l'implémentation et la mise en place de notre plateforme.

PARTIE 3

CHAPITRE VI

IMPLEMENTATION

1. Introduction

Une fois notre système conçu, nous procédons à sa présentation globale et les étapes de la mise en œuvre de notre plateforme « ETLBuild ».

2. Architecture du système :

Il s'agit d'une plateforme d'intégration de données issues de diverses sources. Ces dernières passent par une série de fonctions d'ETL afin de les transformer et les charger dans un entrepôt de données. Pour répondre à cet objectif, la plateforme est structurée globalement comme le montre la figure 23.

Figure 23: Architecture globale du système

2.1 Environnement intégré de configuration de processus ETL :

L'environnement nécessaire pour définir un processus ETL est configuré préalablement au niveau d'un projet, c'est l'élément fédérateur de tous les objets relatifs à l'intégration des données d'un domaine particulier. Nous y trouverons, principalement, six compartiments : processus, sources, DSA, DW, fonctions ETL et l'espace de configuration des processus.

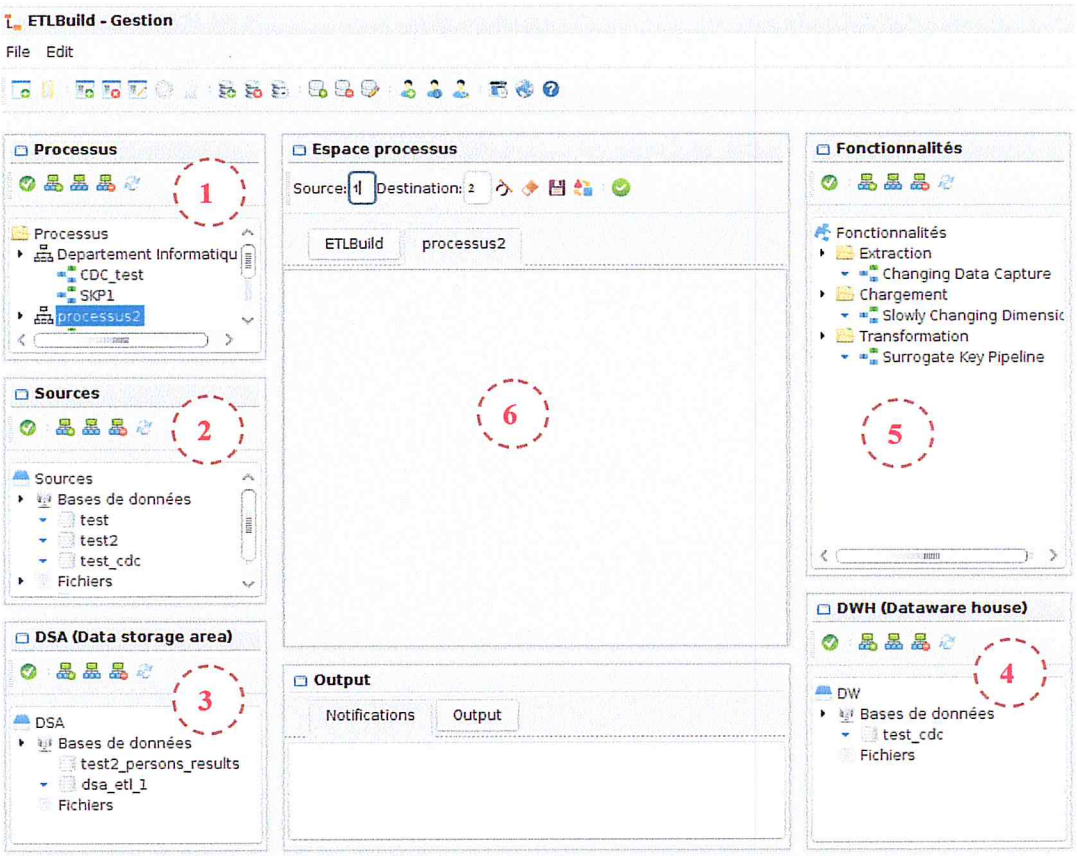


Figure 24 : Interface principale de la plateforme

(1) Compartiment des processus :

En développant le compartiment « processus », les processus déjà configurés seront affichés. La sélection d'un processus, permettra d'afficher, dans l'espace de configuration (6), le schéma de celui-ci (sous forme d'un workflow). Il pourra alors être modifié, validé, supprimé etc.

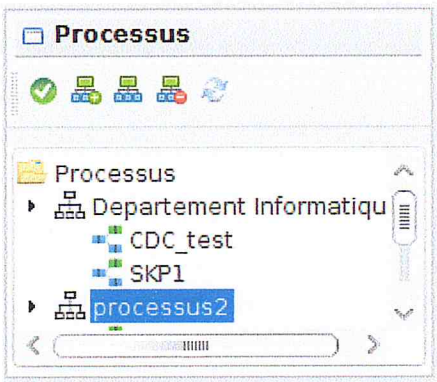


Figure 25: Compartiment « Processus »

(2) Compartiment des sources : De la même manière, le compartiment « sources de données » contient une ou plusieurs sources déjà configurées. Pour rajouter une source dans ce compartiment, la plateforme demandera à se connecter sur la base de données ou localiser physiquement l'emplacement s'il s'agit d'un fichier (csv, txt, excel, …).

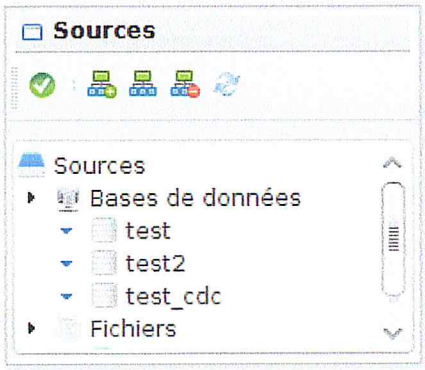


Figure 26: Compartiment « Sources de données »

(3), (4) Compartiments DSA, DW : Ils affichent, sous forme d'arborescences, les bases de données/fichiers situés dans les zones de stockage DSA et DW.

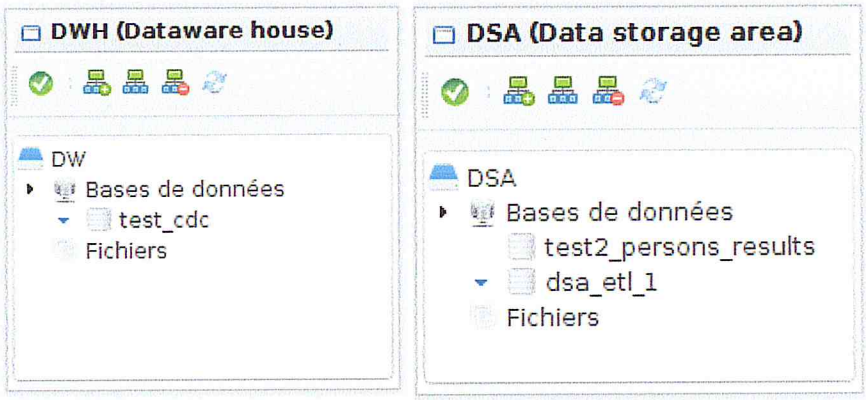


Figure 27 : Compartiments « DSA » et « DW »

(5) Compartiment des fonctions d'ETL :

Lorsqu'il s'agit d'insérer des fonctions d'ETL dans le schéma d'un processus ETL, il faudra accéder à la librairie ETL qui se présente aussi comme une *palette* de fonctions d'ETL. Organisée sous forme d'arborescence, il faudra parcourir celle-ci en catégories pour aboutir à la fonction appropriée.

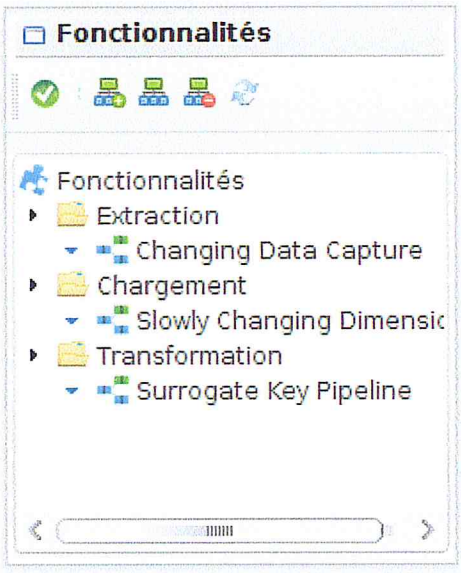


Figure 28: Compartiment « Fonctions d'ETL »

(6) Espace de configuration des processus :

Permet la lisibilité du processus ETL en affichant son schéma dans un espace graphique de configuration contenant une palette d'outils (ajouter, supprimer lien, sauvegarder schéma etc).

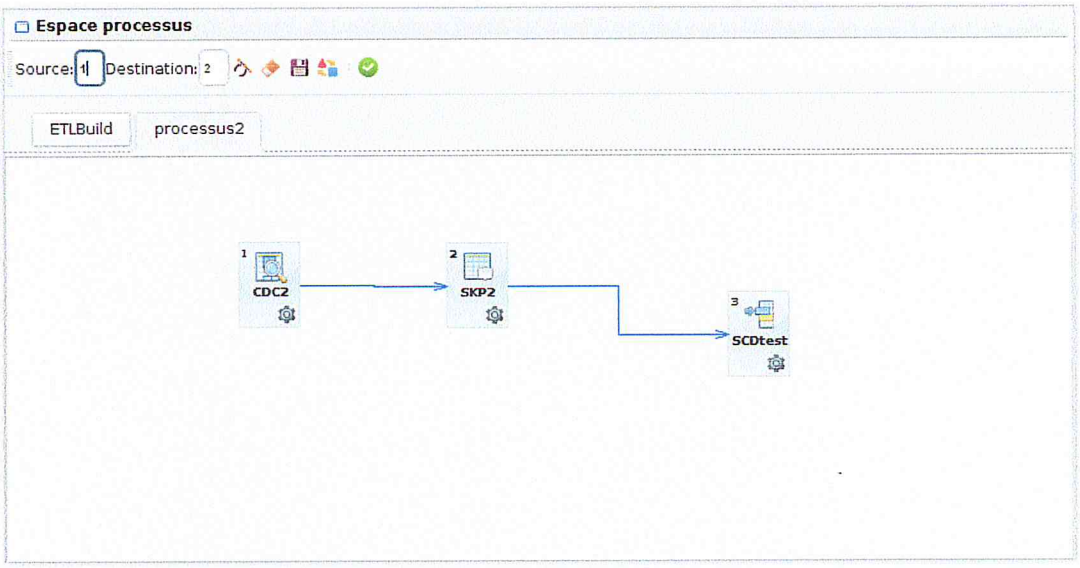


Figure 29: Espace de configuration des processus

2.2 La Structure modulaire de la plateforme :

2.2.1 Module Librairie des fonctionnalités :

Il s'agit d'un ensemble de fonctionnalités d'ETL sous forme de classes java. Elles sont regroupées dans un même package qui constitue la librairie de fonctions d'ETL. Dans une première version de la plateforme, il s'agit de disposer des fonctions les plus courantes (Figure 25).

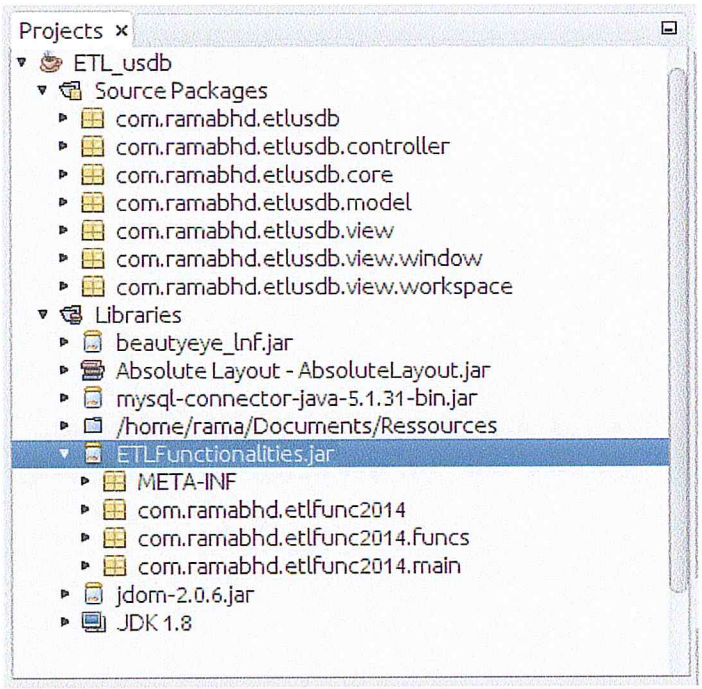


Figure 30 : Librairie de fonctions d'ETL

2.2.2 Module PROJET :

Ce module est chargé de la gestion des projets. Il permet à l'utilisateur de créer et de paramétrer un nouveau projet et de charger tous les éléments d'un projet existant dès son ouverture. Aussi, lorsque des modifications sont opérées dans le projet, celles-ci sont prises en considération.

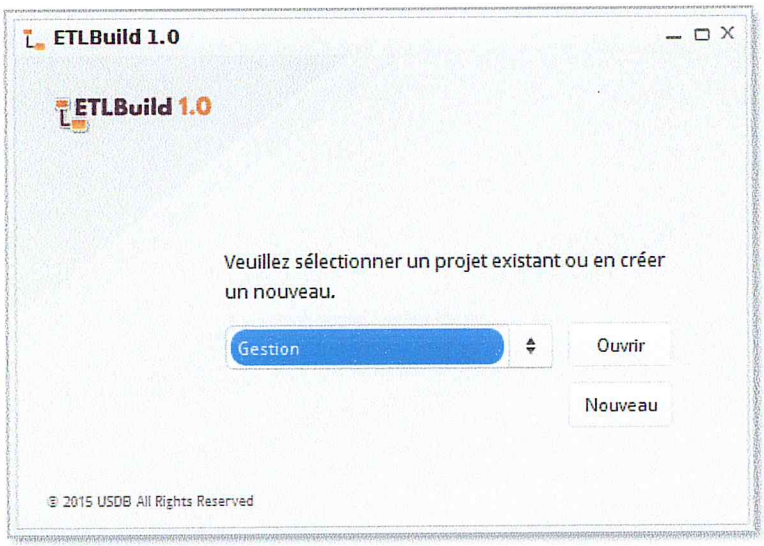


Figure 31: Interface d'accès à la plateforme

2.2.3 Module DATA :

C'est le module de gestion de données. Il permet d'accéder aux différentes zones de stockage de données (sources, DSA et ED), de se connecter aux fournisseurs de données pour récupérer les métadonnées nécessaires (tables, attributs …) et les inscrire par catégorie (sources, DSA, ED) dans l'interface de la plateforme et dans le catalogue de la plateforme (*voir section 2.3*). Lors de l'exécution d'une fonction d'ETL, le module DATA permet de lire les données en entrée (inputs) de celle-ci à partir de ces sources et écrire ses sorties (outputs) vers d'autres fournisseurs de données (DSA, ED).

2.2.4 Module PROCESS :

Le module de configuration des processus ETL permet, grâce aux modules DATA et Librairies des fonctionnalités, de construire un processus ETL de bout en bout :

- Extraction à partir des sources configurées dans le module DATA.
- Insérer une série de fonctions à partir le module Librairie des fonctionnalités pour la transformation de données.
- Charger dans une destination configurée aussi dans le module DATA.
- Stocker les résultats intermédiaires dans la zone DSA configurée dans le module DATA.

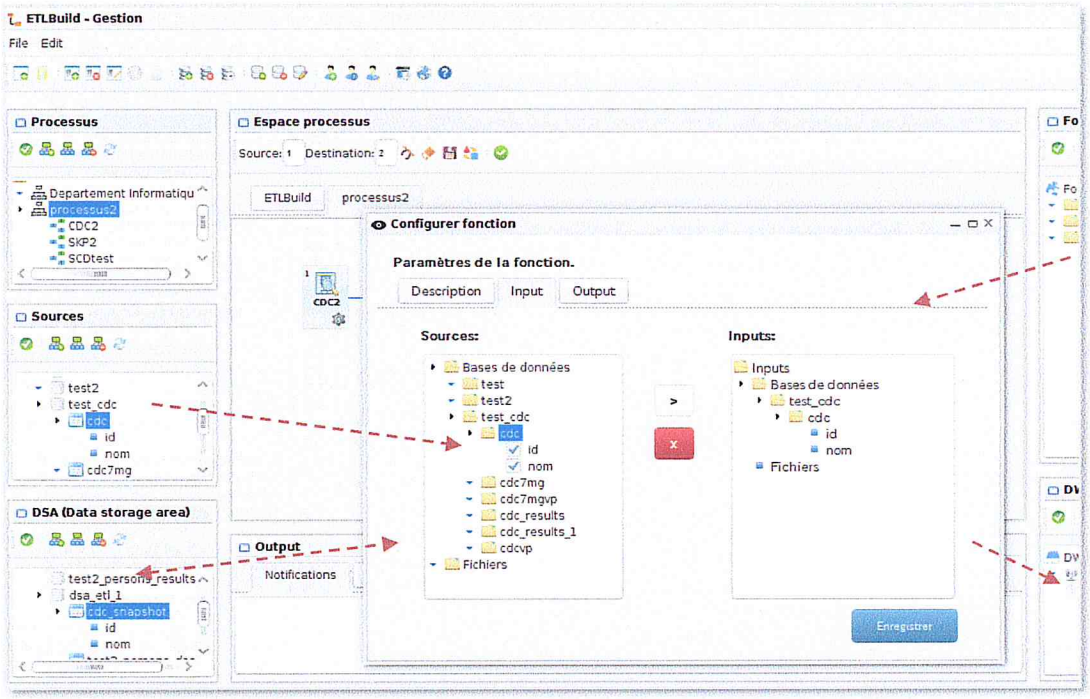


Figure 32 : configuration d'un processus ETL

2.2.5 Module EXEC :

Il constitue la finalité de la plateforme, Lorsqu'un processus est bien paramétré (fonctions d'ETL, données d'E/S de chacune, ordre d'exécution, résultats intermédiaire, destination de données), le module est responsable de :

- Accéder aux sources de données paramétrées et les charger dans le DSA.
- Exécuter les fonctions selon l'ordre spécifié jusqu'à atteindre le résultat final du processus.
- Charger les données transformées dans leur destination (DW).

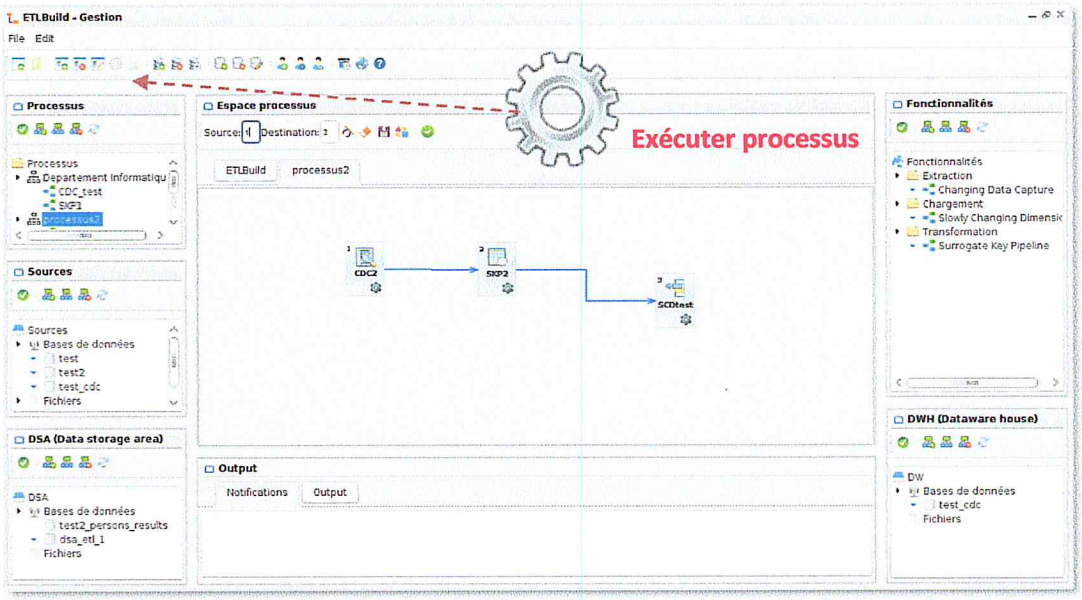


Figure 33 : Environnement d'un processus ETL

Après avoir détaillé les modules de la plateforme, nous pourrons présenter l'architecture détaillée de celle-ci (figure 34).

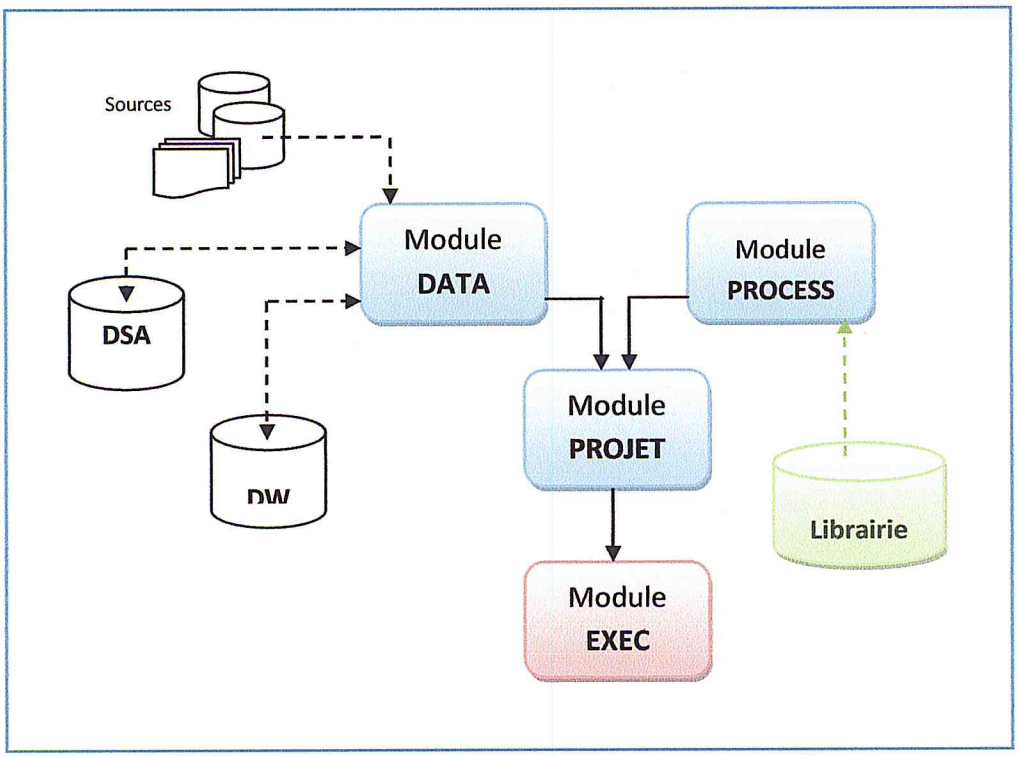


Figure 34: Architecture détaillée de la plateforme

2.3 Stockage des données système:

La plateforme permet le stockage des éléments constituant un projet dans un espace particulier. Celui-ci inclut les zones de stockage : (i) DSA, (ii) processus et (iii) catalogue système.

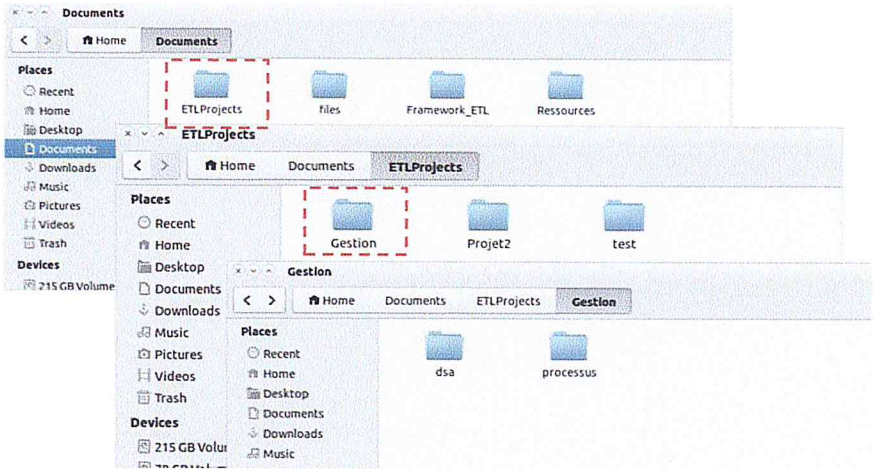


Figure 35: zones de stockage des projets ETL

- Stockage des processus : les processus ETL seront stockés dans des fichiers en format XML (Figure 37). Tous les concepts de ce dernier seront représentés sous forme de balises (Figure 38). Toute modification opérée dans le processus sera prise en considération.

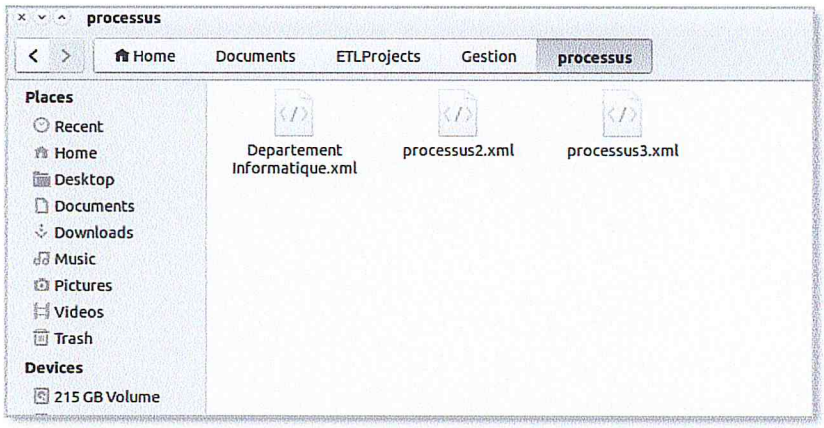


Figure 36: Zone de stockage des processus ETL

```
<?xml version="1.0" encoding="UTF-8"?>
<ETL>
        <functionality>
            <id>1</id>
            <iconame>f1.png</iconame>
            <name>fonction1</name>
            <abreviation>f1</abreviation>
            <description>......</description>
            <input>i1</input>
            <input>i2</input>
            <output>o1</output>
        </functionality>
        <functionality>
            <id>2</id>
            <iconame>f2.png</iconame>
            <name>fonction2</name>
            <abreviation>f2</abreviation>
            <description>......</description>
            <input>o1</input>
            <input>i3</input>
            <output>o2</output>
        </functionality>
</ETL>
```

Figure 37: Processus ETL stocké sous format XML

- **Zone DSA :** toutes les informations intermédiaires obtenues lors de l'utilisation du système qui seront stockées dans la zone DSA propre à chaque projet (Figure 36).

- **Catalogue du système :**

Pour stocker tous les objets relatifs à un projet, la plateforme exploite un catalogue système, qui représente la base de données nommée *ETL_Catalogue* dont le schéma est représenté dans la section 4.1 – chapitre V Figure 19.

C'est une base de données qui centralise toutes les informations sur les objets (zones de stockage, processus …) relatifs à un projet ETL.

3. Description de fonctionnement de la plateforme « ETLBuild » :

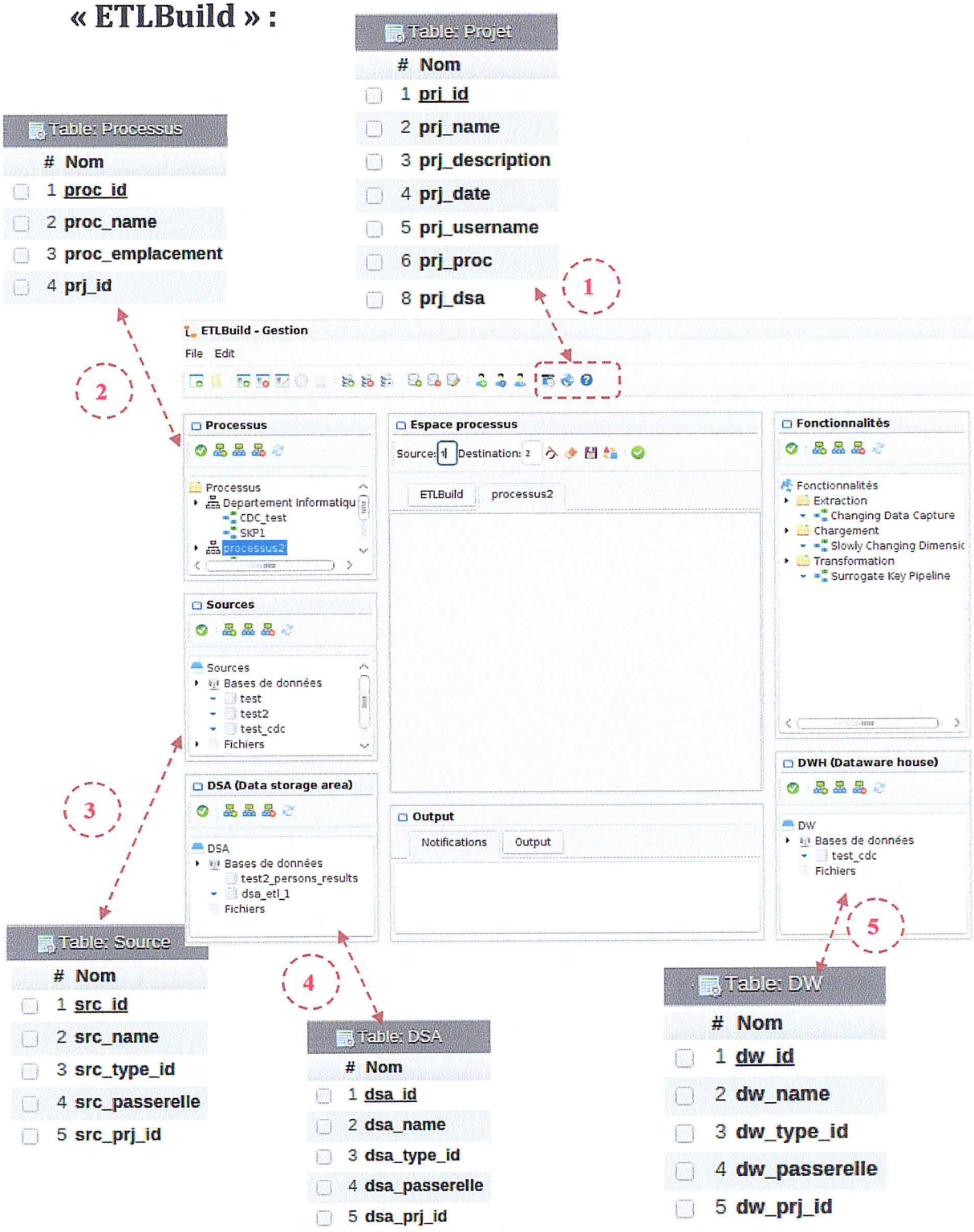


Figure 38: Fonctionnement de la plateforme

Toutes les données relatives aux objets du système sont stockées dans le catalogue système :

Lorsqu'un nouveau projet ETL est crée, le module « PROJET » stocke toutes ses informations dans la table correspondante nommée « Projet » dans ETL_catalogue (1).

A l'ouverture d'un projet existant, ce module devra accéder au catalogue pour charger touts les éléments constituants. Aussi, lorsque des modifications sont opérées dans le schéma de données (processus, sources, DSA, DW), celles-ci doivent être prises en considération dans ETL_catalogue.
De même, lorsqu'un utilisateur effectue une opération; de création, modification ou suppression, sur un objet « Source », « Processus », « DSA » ou « DW », le système mis à jour les données dans les tables correspondantes (2), (3), (4), (5).

4. Présentation de l'environnement de développement

4.1 Environnement de travail :

4.1.1 Langage de programmation Java :

Le langage **Java** est un langage de programmation informatique orienté objet, créé par James Gosling et Patrick Naughton employés de Sun Microsystems présenté en 1995 Le langage Java a la particularité d'être multi-plateforme, c'est-à-dire que les logiciels développés sous ce langage sont utilisables sur plusieurs systèmes d'exploitation tels que Microsoft Windows, Mac OS, Linux ainsi que sur plusieurs appareils mobiles. [Miller, 2010]

Nous avons choisi le langage Java car il convient parfaitement à l'élaboration du projet de part le nombre important de possibilités en matière de programmation objet de programmation d'interfaces graphiques.

4.1.2 Netbeans :

Afin de programmer la plateforme avec le langage Java, nous avons choisi l'environnement de développement intégré **Netbeans EDI (Environnement de développement intégré)** qui permet entre autre de créer des projets en langage Java. Le choix de cette application vient de sa simplicité d'utilisation et du nombre important de possibilités proposées par cette application, par exemple la génération automatique des composants graphiques et les interfaces.

4.2 Observer pattern « MVC » :

En ingénierie logicielle, un design pattern (ou patron de conception) est une solution «abstraite» pour des problèmes communs dans l'architecture logicielle.

Ce n'est pas une architecture utilisable directement dans le code, mais plutôt une simple description (ou Template) qui peut être utilisée dans différentes situations.

Techniquement, les design patterns sont des solutions orientées objets qui montrent des relations et des interactions entre des classes ou objets sans les spécifier concrètement.

Le patron de conception « Observateur » est utilisé dans le cas où on a un « Objet A » (subject) qui sera modifié lors de l'exécution d'un programme. A chaque modification il doit notifier d'autres objets (observers) pour leur indiquer ce changement. Donc ces « observers » n'ont pas besoin de perpétuellement demander l'état de cet « Object A » (l'observable), ils doivent tout simplement s'inscrire à ses notifications.

Diagramme de classes :

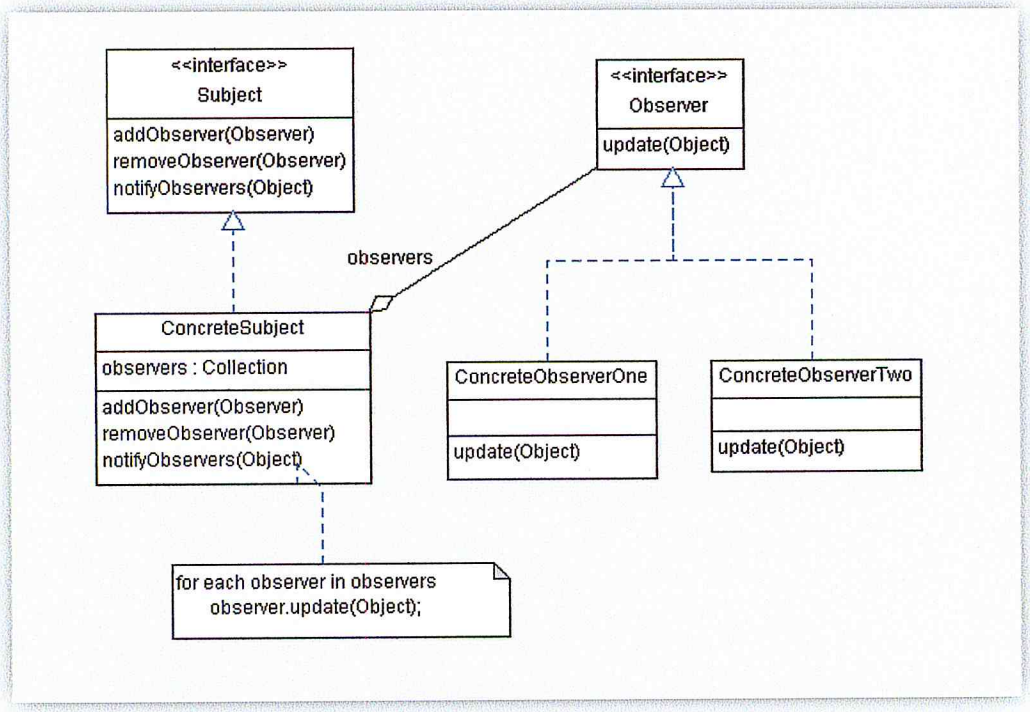


Figure 39: Diagramme de classe du patron Observateur

ConcreteSubject : C'est l'objet observé. Celui qui va être modifié au cours de l'exécution du programme (modification d'état : attributs).

- *addObserver (...)* : permet d'ajouter un objet « observer » à la collection des objets qui veulent être notifiés.
- *removeObserver (...)* : permet de supprimer un « observer » depuis cette collection et donc le désinscrire.

- *notifierObservers (...)* : c'est la méthode la plus importante de ce pattern. Elle doit être appelée à chaque fois qu'il y a eu une modification sur l'observable. Elle permet donc de passer sur tous les objets qui souhaitent être notifiés et leurs fournir les informations pour lesquelles ils se sont inscrits.

ConcreteObserverOne : Un objet parmi les objets qui se sont inscrits pour être notifié lors du changement du « ConcreteSubject »

- *Update (...)* : c'est la méthode qui sera appelée lors de la notification, pour donner la nouvelle information venue du "ConcreteSubject" vers ce "ConcreteObserverOne".

5. Présentation et tests :

Prenons l'exemple d'un entrepot de données nommé « DW_pubs », il contient la table de faits « Ventes » qui permet de calculer le chiffre d'affaire CA par :

- Titles (title_id, title, type)
- Publishers (pub_name, city, country)
- Stores (stor_name, city)
- Période (Mois, Année)

Figure 40 : Base de données « DW_pubs »

On va appliquer l'extraction par la fonction CDC sur la dimension « Store » afin de capturer les changements sur données.

Version précédente :

	NK	SK	store_name	city
Modifier Copier Effacer	1	1	dzetoile	Alger

Figure 41 : Snapshot « Store »

Version récente :

NK	SK	state_name	city	capture	designation_capture
1	1	dzetoile	Blida	2	modification
2	2	ramacontruction	Alger	1	insertion

Figure 42 : version récente de « Store »

Déroulement de l'exécution :

(1) L'utilisateur s'authentifie pour accéder à la plateforme ;

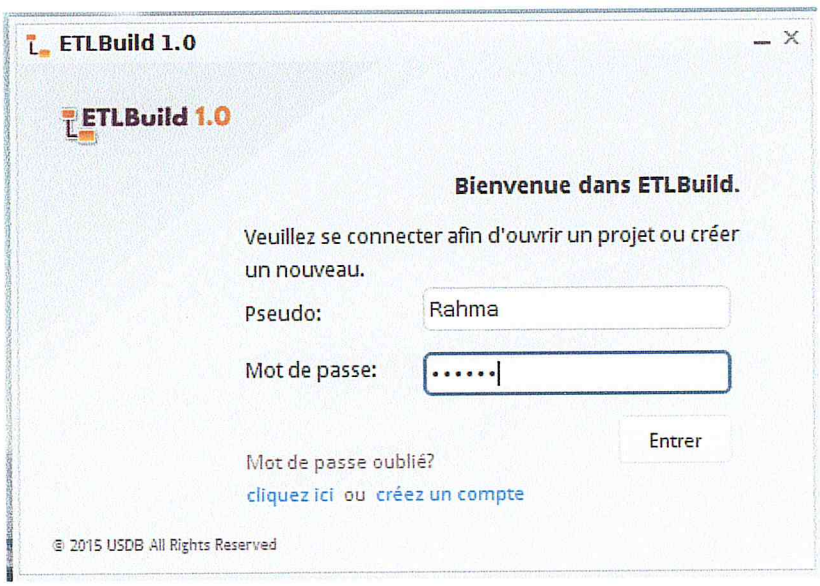


Figure 43 : Interface de connexion à la plateforme

(2) Il ouvre un projet existant nommé « Gestion » :

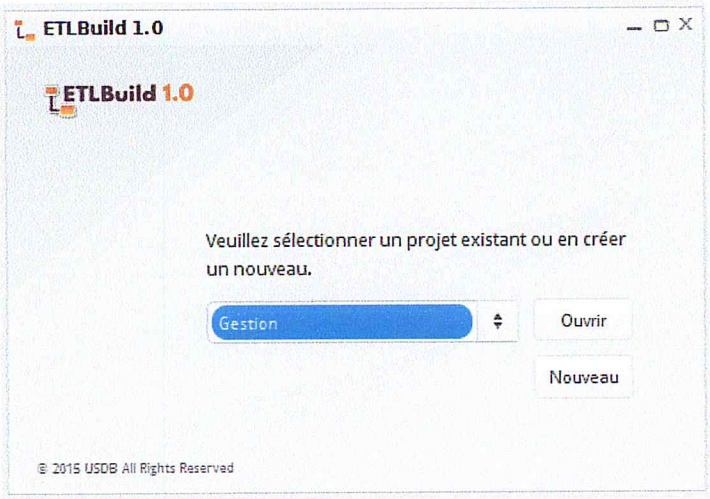


Figure 44 : Interface d'ouverture d'un projet.

(3) Il crée un processus nommé « processus 2 » et ajoute deux fonctions : CDC et SCD :

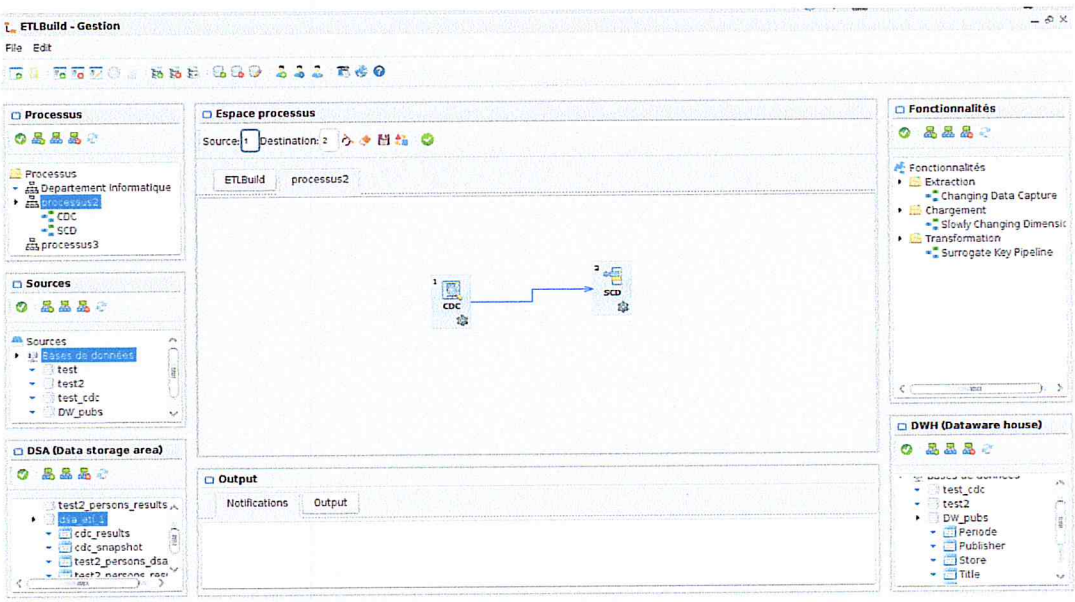


Figure 45 : Interface principale de la plateforme

(4) L'utilisateur configure les inputs et les outputs de la fonction « CDC » :

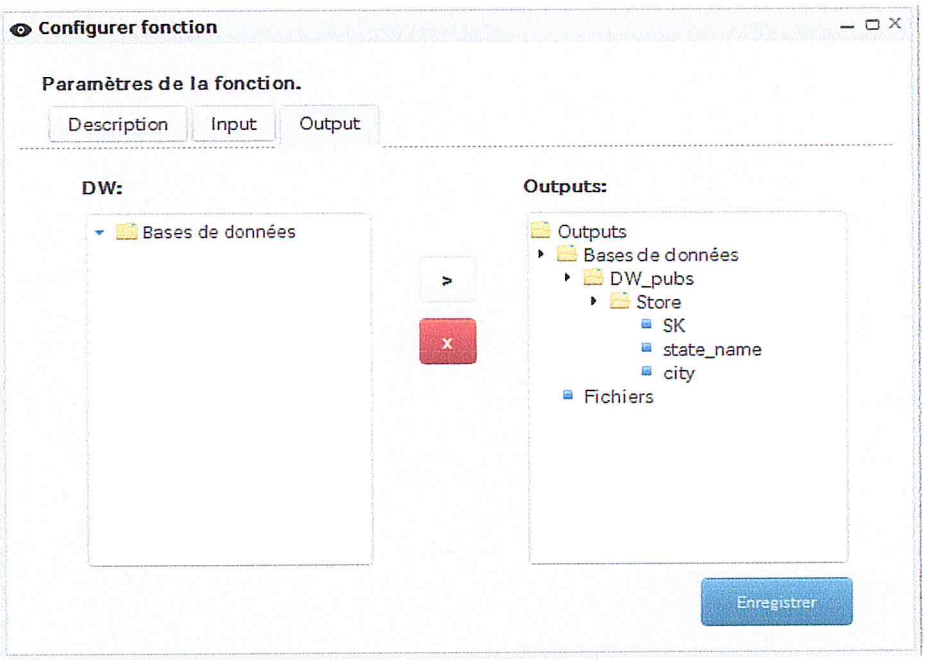


Figure 46 : Interface de paramétrage d'une fonction ETL

- Source de données : table « Store ».
- Dsa: table « cdc_ snapshot ».

(5) Les résultats de l'exécution sont insérés dans la table « cdc_results_82 » :

NK	SK	state_name	city	capture	designation_capture
1	1	dzetoile	Blida	2	modification
2	2	ramacontruction	Alger	1	insertion

Figure 47 : table « cdc_snapshot_84 »

6. Conclusion

Nous avons présenté dans ce chapitre, les fonctionnalités de la plateforme ETLBuild ainsi que la librairie de fonctions ETL implémentée indépendamment et intégrée par la suite dans le système. Ce dernier permet à l'utilisateur final de construire et exécuter des processus d'ETL.

Conclusion générale

La phase d'intégration d'un système d'information décisionnel est basée sur un processus ETL (Extraction, Transformation et Chargement) considéré aujourd'hui comme étant le composant le plus important, le plus complexe et le plus coûteux dans le cadre d'un projet décisionnel. Il est chargé du nettoyage, filtrage, homogénéisation, intégration et de l'agrégation des données hétérogènes issues de diverses sources et ce pour des fins d'analyse et d'aide à la décision.

Face à la complexité du processus ETL, ce travail consiste à définir celui-ci grâce à des fonctions de base appelées fonctionnalités d'ETL. Dans ce contexte, il nous a été demandé de concevoir et de mettre en œuvre une librairie de fonctions d'ETL sous forme d'un package (ensemble de classes java) qui sera par la suite intégrée dans une plateforme présentant les fonctionnalités les plus importantes pour le paramétrage et l'exécution de processus ETL.

Comme ce travail se situe dans un contexte décisionnel, nous avons tout d'abord exposé les fondamentaux sur les systèmes décisionnels, plus particulièrement, les entrepôts de données, les processus ETL, les environnements OLAP et les différents modèles de modélisation à un niveau conceptuel (schéma en étoile, schéma en flocons de neige et schéma en constellation) et logique (ROLAP, HOLAP et MOLAP). L'état de l'art nous a permis ainside mettre l'accent sur l'importance du processus ETL et ses différentes phases :Extraction (E), Transformation (T) et Chargement (L). Nous avons maitrisé les aspects de l'entreposage de données et découvert la complexité du processus ETL notamment en termes de flux de données et de tâches d'ETL.

La seconde étape du travail concerne la librairie d'ETL. Nous avons défini, conçu et implémenté des fonctions sous forme de classes java regroupées dans un package ETL. Nous avons choisi d'étudier les fonctions les plus courantes à

savoir Changing Data Capture (CDC), Slowly Changing Dimension (SCD) et Surrogate Key Pipeline (SKP).

Dans la troisième étape, nous avons conçu et mis en œuvre une plateforme de conception et d'exécution de processus d'ETL organisée en plusieurs modules à savoir : (i) Projets : création de projets d'intégration, (ii) Data : gestion de toutes les données ayant trait au projet d'intégration, (iii) librairie d'ETL : défini dans la phase précédente, (iv) Processus : conception graphique des processus d'ETL et enfin (v) Exécution : module chargé de l'exécution de tous les processus configurés dans un projet.

Perspectives

Au terme de ce travail, nous considérons avoir acquis et maitrisé l'usage des principes fondamentaux de l'intégration des données, ce qui constitue d'ailleurs l'objectif visé, sans toutefois oublier d'apporter d'autres améliorations telles que :

- Extension de la librairie par d'autres fonctions d'ETL importantes a in de prendre en charge des processus d'ETL consistants et complexes.

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

- Proposer une modélisation et une structure pour la zone de préparation de données DSA pour améliorer les performances des tâches d'ETL et garantir une qualité de données meilleure.

- Enfin, améliorer l'environnement intégré de configuration de processus ETL, et plus précisément, offrir des fonctionnalités graphiques plus performantes pour faciliter à l'utilisateur la construction des processus ETL.

Bibliographie

(Alter, 1980) ALTER Steven. Decision Support System: Current Practices and Continuing Challenges. Massachusetts: Addison-Wesley Publishing Co., 1980, 316 p.

(Bruley, 2010) Michel Bruley. Système d'information décisionnel : à quoi cela sert-il ?, 2010, 37 p.

(Kelly, 1997) KELLY Sean. Data warehousing in action. Chichester, West Sussex, England: John Wiley & Sons, 1997, 334 p.

(Atigui, 2013) ATIGUI Faten. Approche dirigée par les modèles pour l'implantation et la réduction d'entrepôts de données. Toulouse 2013, 158p.

(Inmon, 1996) Bill Inmon. Building the Data Warehouse, volume 2nd edition. John Wiley and Sons, New York, 1996.

(Kimball, 1996) Ralph Kimball. The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley, 1996.

(Kimball, 2002) Ralph Kimball & Margy Ross. The Data Warehouse Toolkit - Second Edition - The Complete Guide to Dimensional Modeling. John Wiley, 2002.

(Vassiliadis, 2009) P. Vassiliadis. A Survey of Extract–Transform–Load Technology. 2009.

(Kimbal, 2004) Ralph Kimball & Joe Caserta. The Data Warehouse ETL Toolkit Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. Wiley, 2004.

(Giraldo, 2003) C. Reynaud & G. Giraldo. An application of the mediator approach to services over the web. *Special track "Data Integration in Engineering, Concurrent Engineering (CE'2003)*. July 2003.

(Xuan, 2006) Dung Xuan Nguyen. Intégration de bases de données hétérogènes par articulation à priori d'ontologies: application aux catalogues de composants industriels. 2008.

(Ploix, 2012) D.Ploix. Intégration de données multi-sources. 2012.

(Audibert, 2006) Laurent AUDIBERT. UML2 : de l'apprentissage à la pratique, 2009.

(Bala, 2013) Bala M. et Alimazighi Z. (2013). Modélisation de processus ETL dans un modèle MapReduce, Conférence Maghrébine sur les Avancées des Systèmes Décisionnels (ASD'13), p. 1–12, Marrakech, Maroc.

(Miller, 2010) Frederic P. Miller Java (Langage), 2010

(J-S, 2010) J-F. Desnos. Entrepôt de données – Introduction, 2010.

Web

(Grim, 2008) Yazid Grim, OLAP, les fondamentaux, 2008. URL : http://grim.developpez.com/articles/concepts/olap/

(Doucet, 2007) Anne Doucet. Intégration de données hétérogènes et réparties, 2007. URL : http://www-poleia.lip6.fr/~doucet/CoursBDIA/Cours4.pdf

(Braesch, 2003) Christian Braesch. Construction d'un entrepôt de données, 2003. URL: http://www.christian.braesch.fr/page/construction-dun-entrepot-de-donnees

(Doudoux, 2014) Jean-Michel Doudoux. Java et UML, 2014. URL: http://www.jmdoudoux.fr/java/dej/chap-uml.htm

Elisabeth METAIS. Systèmes informatiques – systèmes d'aide à la décision, Encyclopedia Universalis en ligne, URL: http://www.universalis.fr/encyclopedie/systemes-informatiques-systemes-d-aide-a-la-decision/

Les technologies OLAP. URL: http://hoffmannmathias.free.fr/documents/donnees/OLAP/Les%20Technologies(OLAP).htm

Manipuler les données XML et CSV. URL: http://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/_iles/ressources/pedagogiques/693/693-isn-opendata-seq2-eleve.pdf

Staging Area dans le processus ETL. URL: http://www.dwfacile.com/stag_or_not.htm

Ralph Kimball. Design Tip #63 Building a Change Data Capture System, 2005. URL: http://www.kimballgroup.com/2005/01/design-tip-63-building-a-change-data-capture-system/

Le processus ETL (Extracting – Transforming - Loading) : http://derwarehouse.blogspot.com/2015/01/le-processus-detl-extracting.html