UNIVERSITE SAAD DAHLEB DE BLIDA

Faculté des sciences

Département Informatique

MEMOIRE DE MASTER

EN INFORMATIQUE

Option : Ingénierie Des Logiciels

Conception et réalisation d'un système générique de collecte automatique de données

Réalisé par

BENAYACHE Samira

Encadré par

Mme SEMAR Kahina

Mme GHEBGHOUB Yasmina

2018 / 2019

ملخـــــص

اليوم، تقوم الشركات بمعالجة المزيد والمزيد من البيانات. غالبًا ما يتم توزيع هذه البيانات في أنظمة مختلفة غير متجانسة. في مشروع صنع القرار، يعد جمع هذه البيانات ضروريًا لاستغلالها بشكل فعال.

تمثل مرحلة الاستخراج هذه مرحلة تحول مسبق (وفقًا لمتخصصي الوسط) تقريبًا ثلاثة أرباع مشروع إنشاء مستودع البيانات.

إن عملية ETL هي سلسلة من العمليات اللازمة لتشغيل قاعدة بيانات مستهدفة مع توزيع البيانات في قواعد بيانات مصدر غير متجانسة مختلفة: الاستخراج، والتحول، والتحميل، وتنفيذ عملية ETL في المؤسسة مهمة ثقيلة ومعقدة.

يهدف العمل المقدم في هذه الرسالة إلى تصميم وتنفيذ نظام عام لجمع البيانات تلقائيًا من مصادر البيانات غير المتجانسة (نموذج أو ملف Excel أو قاعدة بيانات واحدة أو أكثر) باستخدام ملفات XML كوسيط الصرف. هذا النظام ليس منافسًا للحلول الثقيلة للسوق ولكنه يقدم نفسه كبديل خفيف ويمكن الوصول إليه من حيث التعلم والإعداد والاستخدام.

الكلمات المفتاحية: مستودع البيانات، الاستخراج، قاعدة البيانات، عدم التجانس، ETL، XML

Résumé

Aujourd'hui, les entreprises traitent des volumes de données de plus en plus importants. Ces données sont souvent réparties dans différents systèmes hétérogènes. Dans un projet décisionnel la collecte de ces données est nécessaire afin de les exploiter efficacement.

Cette phase d'extraction est de transformation préalable représente (selon les spécialistes du milieu) à peu près les trois quarts du projet de création d'un Data Warehouse.

Un processus ETL est une suite d'opérations nécessaires à l'alimentation d'une base cible avec des données réparties dans différentes bases sources hétérogènes : Extraction, Transformation, Chargement, La mise en place de processus ETL dans l'entreprise est une tâche lourde et complexe.

Le travail présenté dans ce mémoire a pour objectif la conception et l'implémentation d'un système générique de collecte automatique de données à partir des sources de données hétérogènes (un formulaire, un fichier Excel, une ou plusieurs bases de données) en utilisant des fichiers XML comme support d'échange. Ce système n'est pas un concurrent des solutions lourdes du marché mais se présente comme une alternative légère et plus accessible en termes d'apprentissage, de mise en place et d'utilisation.

Mots-clés : Data Warehouse, Extraction, Base de données, Hétérogénéité, ETL, XML.

Abstract

Today, companies are processing more and more data. These data are often distributed in different heterogeneous systems. In a decision-making project the collection of these data is necessary in order to exploit them effectively.

This phase of extraction is of prior transformation represents (according to the specialists of the environment) about three quarters of the project of creation of a Data Warehouse.

An ETL process is a series of operations needed to power a target database with data spread across different heterogeneous source databases: Extraction, Transformation, Loading, ETL process setup in the enterprise is a heavy task and complex.

The work presented in this thesis aims to design and implement a generic system for automatic data collection from heterogeneous data sources (a form, an Excel file, one or more databases) using XML files as exchange medium. This system is not a competitor of the heavy solutions of the market but presents itself as a light and more accessible alternative in terms of learning, setting up and use.

Keywords: Data Warehouse, Extraction, Database, Heterogeneity, ETL, XML.

Dédicaces

Je dédie ce travail à toutes les personnes qui m'ont soutenu durant les moments difficiles, en particulier :

A mes parents, les deux personnes qui me sont les plus chers aux mondes, qui ont tout sacrifié pour l'éducation et le bonheur de leurs enfants et qui m'ont permis d'en arriver là, merci pour tout et que dieu vous garde ;

A mon mari Abderrazak ;

A mes enfants : Amir, Mohamed Assil , Adem et Dina ;

A mes sœurs : Sihem, Soumia et mes frères Sami, Samir et Sofiane ;

A tous ceux que j'aime tant et que je n'ai pas cités .

REMERCIEMENTS

Je remercie Dieu le tout puissant qui m'a donné le courage et la volonté pour réaliser ce modeste travail.

Je tiens à remercier mon encadreur au CDTA Mme.SEMAR, pour sa patience infinie, sa disponibilité et surtout ses conseils.

Je remercie, aussi, Mme GHEBROUB ma promotrice à Saad Dahleb Blida, pour son aide précieuse, ses efforts et ses conseils.

Je remercie Mme TARABET pour ces critiques constructives qui m'ont permis d'améliorer ce mémoire.

Je remercie également tous les membres du jury pour m'avoir honoré par leur présence et pour avoir accepté d'évaluer mon travail.

Enfin je tiens à remercier tous mes enseignants pour toutes les connaissances qu'ils m'ont inculquées.

Table de matière

Liste des figures

Liste des tableaux

Introduction générale

Contexte général

C'est dans un environnement fortement complexe et hautement concurrentiel qu'évolue la majeure partie, si ce n'est la totalité, des entreprises. Ce climat de forte concurrence exige de ces entreprises une surveillance très étroite du marché afin de ne pas se laisser distancer par les concurrents et cela en répondant, le plus rapidement possible, aux attentes du marché, de leur clientèle et de leurs partenaires.

Pour se faire, les dirigeants de l'entreprise, quelque en soit d'ailleurs le domaine d'activité, doivent être en mesure de mener à bien les missions qui leur incombent en la matière. Ils devront prendre notamment les décisions les plus opportunes. Ces décisions, qui influeront grandement sur la stratégie de l'entreprise et donc sur son devenir, ne doivent pas être prises ni à la légère, ni de manière trop hâtive, compte tenu de leurs conséquences sur la survie de l'entreprise. Il s'agit de prendre des décisions fondées, basées sur des informations claires, fiables et pertinentes. Le problème est de savoir donc comment identifier et présenter ces informations à qui de droit, sachant par ailleurs que les entreprises croulent d'une part sous une masse considérable de données et que d'autre part les systèmes opérationnels « transactionnels » s'avèrent limités, voire inaptes à fournir de telles informations et constituer par la même un support appréciable à la prise de décision.

C'est dans ce contexte que les « systèmes décisionnels » ont vu le jour. Ils offrent aux décideurs des informations de qualité sur lesquelles ils pourront s'appuyer pour arrêter leurs choix décisionnels. Pour se faire, ces systèmes utilisent un large éventail de technologies et de méthodes, dont les « entrepôts de données » (Data Warehouse) représentent l'élément principal et incontournable pour la mise en place d'un bon système décisionnel.

Les entrepôts de données ont été conçus pour l'aide à la décision. Ils intègrent les informations en provenance des différents systèmes transactionnels de l'entreprise. L'ensemble des données, y compris leur historique, est utilisé pour faire des calculs prévisionnels, des statistiques ou pour établir des stratégies de développement et d'analyses des tendances.

La collecte de données de ces systèmes est un problème complexe. C'est pourtant une tâche que les entreprises peuvent difficilement éviter si elles veulent mettre en route de nouvelles applications ou réorganiser le système d'information existant pour une meilleure productivité. En

effet, les sources sont souvent hétérogènes car elles ont été définies indépendamment les unes des autres. Le processus de collecte de données doit donc permettre de traiter des sources qui ont des modèles de données et/ou des schémas différents.

Dans le cadre de ce stage, nous nous intéressons particulièrement sur le problème de collecte de données à partir de sources hétérogènes (bases de données relationnelles, fichiers Excel, de formulaires de saisies manuelles), en utilisant des fichiers XML comme support d'échange afin de rendre ce cadre plus générique.

Problématique

Les entreprises sont confrontées à une concurrence de plus en plus forte, des clients de plus en plus exigeants, dans un contexte organisationnel de plus en plus complexe. Pour faire face aux nouveaux enjeux économiques, l'entreprise doit anticiper. L'anticipation ne peut être efficace qu'en s'appuyant une information pertinente. Mais généralement, les données sont non organisées dans une perspective décisionnelle et éparpillées dans de multiples systèmes hétérogènes. Il devient fondamental de les rassembler et de les homogénéiser afin de faciliter la prise de décision.

La phase de la collecte des données et de leur transformation était souvent sous-estimée. C'est peut-être là une des principales explications des échecs de réalisations et des très nombreux dépassements de budget. Pourtant cette phase d'extraction est de transformation préalable représente (selon les spécialistes du milieu) à peu près les trois quarts du projet de création d'un Data Warehouse. **[9]**

Dans un tel contexte, il est souvent nécessaire de mettre en place un système générique de collecte automatique de données à partir des sources de données hétérogènes (un formulaire, un fichier, une ou plusieurs bases de données), Le problème général abordé par notre mémoire concerne la méthode générique de collecte de données hétérogènes avec l'objectif d'offrir une vue homogène de l'information d'une manière satisfaisante.

Objectifs du travail

Le travail réalisé dans le cadre de ce projet consiste à rédiger un état de l'art sur les systèmes décisionnels en se basent sur l'outil ETL (Extraction, Transformation, Loading) puis à faire une présentation et une comparaison d'un ensemble de travaux destinés à la phase de conception d'ETL, afin d'adopter une approche d'extraction de données à partir de plusieurs

sources hétérogènes. On a comme objectif aussi de concevoir et de mettre en œuvre un système de collecte de données à partir d'un formulaire de saisie, un fichier Excel et une base de données et d'évaluer les performances du système proposé.

Organisation du mémoire

Pour mener à bien notre recherche, nous avons organisé notre travail comme suite :

Partie I : Etat de l'art.

— Chapitre 1 : Présentera un état de l'art sur les systèmes décisionnels.

— Chapitre 2 : Sera consacré à la présentation d'un ensemble de travaux connexes afin de trouver une approche à adoptée.

Partie II : Contribution et réalisation

— Chapitre 3 : Portera sur la présentation du fonctionnement général du système générique de collecte de données ainsi sur la conception de la solution.

— Chapitre 4 : Décrit le déploiement de la solution.

Ce mémoire sera finalisé par une conclusion générale afin de synthétiser le travail réalisé et de citer les perspectives du projet.

Partie I :

Etat de l’art

Chapitre I : Systèmes décisionnels

1- Introduction

Toutes les entreprises du monde disposent d'une masse de données plus ou moins considérable. Ces informations proviennent soit de sources internes (générées par leurs systèmes opérationnels au fil des activités journalières), ou bien de sources externes (web, partenaire, .. etc.).

Cette surabondance de données, et l'impossibilité des systèmes opérationnels de les exploiter à des fins d'analyse conduit, inévitablement, l'entreprise à se tourner vers une nouvelle informatique dite décisionnelle qui met l'accent sur la compréhension de l'environnement de l'entreprise et l'exploitation de ces données à bon escient.

En effet, les décideurs de l'entreprise ont besoin d'avoir une meilleure vision de leur environnement et de son évolution, ainsi, que des informations auxquelles ils peuvent se fier. Cela ne peut se faire qu'en mettant en place des indicateurs « business » clairs et pertinents permettant la sauvegarde, l'utilisation de la mémoire de l'entreprise et offrant à ses décideurs la possibilité de se reporter à ces indicateurs pour une bonne prise de décision.

Le « Data Warehouse », « Entrepôt de données » en français, constitue, dans ces conditions, une structure informatique et une fondation des plus incontournables pour la mise en place d'applications décisionnelles.

Le concept de Data Warehouse, tel que connu aujourd'hui, est apparu pour la première fois en 1980 ; l'idée consistait alors à réaliser une base de données destinée exclusivement au processus décisionnel. Les nouveaux besoins de l'entreprise, les quantités importantes de données produites par les systèmes opérationnels et l'apparition des technologies aptes à sa mise en œuvre ont contribué à l'apparition du concept « Data Warehouse » comme support aux systèmes décisionnels.

2- Les systèmes décisionnels

La raison d'être d'un entrepôt de données, comme évoqué précédemment, est la mise en place d'une informatique décisionnelle au sein de l'entreprise. Pour cela il serait assez intéressant de définir quelques concepts clés autour du décisionnel.

Afin de mieux comprendre la finalité des systèmes décisionnels, nous nous devons de les placer dans leurs contextes et rappeler ce qu'est un système d'information.

« Le système d'information est l'ensemble des méthodes et moyens de recueil de contrôle et de distribution des informations nécessaires à l'exercice de l'activité en tout point de l'organisation. Il a pour fonction de produire et de mémoriser les informations, de l'activité du système opérant (système opérationnel), puis de les mettre à disposition du système de décision (système de pilotage) » **[10]**.

Les différences qui existent entre le système de pilotage et le système opérationnel, du point de vue fonctionnel ou des tâches à effectuer, conduit à l'apparition des *« systèmes d'information décisionnels » (S.I.D.)*. Ces différences seront clairement illustrées un peu plus loin dans notre document.

Les origines des SID remontent au début de l'informatique et des systèmes d'information qui ont, tous deux, connu une grande et complexe évolution liée notamment à la technologie. Cette évolution se poursuit à ce jour **[11]**.

2-1- La place du décisionnel dans l'entreprise :

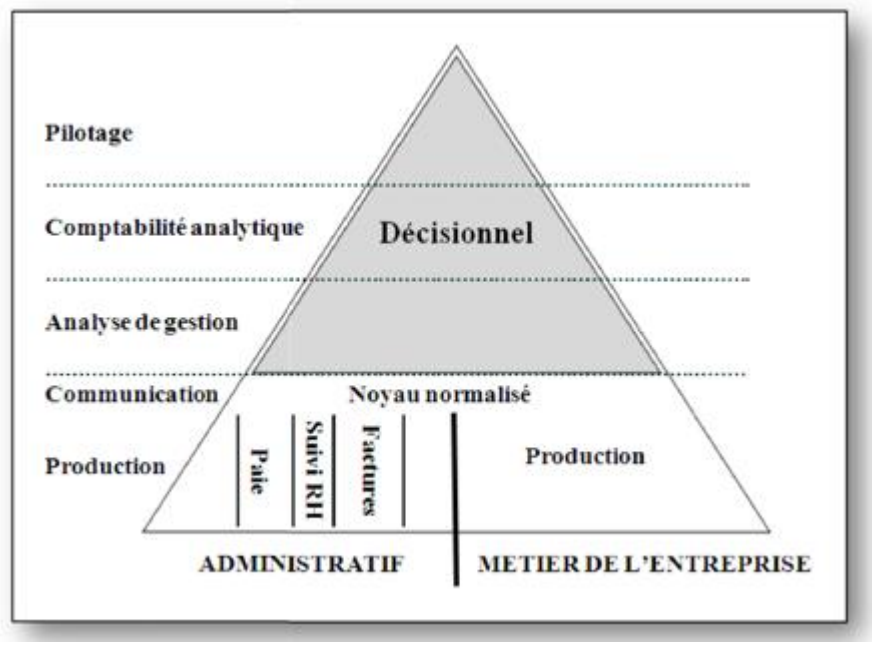


Figure 1.1 : Le décisionnel au sein du Système d'information **[23]**.

La figure ci-dessus illustre parfaitement la place qui revient au décisionnel au sein d'une entreprise. Cette place, comprend plusieurs fonctions clés de l'entreprise. Les finalités décisionnelles, étant différentes selon le poste et la fonction occupée, ont pour but d'engendrer plusieurs composantes

2-2- Les différentes composantes du décisionnel

En relation étroite avec les nouvelles technologies de l'information et des télécommunications, le système décisionnel se manifeste à différents niveaux selon leurs utilités et leurs missions principales, comme illustré dans la figure suivante :

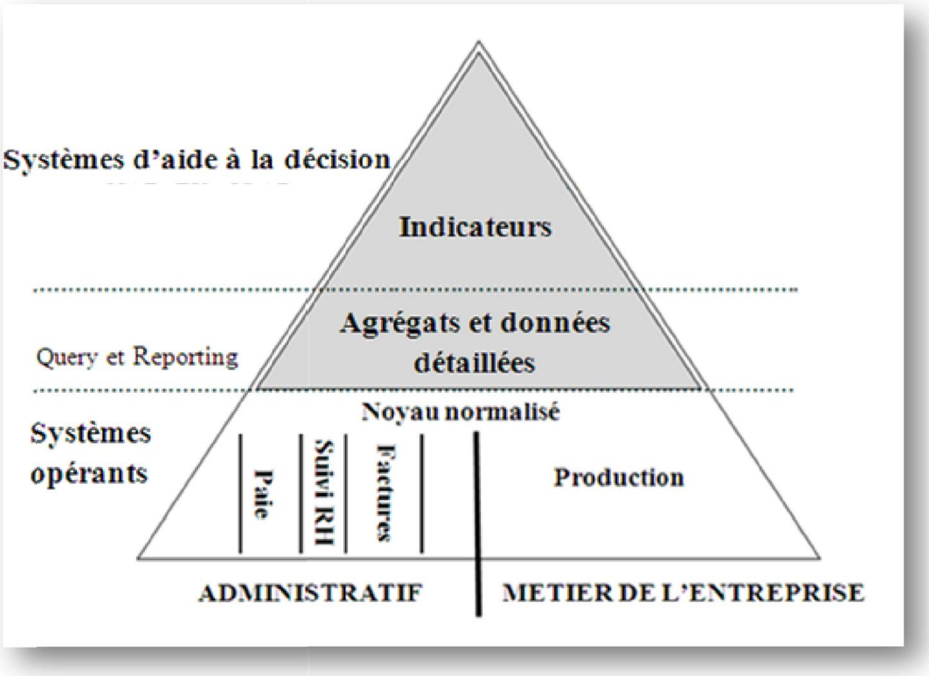


Figure 1.2 : Les différentes composantes du décisionnel **[23]**.

3- Décisionnel vs transactionnel

Le tableau suivant résume de façon non exhaustive les différences qu'il peut y avoir entre les systèmes transactionnels et les systèmes décisionnels selon les données et l'usage fait des systèmes.

Différence	Systèmes transactionnels	SID
par les données	Orienté applications	Orienté thèmes et sujets
	Situation instantanée	Situation historique
	Donnée détaillée et codées non redondantes	Informations agrégées cohérentes souvent avec redondance
	Données changeantes constamment	Informations stables et synchronisées dans le temps
	Pas de référentiel commun	Un référentiel unique
L' usage	Assure l'activité au quotidien	Permet l'analyse et la prise de décision
	Pour les opérationnels	Pour les décideurs
	Mises à jour et requêtes simples	Lecture unique et requêtes complexes transparentes
	Temps de réponse immédiats	Temps de réponse moins critiques
	Faibles volumes à chaque transaction	Large volume manipulé
	Conçu pour la mise à jour	Conçue pour l'extraction
	Usage maîtrisé	Usage aléatoire

Tableau 1.1 : Tableau comparatif entre les systèmes transactionnels et les systèmes décisionnels. **[26]**

Ces différences font ressortir la nécessité de mettre en place un système répondant aux besoins décisionnels. Ce système n'est rien d'autre que le « ***Data Warehouse*** ».

3-1- Qu'est-ce qu'un Data Warehouse

Bill Inmon définit le Data Warehouse, dans son livre considéré comme étant la référence dans le domaine "Building the Data Warehouse" **[12]** comme suit :

« Le Data Warehouse est une collection de données orientées sujet, intégrées, non volatiles et évolutives dans le temps, organisées pour le support d'un processus d'aide à la décision. »

Les paragraphes suivants illustrent les caractéristiques citées dans la définition d'Inmon.

Orienté sujet : le Data Warehouse est organisé autour des sujets majeurs de l'entreprise, contrairement à l'approche transactionnelle utilisée dans les systèmes opérationnels, qui sont conçus autour d'applications et de fonctions telles que : cartes bancaires, solvabilité client…, les Data Warehouse sont organisés autour de sujets majeurs de l'entreprise tels que : clientèle, ventes, produits…. Cette organisation affecte forcément la conception et l'implémentation des données contenues dans le Data Warehouse. Le contenu en données et en relations entre elles diffère aussi. Dans un système opérationnel, les données sont essentiellement destinées à satisfaire un processus fonctionnel et obéit à des règles de gestion, alors que celles d'un Data Warehouse sont destinées à un processus analytique.

Intégrée : le Data Warehouse va intégrer des données en provenance de différentes sources. Cela nécessite la gestion de toute incohérence.

Evolutives dans le temps : Dans un système décisionnel il est important de conserver les différentes valeurs d'une donnée, cela permet les comparaisons et le suivi de l'évolution des valeurs dans le temps, alors que dans un système opérationnel la valeur d'une donnée est simplement mise à jour. Dans un Data Warehouse chaque valeur est associée à un moment

« Every key structure in the data warehouse contains - implicitly or explicitly -an element of time » [13].

Non volatiles : c'est ce qui est, en quelque sorte la conséquence de l'historisation décrite précédemment. Une donnée dans un environnement opérationnel peut être mise à jour ou supprimée, de telles opérations n'existent pas dans un environnement Data Warehouse.

Organisées pour le support d'un processus d'aide à la décision : Les données du Data Warehouse sont organisées de manière à permettre l'exécution des processus d'aide à la décision (Reporting, Data Mining…).

3-2-Historique des Data Warehouse

L'origine du concept *« Data Warehouse » D.W* (entrepôt de données en français) remonte aux années 80, durant lesquelles un intérêt croissant au système décisionnel a vu le jour, dû essentiellement à l'émergence des SGBD relationnel et la simplicité du modèle relationnel et la puissance offerte par le langage SQL,

Au début, le Data Warehouse n'était rien d'autre qu'une copie des données du système opérationnel prise de façon périodique, dédiée à un environnement de support à la prise de décision. Ainsi, les données étaient extraites du système opérationnel, stockées dans une nouvelle base de données «concept d'infocentre », le motif principal étant de répondre aux requêtes des décideurs sans pour autant altérer les performances des systèmes opérationnels.

Le Data Warehouse, tel qu'on le connaît actuellement, n'est plus vu comme une copie ou un cumul de copies prises de façon périodique- des données du système opérationnel. Il est devenu une nouvelle source d'information, alimenté avec des données recueillies et consolidées des différentes sources internes et externes.

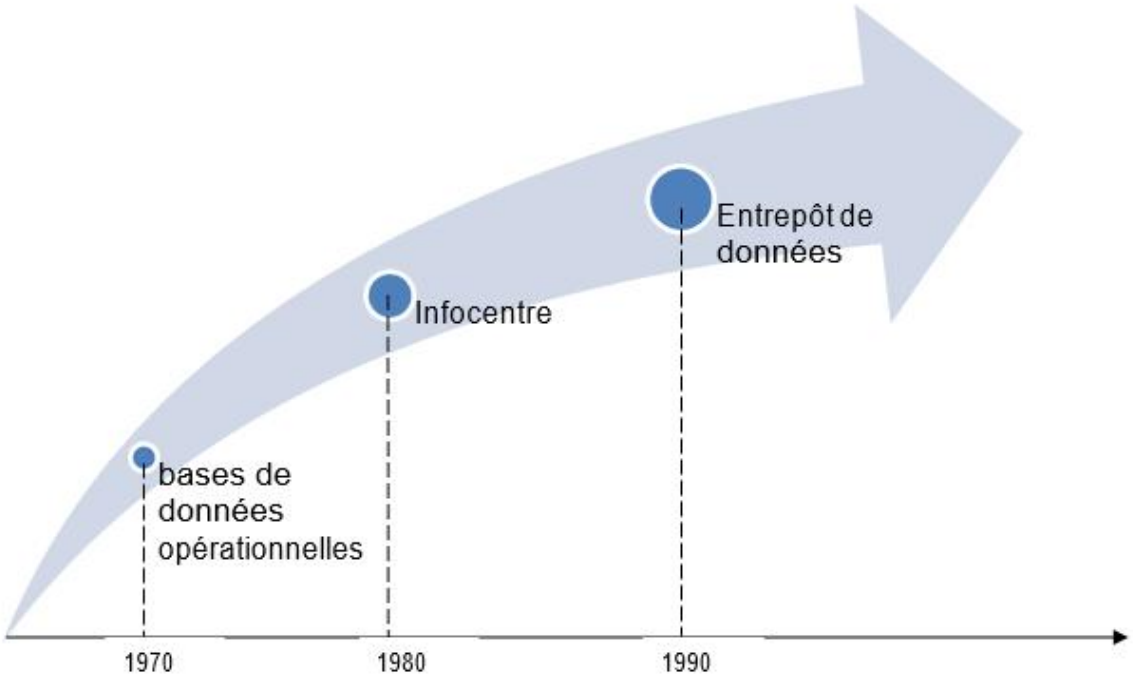


Figure 1.3 : évolution des bases de données décisionnelles**[26].**

3-3- Structure des données d'un Data Warehouse

Le Data Warehouse a une structure bien définie, selon différents niveaux d'agrégation et de détail des données. Cette structure est définie par Inmon [13] comme suit :

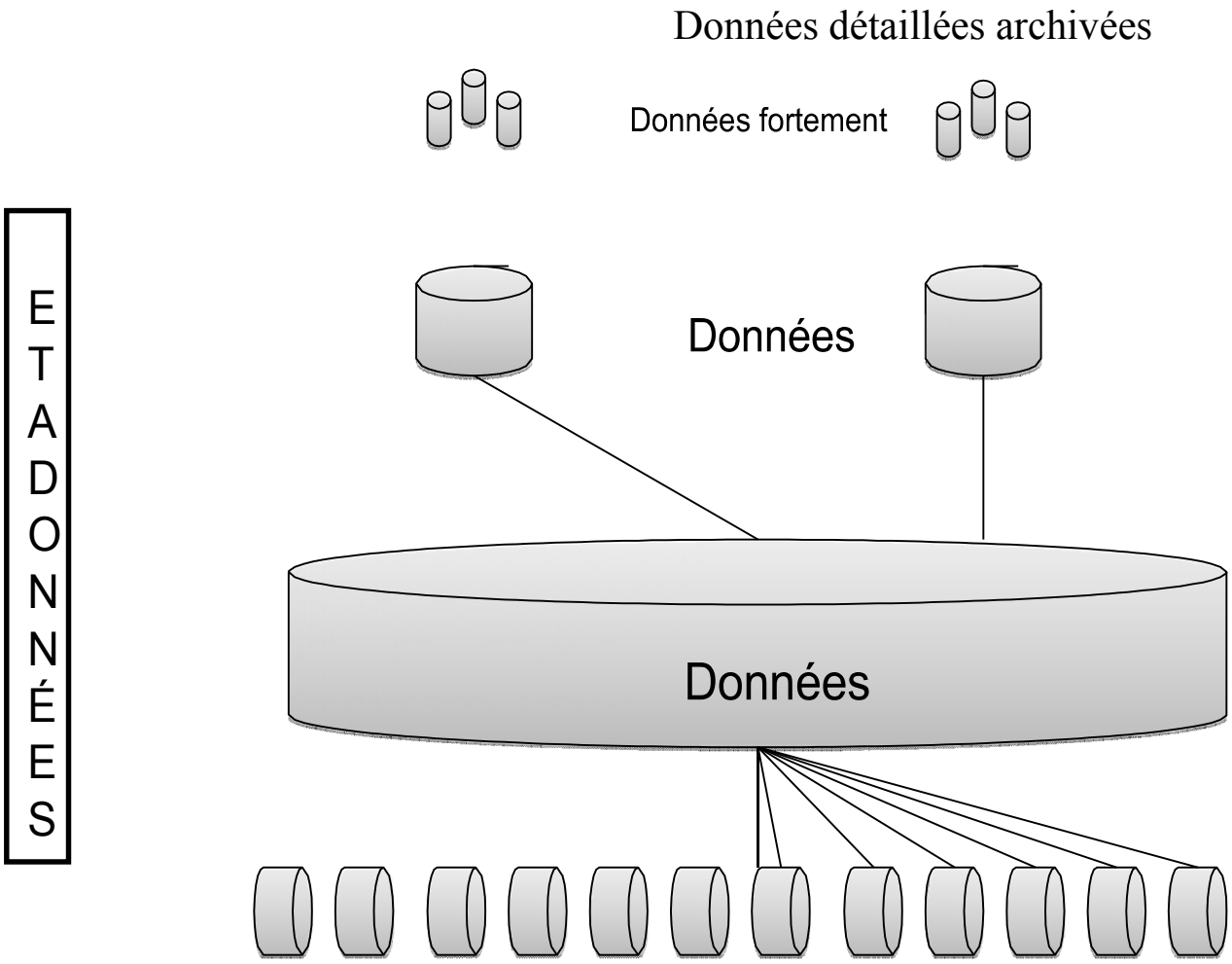


Figure I.4 : Structure des données d'un Data Warehouse**[26].**

Données détaillées : ce sont les données qui reflètent les événements les plus récents, fréquemment consultées, généralement volumineuses car elles sont d'un niveau détaillé.

Données détaillées archivées : anciennes données rarement sollicitées, généralement stockées dans un disque de stockage de masse, peu coûteux, à un même niveau de détail que les données détaillées.

Données agrégées : données agrégées à partir des données détaillées.

Données fortement agrégées : données agrégées à partir des données détaillées, à un niveau d'agrégation plus élevé que les données agrégées.

Meta données : ce sont les informations relatives à la structure des données, les méthodes d'agrégation et le lien entre les données opérationnelles et celles du Data Warehouse. Les métadonnées doivent renseigner sur :

- Le modèle de données,
- La structure des données telle qu'elle est vue par les développeurs,
- La structure des données telle qu'elle est vue par les utilisateurs,
- Les sources des données,
- Les transformations nécessaires,
- Suivi des alimentations,

3-4- Les éléments d'un Data Warehouse

L'environnement du Data Warehouse est constitué essentiellement de quatre composantes : les applications opérationnelles, la zone de préparation des données, la présentation des données et les outils d'accès aux données.

Les applications opérationnelles : ce sont les applications du système opérationnel de l'entreprise et dont la priorité est d'assurer le fonctionnement de ce dernier et sa performance. Ces applications sont extérieures au Data Warehouse.

Préparation des données : la préparation englobe tout ce qu'il y a entre les applications opérationnelles et la présentation des données. Elle est constituée d'un ensemble de processus appelé ETL, « Extract, transform and Load », les données sont extraites et stockées pour subir les transformations nécessaires avant leur chargement.

« Un point très important, dans l'aménagement d'un entrepôt de données, est d'interdire aux utilisateurs l'accès à la zone de préparation des données, qui ne fournit aucun service de requête ou de présentation » **[14]**.

Présentation des données : c'est l'entrepôt où les données sont organisées et stockées. Si les données de la zone de préparation sont interdites aux utilisateurs, la zone de présentation est tout ce que l'utilisateur voit et touche par le biais des outils d'accès.

3-5- Architecture d'un Data Warehouse

Après avoir exposé et défini chacun des éléments constituant l'environnement d'un Data Warehouse, il serait intéressant de connaitre le positionnement de ces éléments dans une architecture globale d'un Data Warehouse :

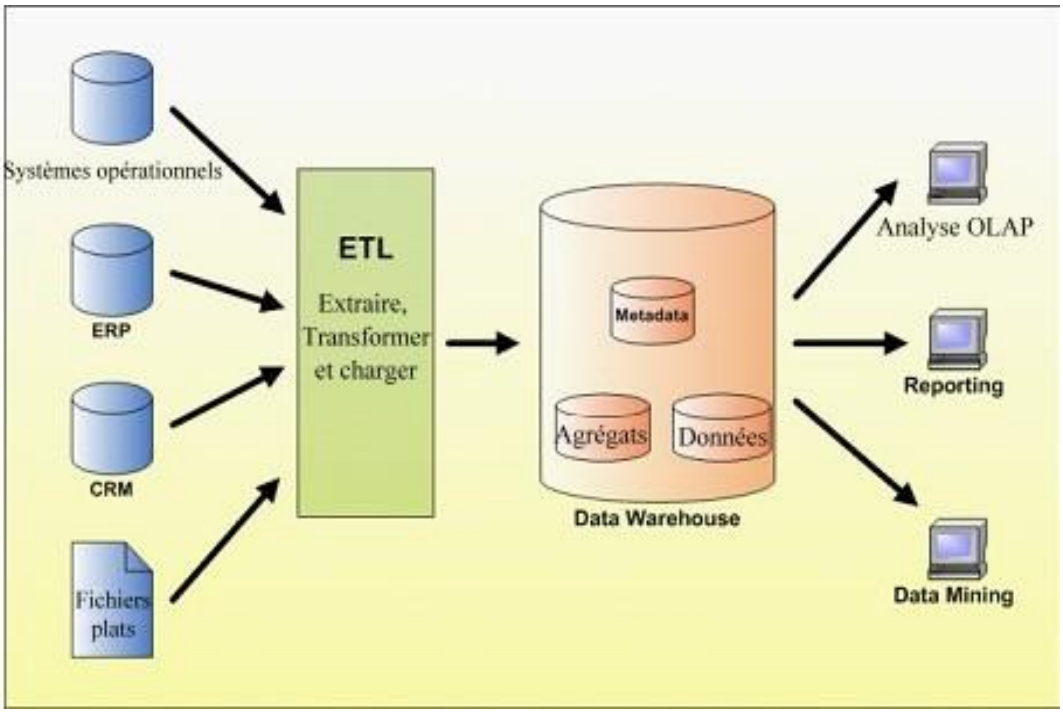


Figure 1.5 : Architecture globale d'un Data Warehouse **[25].**

3-6- Démarche de Construction d'un Data Warehouse

Plusieurs chercheurs ou équipes de recherche ont essayé de proposer des démarches pour la réalisation d'un projet Data Warehouse, ces démarches se croisent essentiellement dans les étapes suivantes :

- Modélisation et conception du Data Warehouse,
- Alimentation du Data Warehouse,
- Mise en œuvre du Data Warehouse,
- Administration et maintenance du Data Warehouse,

3-6-1-Modélisation et conception du Data Warehouse

Les deux approches les plus connues dans la conception des Data Warehouse sont :

- L'approche basée sur les besoins d'analyse,
- L'approche basée sur les sources de données,

Aucune des deux approches citées n'est ni parfaite, ni applicable à tous les cas. Toutes deux doivent être étudiées pour choisir celle qui s'adapte le mieux à notre cas.

Quelque soit l'approche adoptée pour la conception d'un Data Warehouse, la définition de celui-là reste la même. En étant un support d'aide à la décision, le Data Warehouse se base sur une architecture dimensionnelle.

- **Approche « Besoins d'analyse »**

Le contenu du Data Warehouse sera déterminé selon les besoins de l'utilisateur final. Cette approche est aussi appelée « approche descendante » (Top-Down Approach) et est illustrée par R. Kimball grâce à son cycle de vie dimensionnel comme suit :

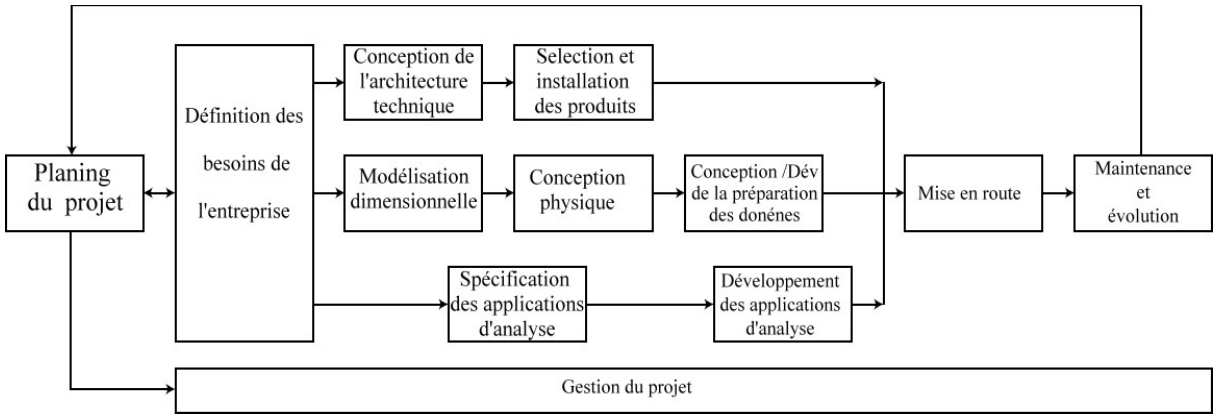


Figure 1.6 : illustration de l'approche « Besoins d'analyse » grâce au cycle de vie dimensionnel de Kimball **[15].**

- **Approche « Source de données »**

Le contenu du Data Warehouse est déterminé selon les sources de données. Cette approche est appelée : Approche ascendante (Bottom-up Approach).

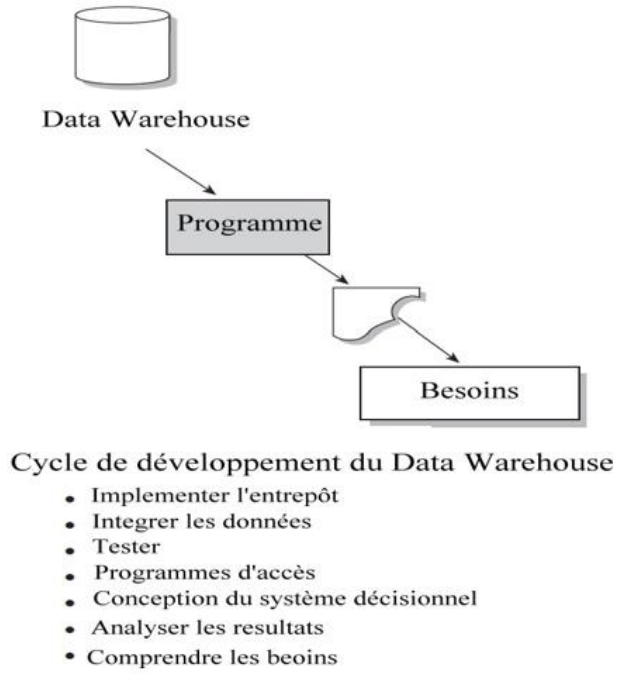


Figure 1.7 : Illustration de l'approche « Source de données » grâce au cycle de développement du DW de Inmon **[12]**.

Inmon considère que l'utilisateur ne peut jamais déterminer ses besoins dès le départ,

« *Donnez moi ce que je vous demande, et je vous direz ce dont j'ai vraiment besoin* » **[12]**., il considère que les besoins sont en constante évolution.

- **Approche mixte**

Une combinaison des deux approches appelée hybride ou mixte peut s'avérer efficace.

Elle prend en considération les sources de données et les besoins des utilisateurs.

Cette approche consiste à construire des schémas dimensionnels à partir des structures des données du système opérationnel, et les valider par rapport aux besoins analytiques. Cette approche cumule les avantages et quelques inconvénients des deux approches déjà citées, telles que la complexité des sources de données et la difficulté quant à la détermination des besoins analytiques.

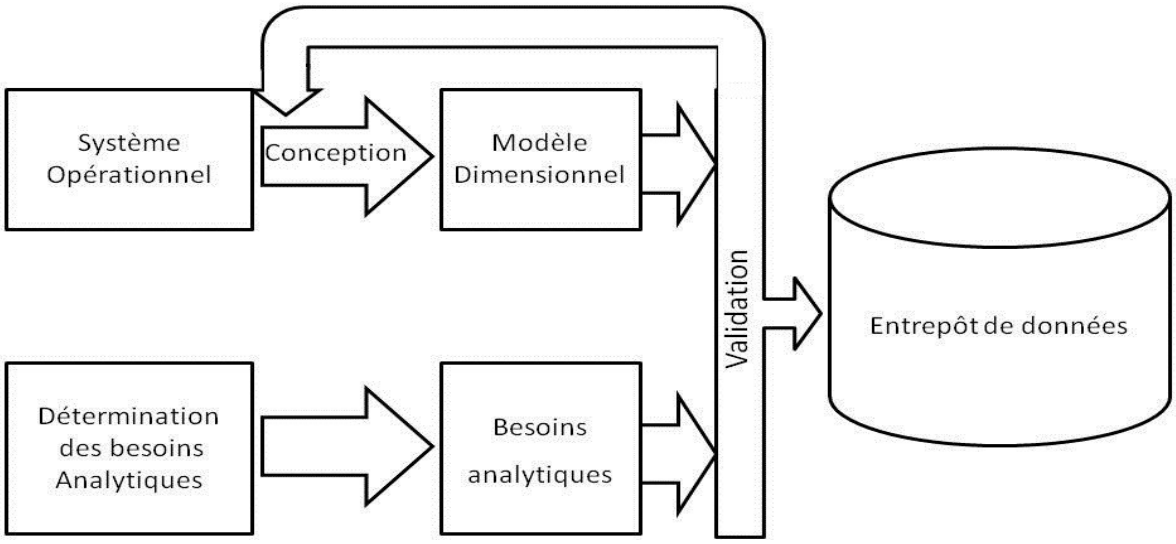


Figure 1.8 : Illustration de l'approche mixte**[12]**.

Cette étape aboutit à l'établissement du modèle dimensionnel validé du Data Warehouse. Ce modèle dimensionnel sera transformé en modèle physique, qui différera du modèle dimensionnel.

3-6-2- Alimentation du Data Warehouse

Une fois le Data Warehouse conçu, il faut l'alimenter et le charger en données. Cette alimentation (le plus souvent appelée processus ETL « **E**xtract-**T**ransform-**L**oad ») se déroule en 3 phases qui sont :

- Extraction des données primaires (issues par exemple des systèmes de production),
- Transformation des données,
- Le chargement des données traitées dans l'entrepôt de données,

Ces trois étapes décrivent une mécanique cyclique qui a pour but de garantir l'alimentation du Data Warehouse en données homogènes, propres et fiables.

3-6-3- Les phases de l'alimentation « E.T.L. »

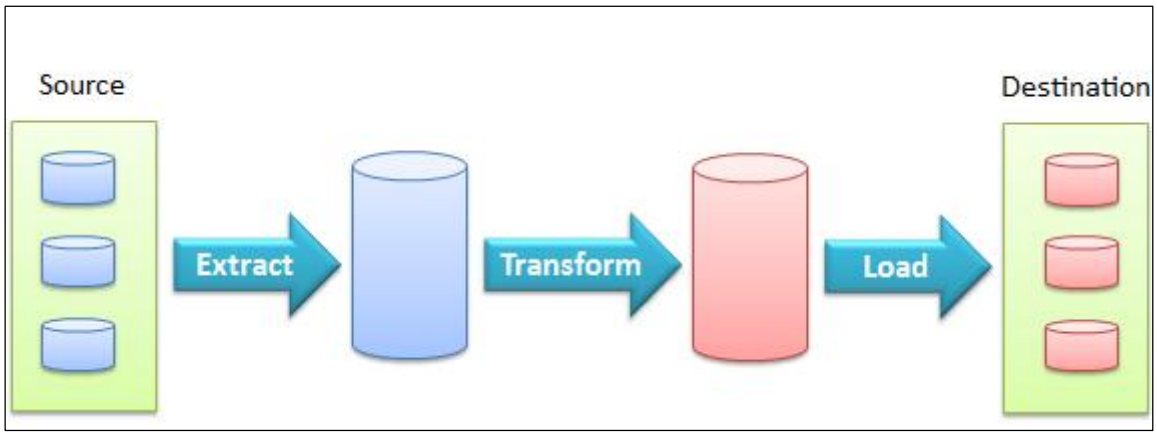


Figure 1.9 : Le processus ETL**[24]**

- **Extraction**

L'extraction permet d'extraire les données depuis les sources de données de production de l'entreprise (CRM, ERP, SBGD métier, …). L'extraction des données peut s'effectuer selon deux méthodes :

- à l'aide de triggers (méthode push) : ceux-ci sont déclenchés lors de modifications sur les bases sources et « poussent » les données modifiées vers l'outil ETL. Cette méthode nécessite d'intervenir au niveau des bases sources pour mettre en place ces triggers ce qui n'en fait pas la méthode la plus répandue ;

- à l'aide de requêtes (méthode pull) : le système ETL interroge les bases sources pour extraire les données. Cette méthode est la plus répandue car elle ne nécessite pas la modification des sources de données pour mettre en place les processus ETL.

L'extraction peut considérablement charger les systèmes de gestion des bases de données opérationnelles et perturber les applications OLTP.

- **Transformation**

La phase de transformation des données permet d'effectuer différentes opérations sur celles-ci afin de les concilier et d'obtenir un format qui respecte celui de la base cible. L'utilisateur définit des correspondances entre les schémas des bases sources et de la base cible. L'ETL s'appuie sur ces correspondances pour appliquer les transformations nécessaires sur les données et résoudre ainsi l'hétérogénéité sémantique.

La tendance actuelle est de proposer une interface graphique permettant de définir visuellement les correspondances.

Les transformations sont le cœur d'un ETL.

- **Chargement**

Le chargement permet de charger les données obtenues à l'issue de la phase de transformation vers la base cible (entrepôt de données ou autre). La réalimentation d'une base cible (ou rafraîchissement) est un procédé un peu différent de l'alimentation (ou chargement) initiale.

Les données anciennes (déjà présentes dans la base cible) ne sont jamais mises à jour avec les nouvelles données, mais archivées ou supprimées. L'insertion des nouvelles données ne doit pas modifier les données existantes. Une technique pour garantir la cohérence des données consiste à générer de manière automatique pour les nouvelles données les valeurs des clés des tables de la base cible au moment du chargement.

- **Nettoyage**

Selon les besoins, une phase de nettoyage de données est réalisée. Celle-ci est effectuée conjointement à la phase de transformation. Elle vise à améliorer la qualité des données à transférer vers la base cible.

Les données « à nettoyer » sont :

- les doublons,
- les données erronées,
- les valeurs manquantes,
- …

Les techniques employées sont :

- le rejet des données,
- le dédoublonnage,
- l'introduction de valeurs fixes, moyennes, …
- …

3-7- Volumétrie des données

Les processus ETL travaillent sur de gros volumes de données. Les outils doivent être capables de gérer cette importante volumétrie en s'appuyant sur des techniques particulières qui permettent d'améliorer leurs performances :

- la technique du streaming consiste à transférer les données en flux continu,
- la technique du parallélisme consiste à utiliser plusieurs processeurs ou threads pour une tâche.

3-7-1- Mode batch

Les processus ETL sont gourmands en termes de ressources. Ils fonctionnent la plupart du temps en mode batch : le processus ETL complet est programmé pour être exécuté à des moments où l'impact sera réduit pour les utilisateurs de ces mêmes ressources. Ces processus

ETL se déroulent souvent la nuit ou le week-end pour ne pas gêner les utilisateurs qui partagent ces mêmes ressources (réseau ou sources de données). Les données sont potentiellement moins soumises à modification durant ces périodes.

3-7-2-Cas de l'entrepôt de données

Les ETL utilisés pour l'alimentation d'entrepôts de données fournissent des fonctionnalités particulières adaptées aux caractéristiques spécifiques des entrepôts.

- **Marquage et datation des données**

Les données chargées doivent être marquées et datées. À tout moment le système peut identifier la provenance de données et connaître leur date de chargement.

- **Réalimentation**

La réalimentation (ou rafraîchissement) d'un entrepôt de données peut être effectuée de manière :

- complète : toutes les données sources sont chargées,

- incrémentale : seules sont chargées les nouvelles données sources par rapport au précédent chargement.

Dans le cas d'une réalimentation incrémentale, l'ETL doit être capable d'identifier les « nouvelles » données. Il existe, pour cela, plusieurs possibilités :

- si les données sources sont datées, le système peut se reposer sur ces informations,

- le système peut effectuer des comparaisons de données entre les sources et la cible,

- des triggers peuvent être mis en place au niveau des sources de données. Ceux-ci se déclenchent à la mise à jour des données et stockent ainsi les changements effectués dans un espace réservé,

- les logs de transactions peuvent être analysés afin de tracer les changements,

- …

- **Gestion des performances**

La gestion des performances est cruciale dans le domaine des entrepôts de données. Les techniques utilisées par les entrepôts de données, comme les index ou les vues persistantes, sont

prises en compte par les ETL qui effectuent la mise à jour de ces index et vues persistantes lors des phases d'alimentation.

- **Gestion des dysfonctionnements**

L'exécution d'un processus ETL peut être source de dysfonctionnements. L'ETL fournit des outils permettant de les gérer :

- reprise de processus lorsque celui-ci a été stoppé avant d'être terminé,

- identification et traitement les données rejetées lors de la phase de nettoyage ou d'alimentation de la cible.

4- Conclusion

Le concept « Data Warehouse » est apparu comme une réponse à des besoins grandissants dans le domaine décisionnel. Son adaptabilité et sa capacité de fournir les données nécessaires à une bonne analyse, ont fait de lui un atout majeur et incontournable pour toute entreprise soucieuse du suivi de ces performances.

Afin de mettre en place ce genre de système, il est nécessaire de choisir et d'adopter une démarche précise qui doit tenir compte des réalités de l'entreprise et des contraintes du projet. La modélisation de l'entrepôt se fait dans tous les cas grâce à la modélisation dimensionnelle. L'alimentation en données constitue l'étape à laquelle il faut accorder le plus d'attention et de temps. En effet, elle est le garant de contenance de l'entrepôt en données fiables et correctes. Une fois l'alimentation terminée, l'exploitation des données peut alors se faire par différentes méthodes.

Au cours de la seconde partie de cette étude, nous allons présenter les travaux ont été proposées purement destinées à la phase de conception ETL.

Chapitre II : Les travaux connexes

1- Introduction

Dans les scénarios de Data Warehouse (DW), les processus ETL (Extraction, Transformation, Loading) sont responsables de l'extraction de données à partir de sources de données opérationnelles hétérogènes, de leur transformation (conversion, nettoyage, normalisation, etc.) et de leur chargement dans le DW.

Durant ce travail de mémoire, nous avons eu l'occasion d'aborder plusieurs projets liés à notre travail de recherche. Nous présentons dans ce chapitre quelques-uns de ces travaux, nous faisons par la suite une comparaison entre eux selon un ensemble de critère afin d'adopter une approche simple et facile et qui repend à notre problématique

2- Travaux connexes

Il existe plusieurs travaux ont été proposées purement destinées à la phase de conception ETL, nous présentons les principaux travaux dédiés à la modélisation du processus ETL.

2-1- Ben Taher et al,2010 [1] :

Ils proposent une approche qui assure la génération semiautomatique des procédures ETL en se basant sur une correspondance structurelle et sémantique entre la source de données et les éléments multidimensionnels.

Ils ont proposé une approche de génération des opérateurs ETL pour alimenter un magasin de données à partir d'une source relationnelle. Pour ce faire, ils prennent en entrée le schéma conceptuel du magasin de données, mis en correspondance avec une source relationnelle. Ils définissent un ensemble de règles pour générer les opérateurs ETL pour un SGBD relationnel. Ces travaux offrent la majorité des règles nécessaires mais d'autres règles ont dû être ajoutées. L'objectif de ce travail est d'étudier les apports de la transformation de modèles pour le développement des entrepôts de données. En particulier, il vise à examiner l'applicabilité des techniques de transformation disponibles dans la phase de génération des procédures d'extraction-transformation-chargement des entrepôts de données.

2-2- D. Skoutas et al ,2006 [2]:

Ils présentent une approche basée sur les technologies du web sémantique pour faciliter le processus de sélection des informations pertinentes à partir des sources de

données pour transformer ces données et les charger dans un entrepôt de données. Donc, ces auteurs utilisent l'ontologie dans le but de spécifier la sémantique des schémas de la source de données ainsi que le schéma de l'entrepôt de données. Cette approche est adaptée pour capturer les informations nécessaires pour la conception des processus ETL. Elle est composée de quatre étapes qui sont : Définition du vocabulaire commun de l'application (ensembles de termes), Annotation des sources de données, Génération d'ontologie à partir du vocabulaire et de l'annotation des sources de données et Génération des opérateurs ETL. Par ailleurs, dans la génération des opérateurs ETL l'ontologie est réalisée manuellement.

2-3- Z. Zhang et al, 2008 [3]

Dans ce travail, ils proposent une approche de génération des opérateurs ETL en se basant sur l'utilisation d'une ontologie. Cette approche est composée de quatre principales phases (voir figure 2.1) dont la première détermine le processus d'extraction manuelle des données à partir des sources de données, ces données seront utilisées pour la construction de l'ontologie. A ce niveau, Il existe une ontologie globale et ontologie locale. La deuxième est la phase de mapping de l'ontologie dans laquelle définissent des relations sémantiques entre chaque ontologie locale avec celle globale. La troisième phase assure la dérivation de l'ontologie à partir des résultats de la phase précédente. La quatrième phase sert à définir manuellement les règles d'ETL afin d'opérer le processus ETL.

Bien que ces approches se basent sur une ontologie, elles ne s'intéressent pas à plusieurs types de relations sémantiques, comme la synonymie, hyperonyme, Matronyme, hyponyme et homonyme. De plus, nous constatons que ces approches raisonnent sur les ontologies pour dégager les opérateurs sans les générer. Par ailleurs, ces approches ne déterminent pas les enchainements à suivre pour aboutir au chargement de l'entrepôt.

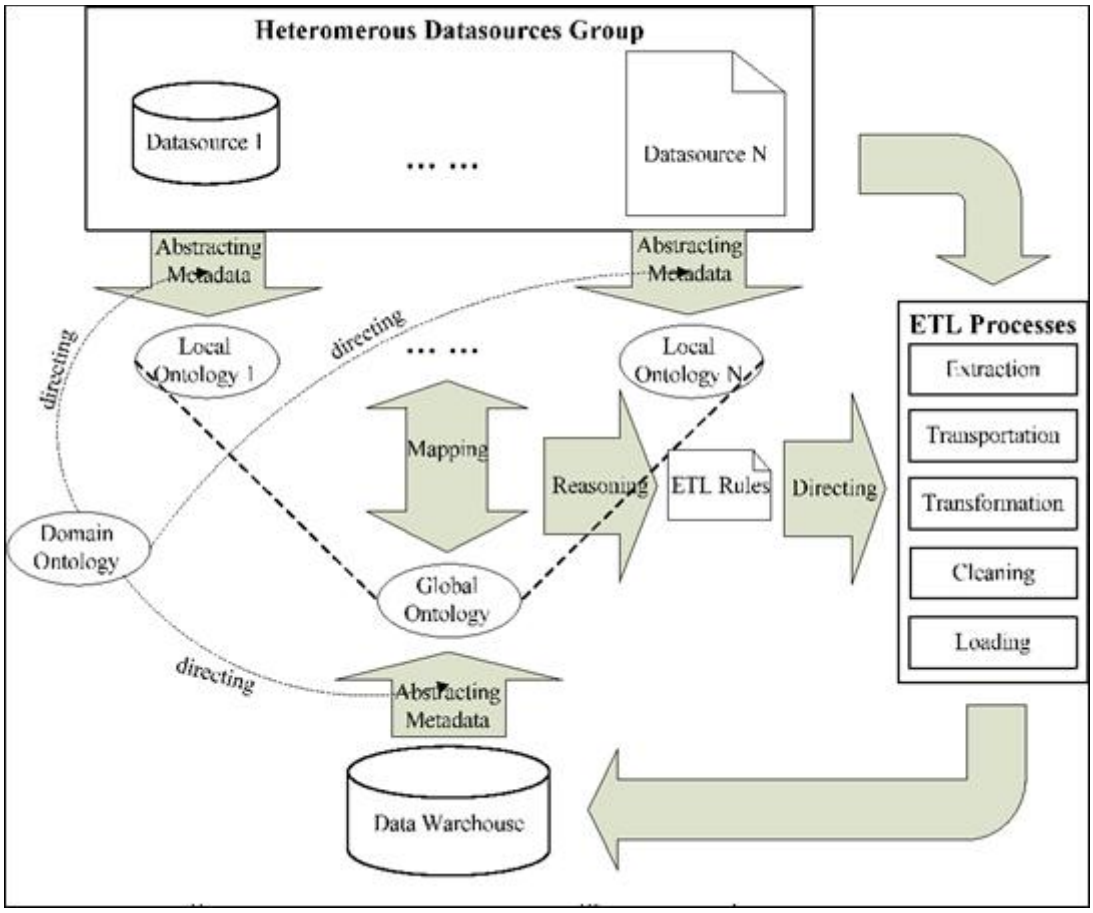


Figure 2.1 : A framework model for ontology-driven ETL processes

2-4- Castellanos et al, 2006 [4]

Ils proposent une deuxième approche de construction de l'ontologie, cette approche comprend trois phases principales, comme illustré à la Figure.

La première phase, appelée phase de pré-traitement, vise à l'extraction manuelle de la sémantique des schémas des magasins de données disponibles. Ce se déroule en deux étapes, à savoir extraction et sélection de termes.

Tout d'abord, pour chaque schéma de magasin de données, un ensemble de termes est extrait, qui transmet la sémantique des informations qu'il contient. Ensuite, ces ensembles de termes sont fusionnés, pour dériver la "terminologie" finale ou "Lexique" pour l'application en question. Ces termes choisis seront utilisés ensuite comme colonne vertébrale pour la création de l'ontologie pour l'ETL.

La deuxième phase, appelée phase de conception ETL, est la phase où l'ontologie est construite et la spécification conceptuelle du processus ETL est généré. En particulier, cela implique trois étapes principales : construction de l'ontologie, annotation du magasin de données, et génération ETL. La troisième phase, appelée phase de rapport, est responsable pour

générer des rapports concernant le processus ETL, ainsi que les magasins de données concernés, dans un format ressemblant au langage naturel.

❖

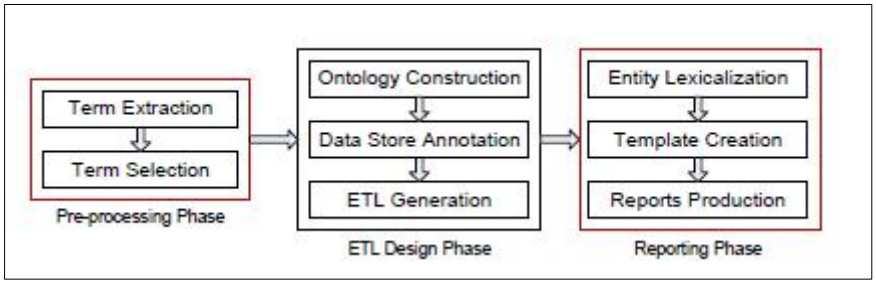


Figure 2.2 :Architecture du systéme

2-5- **Bala** et **Alimazighi ,2015 [5]**

Ils ont proposé une modélisation de l'ETL pour le Big Data selon le paradigme MapReduce enrichi par des notations graphiques pour modéliser les spécificités du modèle MR (voir figure 2.3).

MapReduce est un modèle de programmation ayant connu plusieurs implémentations sous forme de Frameworks destiné pour le traitement de données massives. Dans ce modèle, le programmeur spécifie le traitement en deux étapes en utilisant une fonction Map() et une fonction Reduce(). Le système MapReduce, fonctionnant sur une plateforme de type cluster, parallélise alors automatiquement le traitement en découpant le processus en sous processus où chacun sera confié à un nœud (fonction Map exécutée sur une machine du cluster) dont les résultats partiels seront soumis aux reducers (fonctions Reduce exécutées sur une machine du cluster) afin de restituer le résultat final.

Afin de préserver la trajectoire des données le long du processus, le mappage des données est représenté entre les données sources, la tâche de partitionnement des données et la phase Map, entre la phase Map et la phase Reduce et bien sûr entre la phase Reduce et le Datawarehouse. Le partitionnement est représenté afin de montrer comment les données sources sont partitionnées pour soumettre à chaque Mapper sa partition sur laquelle il opère les transformations nécessaires. Entre la phase Map et la phase reduce, le partitionnement montre comment les résultats partiels des mappers sont soumis aux reducers. La sémantique des Partitionneurs, des Mappers et des Reducers est explicitée par des Notes.

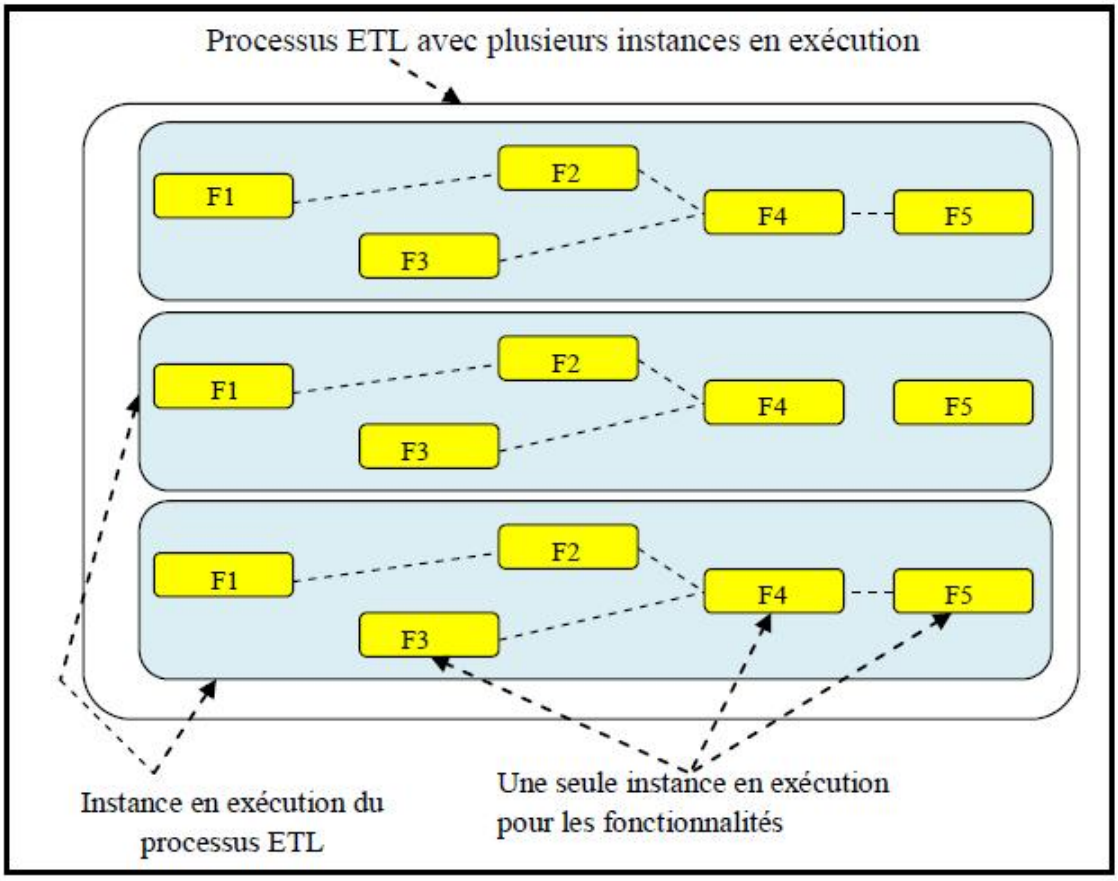


Figure 2.3 : Processus ***ETL*** basé sur le modèle ***MR***

2-6- Laifa et Tabiou,2016 [6]

Proposent une architecture qui modélise le processus ETL basé en utilisant Hive MapReduce dans les 3 phases du processus ETL (voir figure 2.4).

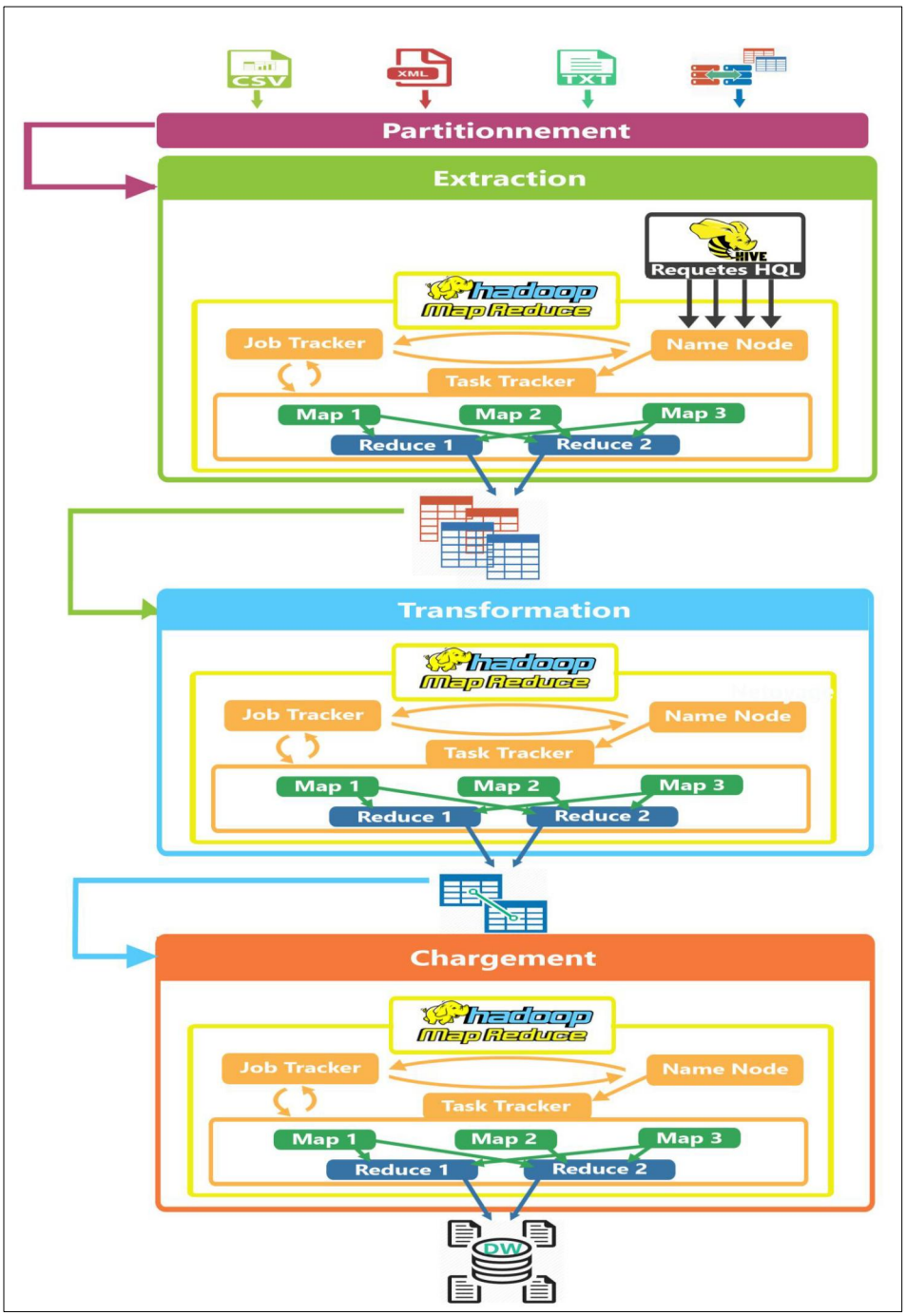


Figure 2.4 : architecture proposée modélise le processus ***ETL*** basé sur ***MAPREDUCE***

Après le partitionnement des données des sources avec la méthode round robine, pour minimiser le temps de recherche.

On commence par la première phase d'***ETL***

- **La phase d'extraction** : Connecter aux différentes sources hétérogène et distribué pour extraire des données destinées à l'exploitation pour analyser un sujet bien précis sera gardées, le but de cette phase est le nettoyage des donnes. Cette extraction se faire à travers des requêtes ***SQL***. ***Hive*** communique avec le ***Jobtracker*** pour lancer le travail

de ***MapReduce***. Le ***Jobtracker*** reçoit le job ***MapReduce*** pour partager le job et établit une liste des sous taches qui seront donnée au ***Tasktrackers*** Dans la figure on vue des pourparlers de ***Jobtracker*** au ***Name node*** pour :

- Déterminer l'emplacement des données .

- Copie avec une réplication haute les blocs des données dans un système de fichier du ***Jobtracker*** pour les ***Tasktrackers*** .

Le ***Tasktracker*** communique avec ***Jobtracker*** pour signifier leur disponibilité à exécuter les jobs, si le ***Jobtracker*** possède des jobs en file d'attente il affecté la tache au ***Tasktracker*** ce dernier copie le fichier depuis le système de fichier .Ensuite le ***Jobtracker*** ordonnance les différentes tâches des jobs soumis et assigne les tâches aux ***Tasktrackers***.

- **La phase de transformation**

Apres l'exécution du Reduce de la phase d'extraction les résultats obtenu seront transformer et filtrées et agrégés avec des fonctions d'agrégation (SUM, CAUNT, etc…) . C'est une suite d'opérations qui a pour but de rendre les données cibles homogènes et puissent être traitées de façon cohérente.

L'ensemble des données sources, après nettoyage ou transformation d'après des règles précises ou par application de programmes (pour un contrôle de vraisemblance par des méthodes statistiques), seront restructurées et converties dans un format cible.

On Applique cette phase de transformation avec le même mécanisme appliqué dans l'extraction ***MR*** , Pour obtenir des données prêts à être chargé dans ***l'entrepôt de données*** .

- **La phase du chargement**

C'est l'opération qui consiste à charger les données nettoyées et préparées dans le ***DW*** avec le même mécanisme ***MR*** appliqué dans la phase d'extraction et de transformation.

2-7- W.Bakari et al [7]

Pour la génération des opérateurs, les auteurs de [8] proposent une approche qui se base sur la transformation de modèles. Donc, la génération des opérateurs ETL se base essentiellement sur la correspondance entre la source et la cible de données. Dans ce cas, ils parlent de la correspondance entre les éléments de l'entrepôt et ceux de la source de données. Cette correspondance est apparue dans le fichier de départ qui est le Mapping. Leur approche détermine des scénarii de chargement des magasins de données. Pour assurer la génération des opérateurs ETL, en tenant compte des tâches suivantes : La génération des opérateurs ETL, les

règles de transformation, le processus de transformation de modèles et de chargement du modèle cible.

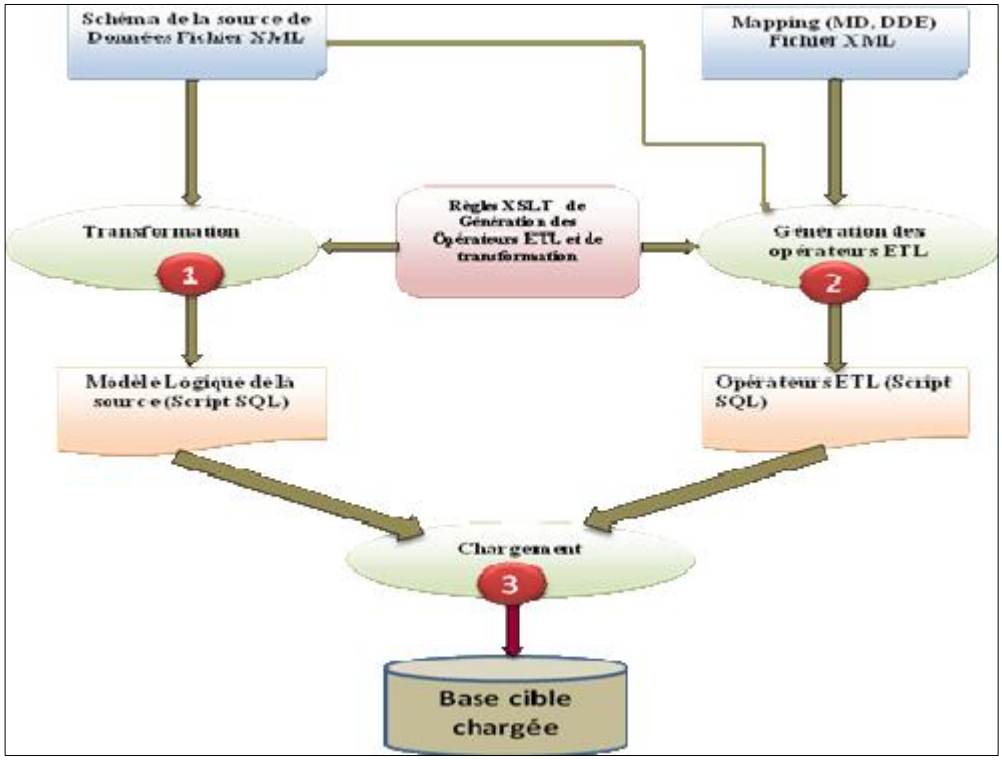


Figure 2.5: Démarche de génération des opérateurs ETL

Comme illustré dans la Figure ci-dessus, leur démarche de génération des opérateurs ETL prend en entrée deux fichiers XML : le premier c'est le Mapping [1] qui décrit pour chaque élément du magasin de données ses correspondances avec les éléments de la source. Le deuxième fichier d'entrée contient le schéma de la source de données. Ceci décrit des informations concernant les tables, leurs attributs, les contraintes de clé primaire et étrangère, etc. Ce schéma est utilisé pour définir le modèle logique de la source de données. Ce dernier est utilisé pour la génération des opérateurs ETL.

2-8- C.Gueydan ,2010 [8]

Présente un outil de type ETL qui permet de concevoir et d'exécuter des processus ETL afin d'alimenter une base de données cible avec des données issues de différentes bases de données sources (voir figure 2.6). Cette base de données cible peut être une base de données standard ou prendre la forme d'un entrepôt de données. XeuTL utilise des fichiers XML comme support d'échange.

XeuTL se compose d'un noyau chargé de réaliser l'intégration de données et de différents modules qui permettent au système d'interagir avec l'utilisateur, les sources de données et les fichiers XML utilisés par l'application.

Les trois modules implémentés autour du noyau sont :

- le gestionnaire de communication : il est chargé de dialoguer avec les différentes bases de données concernées. Il effectue les connexions aux différentes bases de données, exécute les requêtes SQL fournit les résultats au noyau ;

- le gestionnaire de documents XML : il permet de générer et d'analyser les documents

XML utilisés par l'application. Il crée les fichiers XML lorsque cela est nécessaire, lit le contenu des fichiers XML et y écrit des données ;

- l'outil interactif de manipulation : il permet à l'utilisateur d'interagir avec le système.

Deux modes de manipulation sont possibles : le premier via une API qui fournit l'ensemble des méthodes nécessaires à l'utilisateur pour manipuler XeuTL, le second via une interface graphique. Celle-ci étant en cours de développement, elle n'est pas présentée dans ce mémoire.

Le noyau de l'application est composé de trois modules :

- le générateur de requêtes : il est chargé de créer les ordres SQL qui permettent d'extraire, de transformer ou d'importer des données depuis ou vers une base de données ;

- le gestionnaire de représentations des sources de données : il permet de représenter en mémoire le schéma physique des bases de données à traiter ;

- le gestionnaire de transformations : il travaille en étroite collaboration avec le générateur de requêtes et le gestionnaire de représentations, afin de traiter les demandes de transformations paramétrées par l'utilisateur sous forme de correspondances.

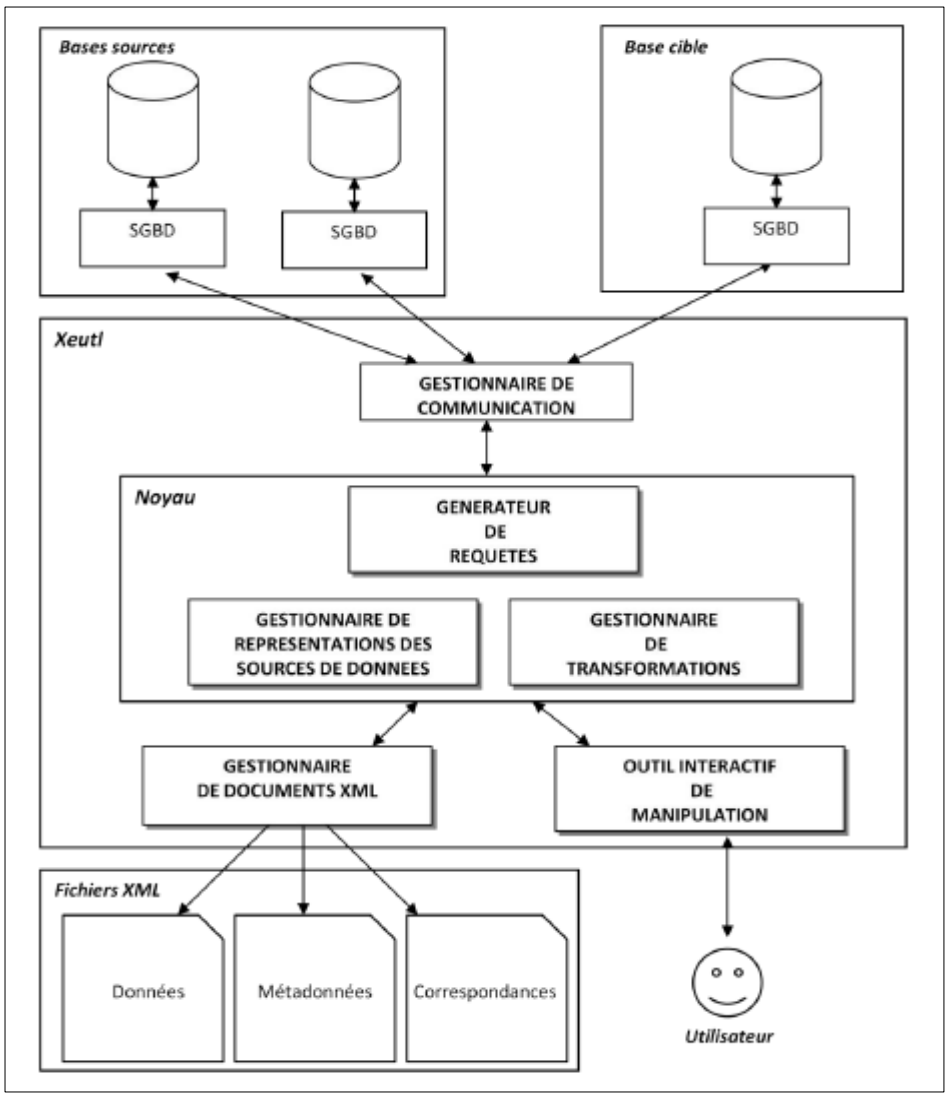


Figure 2.6: architecture de XeuTL

3- Analyse et synthèse des travaux

Dans cette partie on va faire une étude comparative des travaux antérieurs en se basant sur plusieurs critères de comparaison afin d'adopter une approche opportune.

3-1- Critère de comparaison

- ✓ **Automaticité :** Le premier critère déterminer si la phase ETL suivre une approche manuelle, semi-automatique ou automatique on a choisis ce critère de comparaison afin d'éviter les approches manuelles.

MAN : manuelle

SAU : semiautomatique

AUT : automatique

- ✓ **Phase modélisée :** Ce critère indique quelles sont les phases ***ETL*** traités : extraction, transformation ou chargement et puisque notre système est un système de collecte de données donc on va se baser sur la phase d'extraction.

E : extraction
T : transformation
L : chargement

Type de source : Ce critère indique quelles sont les sources de données traitées : base de données, fichiers structurels… on a choisis ce critère afin de baser sur les travaux qui traitent des sources hétérogènes.

BDD : base de données.
FC : Fichiers structurels.

- ✓ **Approche adoptée :** Ce critère détermine la broche adoptée dans le travail basé sur : les ontologies, web sémantique, correspondance…

ONT : ontologies
MR : MapReduce.
COR : Correspondance.

- ✓ **Simplicité :** Ce dernier critère détermine si l'approche est une approche simple ou complexe à réaliser. Ce critère permet de nos orienter vers des travaux simple et facile à adopter vus le temps limité de réalisation.

SIM : simple.
COM : complexe.

3-2- Synthèse des travaux

Travail	Automaticité			Phase modélisé			Type de source		Approche adoptée			Simplicité	
	MAN	SAU	AUT	E	T	L	BDD	FC	ONT	MR	COR	SIM	COM
1		x		x	x	x	x				x		x
2		x		x	x	x	x		x				x
3		x		x	x	x	x		x				x
4		x		x	x	x	x		x				x
5		x		x	x	x	x	x		x			x
6		x		x	x	x	x	x		x			x
7			x	x			x				x	x	
8		x		x			x				x	x	

Tableau 2.1 : Tableau comparatif

Nous avons résumé dans ce tableau, les différents travaux dédiés à la conception du processus ETL, on remarque que la plupart des approches sont semi-automatiques ou le concepteur fait partie du processus ; Quelques travaux comme [7] et[8] étudient une partie du processus ***ETL*** (la partie extraction), mais la plupart traitent les trois activités du processus ETL. (Extraction, transformation, chargement).

Plusieurs approches sont basées sur une ontologie dont l'utilisation ou la construction comme [2], [3] et [4]. Par contre, il y a d'autres approches qui évitent la construction d'une telle ontologie, par exemple, l'approche proposée par les auteurs dans [1], pour la génération semiautomatique des procédures ETL en se basant sur une correspondance structurelle et sémantique entre la source de données et les éléments multidimensionnels, les auteurs de [7] proposent une approche qui se base sur la transformation de modèles par contre les auteurs de [8] proposent une approches qui consiste à la génération des opérateurs ETL qui se base essentiellement sur la correspondance entre la source et la cible de données.
Les travaux de recherche comme [5]et [6] proposant des contributions voir des améliorations et adaptations du processus ***ETL*** basé sur ***MapReduce*** pour le traitement de données massives. La plupart des travaux sont un peu compliqué et nécessitent beaucoup de temps pour les réaliser.

4- Conclusion

Dans ce chapitre, nous avons étudié quelques travaux connexes ayant abordé la conception du processus ETL. Ensuite nous avons tenté une étude comparative des travaux connexes à notre problématique en utilisant plusieurs critères de comparaison tels que : l'automaticité, Phase modélisé, Type de source, Approche adoptée…

En se basant à la contribution des auteurs de [8] qui consiste à la génération des opérateurs ETL qui se base essentiellement sur la correspondance entre la source et la cible de données où l'utilisateur définit des correspondances entre les schémas physiques des bases sources et celui de la base cible, résolvant ainsi l'hétérogénéité sémantique des données et en utilisant des fichier XML comme support d'échange. Notre contribution consiste à généraliser cette contribution et l'appliquer sur des fichiers Excel et des formulaires de saisie et non seulement sur les bases de données relationnels.

Partie II :

Contribution et réalisation

Chapitre III : Proposition de solution et conception du système

1- Introduction

Le système de collecte de données un système d'extraction de données de type ETL qui permet de concevoir et d'exécuter des processus ETL afin d'alimenter une base de données cible avec des données issues de différentes sources de données.

Cette base de données cible peut être une base de données standard ou prendre la forme d'un entrepôt de données. Notre système de collecte de données utilise des fichiers XML comme support d'échange.

Ce chapitre présente le fonctionnement général du système de collecte de données ainsi qu'une étude conceptuelle en utilisant le langage de modélisation, UML (diagramme des cas d'utilisation, diagramme de séquence).

2- Objectifs du système de collecte de données

Un processus ETL est une suite d'opérations nécessaires à l'alimentation d'une base cible avec des données réparties dans différentes bases sources hétérogènes : Extraction,

Transformation, Chargement. La mise en place de processus ETL dans l'entreprise est une tâche lourde et complexe. Notre système de collecte de données a pour objectif de simplifier cette tâche.

Notre système ne se veut pas un concurrent des solutions lourdes du marché mais se présente comme une alternative **légère** et plus **accessible** en termes d'apprentissage, de mise en place et d'utilisation, il permet de réaliser des migrations de données et des alimentations ou des réalimentations et il permet de « combiner » les données de différentes sources hétérogènes vers une base cible.

Notre système de collecte de données est centré sur les processus ETL et ne fournit pas de composants décisionnels.

3- Fonctionnement général

Notre système de collecte de données permet aux utilisateurs d'extraire les données de différentes sources hétérogènes (bases de données relationnelles, fichiers Excel, de formulaires de saisies manuelles) de manière simple.

La figure 1 présente les différentes étapes de notre système de collecte de données.

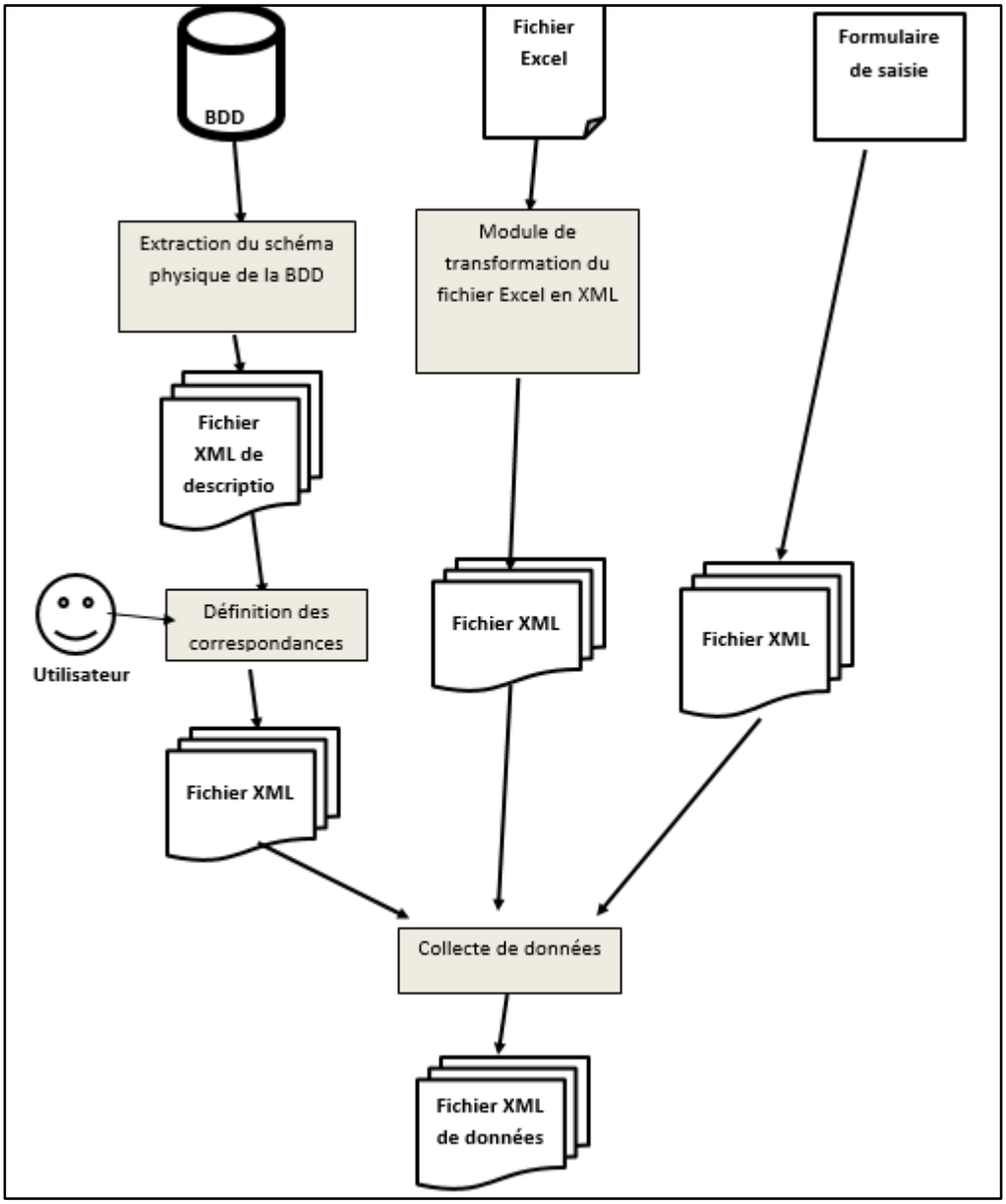


Figure 3.1 : Etape de système de collecte de données

Dans une première étape, Notre système de collecte de données il convertit les fichiers Excel en XML ainsi il enregistre les données des formulaires sous la forme des fichiers XML et extrait le schéma physique et les données des bases sources. L'utilisateur définit ensuite les correspondances entre les bases sources et cible et le système fait la collecte des données sous la forme d'un fichier XML.

4- **Principales fonctionnalités**

Cette section présente les principales fonctionnalités du système de collecte de données.

Connexion au compte utilisateur

Chaque utilisateur du système possède un compte, où il doit se connecter pour avoir un accès à ses différentes sources de données.

Gérer les formulaires de saisie manuelle

Le système permet à l'utilisateur de créer un nouveau formulaire ou de choisir un formulaire existant afin de saisir les données. Le système enregistre la structure du formulaire ainsi que les données saisies dans un fichier XML.

Transformer un fichier Excel

Le système permet de convertir un fichier Excel à un fichier XML contenant la structure et les données du fichier Excel.

Extraction du schéma physique des bases de données

Cette fonctionnalité permet de se connecter à une base et d'extraire ses métadonnées (tables du schéma physique, clés, contraintes, …) les stockées dans un fichier XML (un fichier par base). Notre système de collecte de données est compatible avec différents systèmes de gestion de bases de données.

Etablissement des correspondances entre les schémas sources et cible

Notre système de collecte de données fournit des outils permettant de mettre en place des correspondances variées entre les schémas sources et cible d'un processus ETL. Ces correspondances sont utilisées par Notre système de collecte de données pour effectuer les transformations qui sont au coeur du processus ETL.

Extraction des données des bases sources au format XML

Le système de collecte de données possède un mécanisme d'extraction de données adapté à différents SGBD. Lors de l'extraction, les données sont transformées en utilisant les correspondances précédemment définies. Le résultat de l'extraction de données des bases sources est un fichier XML contenant les données à transférer vers la base cible.

Beaucoup de SGBD (notamment tous les SGBD relationnels) utilisent le langage SQL comme langage de manipulation de données. Cependant, les implémentations du langage SQL par différents constructeurs ne sont pas toutes identiques. Les différences sont encore plus marquées lorsque les gestionnaires des sources de données sont de différents types (fichiers,

SGBD réseaux, hiérarchiques, relationnels, …).

Dans Notre système de collecte de données, le langage XML est utilisé comme format pivot. Ceci permet de transférer des données entre des SGBD hétérogènes.

5- Les correspondances

Les correspondances définies par l'utilisateur indiquent au système la provenance des données qui vont alimenter la base cible, ainsi que le format qu'elles doivent respecter.

Chaque correspondance peut être vue comme un lien entre une ou plusieurs colonnes des bases sources et une colonne de la base cible.

Nous avons distingué deux types de correspondances :

- les correspondances simples, définies entre une unique colonne d'une base source et une unique colonne de la base cible,

- les correspondances complexes, qui peuvent être définies entre plusieurs colonnes des bases sources et une unique colonne de la base cible.

5-1- Les correspondances simples

- **Correspondances atomiques**

Une correspondance atomique est la correspondance la plus simple qui puisse être définie dans le système. Elle associe une colonne d'une table de la base source à une colonne d'une table de la base cible. Les données de la colonne source seront copiées dans la colonne cible.

Exemple :

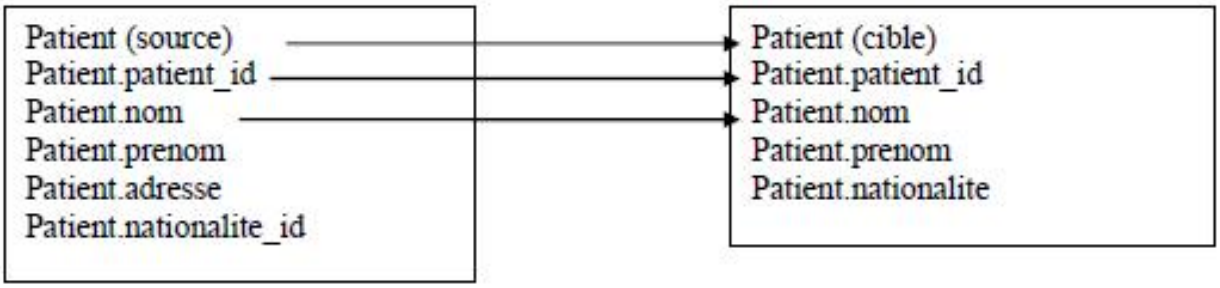


Figure 3.2 : exemples de correspondances atomiques

Patient				
patient_id	nom	prenom	adresse	nationalite_id
1	Dupond	Alexandre	Rue Montorge	1
2	Durand	Martine	7 Avenue du bois, 39098	2
3	Martin	Celia	Lyon	2

Tableau 3.1 : table *Patient* de la base source

Patient			
patient_id	nom	prenom	nationalite_id
1	Dupond	Alexandre	
2	Durand	Martine	
3	Martin	Celia	

Tableau 3.2 : table *Patient* de la base cible une fois la correspondance est effectué

- **Correspondances de type calcul**

Une correspondance de type calcul est une correspondance qui va appliquer un calcul sur une valeur d'une colonne source avant de copier le résultat obtenu dans la colonne.

Les calculs disponibles sont des calculs simples relativement standard (addition, soustraction, multiplication, division, …).

Exemple :

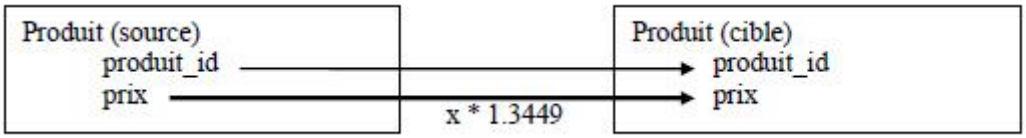


Figure 3.3 : exemple de correspondance de type calcul

Produit	
produit_id	prix
1	10
2	12.5
3	2.30

Tableau 3.3 : table *Produit* de la base source

Produit	
produit _id	prix
1	13.449
2	16.81125
3	28.75

Tableau 3.4 : table *Produit* de la base cible après exécution

- **Correspondances de type valeur fixe**

Une correspondance de type « valeur fixe » est une correspondance qui va affecter pour chaque enregistrement une même valeur fixe à la colonne cible.

Exemple :

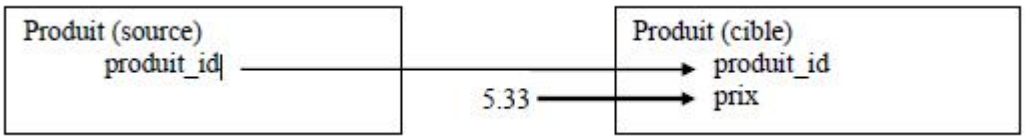


Figure 3.4: exemple de correspondance de type valeur fixe

Produit produit_id
1
2
3

Tableau 3.5 : table *Produit* de la base source

Produit	
produit_id	prix
1	5.33
2	5.33
3	5.33

Tableau 3.6 : table *Produit* de la base cible après exécution

- **Correspondances de type transtypage**

Une correspondance de type transtypage est une correspondance qui va effectuer une modification du type des valeurs copiées de la colonne source vers la colonne cible lors de l'exécution du processus ETL. Ce type de correspondance est particulièrement utile lorsque le système de gestion de base de données de la base source est différent de celui de la base cible.

Par exemple, le type *VARCHAR* en Mysql a une capacité de 255 caractères alors qu'en SQL

Server, le type *VARCHAR* a une capacité de 4000 caractères. Le type de champ Mysql correspondant à un *VARCHAR* SQL Server serait plutôt un champ de type *TEXT*.

- **Correspondances de type clé de substitution**

Une correspondance de type clé de substitution est une correspondance qui permet de générer une clé automatique pour une colonne de la base cible.

Exemple :

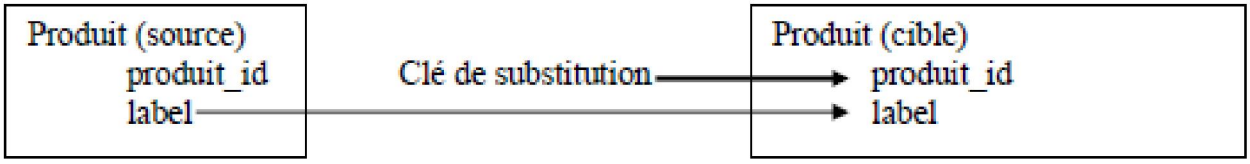


Figure 3.5 : exemple de correspondance de type clé de substitution

Produit	
produit_id	label
1	Stylo
2	Papier
3	Pile

Tableau 3.7 : Table *Produit* de la base source

Produit	
produit_id	label
20	Livre
21	Chargeur
22	DVD
23	CD
24	Prise

Tableau 3.8 : table *Produit* de la base cible avant exécution

Produit	
produit_id	label
20	Livre
21	Chargeur
22	DVD
23	CD
24	Prise
25	Stylo
26	Papier
27	Pile

Tableau 3.9 : table *Produit* de la base cible après exécution

Le résultat de l'application d'une transformation de type clé de substitution est identique à celui d'une requête d'insertion effectuée sur une table dont la clé primaire est auto-incrémentée.

Dans notre exemple, la correspondance définie aurait le même résultat que l'exécution des requêtes SQL suivantes si la colonne *produit_id* de la table *produit* de la cible était auto-incrémentée

:

```
Insert into Produit(Label) values ('Stylo');
Insert into Produit(Label) values ('Papier');
Insert into Produit(Label) values ('Pile');
```

- **Correspondances de type troncature**

Une correspondance de type troncature est une correspondance qui va tronquer la valeur d'une colonne source avant de copier le résultat obtenu dans la colonne cible.

Exemple :

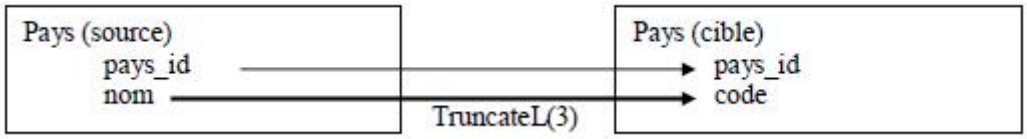


Figure 3.6 : exemple de correspondance de type troncature

Pays	
pays_id	nom
1	FRANCE
2	ANGLETERRE
3	IRLANDE

Tableau 3.10 : table *Pays* de la base source

Pays	
pays_id	code
1	FRA
2	ANG
3	IRL

Tableau 3.11 : table *Pays* de la base cible après exécution

5-2- Les correspondances complexes

- **Correspondances de type référence**

1- Depuis la base source

Soient deux tables sources T1 et T2 et C1 une des colonnes de T1 définie comme clé étrangère référençant C2, une des colonnes de T2. Une correspondance de type référence associe une colonne de T2 (autre que la colonne référencée par la clé étrangère de T1) à une colonne de la table cible. La valeur copiée dans la base cible sera celle de l'enregistrement référencé par la clé étrangère. En d'autres termes, une table source fait référence à une autre table source. La valeur récupérée pour alimenter la cible sera alors la valeur de la colonne de la seconde table de l'enregistrement référencé par la clé étrangère de la première table.

Exemple :

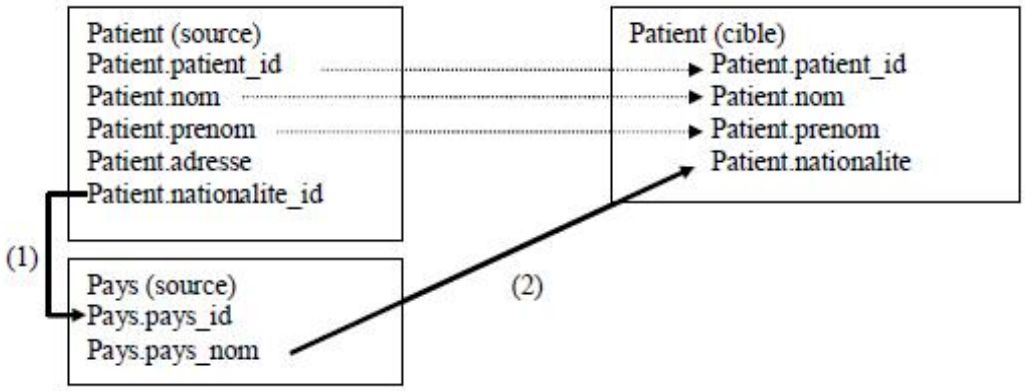


Figure 3.7 : correspondance de type référence (en gras)

La flèche (1) indique la clé étrangère définie entre ces deux tables. La flèches (2) indique la correspondance de type référence ; les autres flèches indiquent des correspondances de type atomiques.

Patient				
patient_id	nom	prenom	adresse	nationalite_id
1	Dupond	Alexandre	Rue Montorge	1
2	Durand	Martine	7 Avenue du bois, 39098	2
3	Martin	Celia	Lyon	2

Tableau 3.12 : table *Patient* de la base source

Pays	
pays_id	pays_nom
1	France
2	Suisse
3	Royaume-Uni

Tableau 3.13 : table *Pays* de la base source

Patient			
patient_id	nom	prenom	nationalite
1	Dupond	Alexandre	France
2	Durand	Martine	Suisse
3	Martin	Celia	Suisse

Tableau 3.14 : table *Patient* de la base cible après exécution

2- Depuis la base cible

Une correspondance de type référence depuis la cible est une correspondance définie entre une colonne d'une table source et une colonne d'une table cible, cette dernière possédant une clé étrangère référençant une autre table cible. La colonne cible de la correspondance

prendra comme valeur celle de la colonne source. Le système génèrera de manière automatique une clé qui alimentera les colonnes référencées et référençant la clé étrangère définie entre les deux tables cibles.

Exemple :

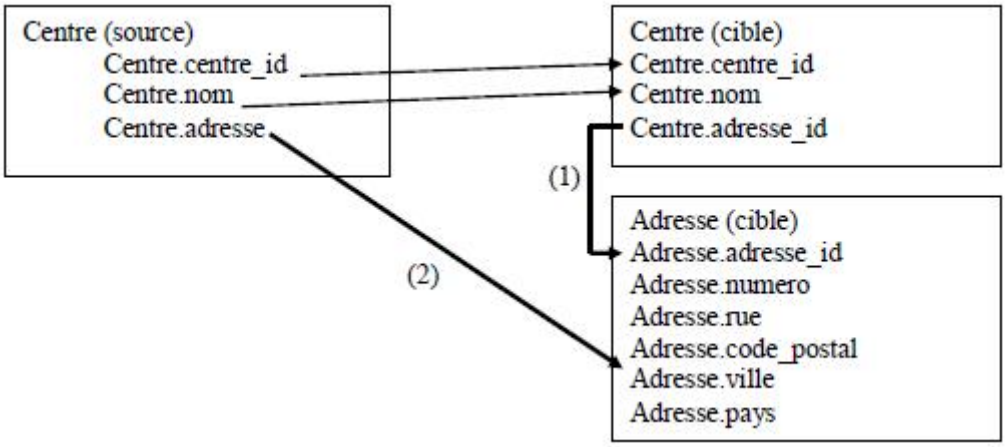


Figure 3.8 : correspondance de type référence (en gras)

La flèche (1) indique une clé étrangère. La flèche (2) indique la correspondance de type référence, les autres indiquent les correspondances atomiques.

Centre		
centre_id	nom	adresse
1	CHU Grenoble	Grenoble
2	HUG Genève	Genève
3	Hôpital sud	Echirolles

Tableau 3.15 : table *Centre* de la base source

Centre		
centre_id	nom	adresse_id
1	CHU Grenoble	1
2	HUG Genève	3
3	Hôpital sud	2

Tableau 3.16 : table *Centre* de la base cible après l'opération de migration

Adresse					
adresse_id	numero	rue	code_postal	ville	pays
1				Grenoble	
2				Echirolles	
3				Genève	

Tableau 3.17 : table *Adresse* de la base cible après l'opération de migration

- **Correspondances de type concaténation**

Une correspondance de type concaténation permet de concaténer les valeurs de plusieurs champs d'un enregistrement de la base source et de copier la chaîne ainsi obtenue dans un champ de la base cible.

Exemple :

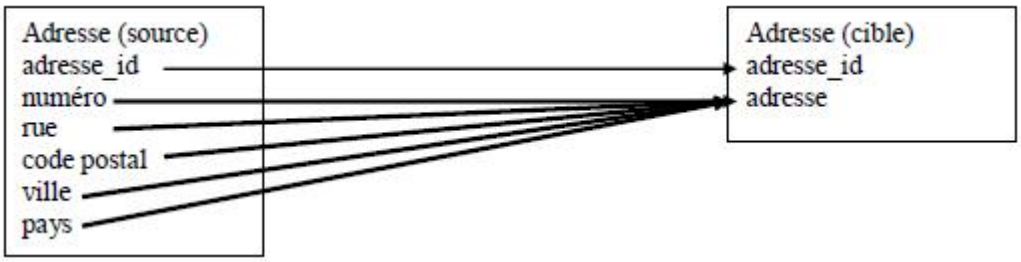


Figure 3.9 : correspondance de type concaténation

Les flèches en gras indiquent les correspondances de type concaténation.

Adresse				
adresse_id	numéro	rue	code postal	ville
1	12	Turenne	38000	Grenoble
2	102	Alembert	38000	Grenoble
3	7	République	69000	Lyon

Tableau 3.18 : table *Adresse* de la base source

Adresse	
adresse_id	adresse
1	12 Turenne 38000 Grenoble
2	102 Alembert 38000 Grenoble
3	7 République 69000 Lyon

Tableau 3.19 : table *Adresse* de la base cible

- **Correspondances de type requête imbriquée**

Les correspondances de type requête imbriquée permettent de déduire la valeur d'un champ cible d'une requête effectuée sur la base source.

Exemple :

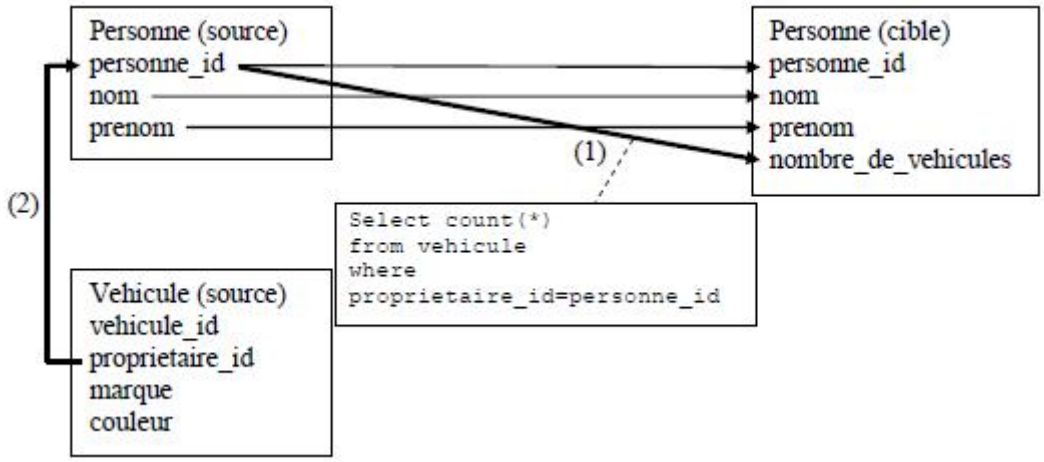


Figure 3.10: correspondance de type requête imbriquée

La flèche en gras entre la base source et la base cible indique une correspondance de type requête imbriquée ; celle entre les deux bases sources indique une clé étrangère, et les autres indiquent les correspondances atomiques. `proprietaire_id` est une clé étrangère référençant `personne_id` et implémentant la relation *propriétaire*.

Dans ce contexte, la requête d'agrégation

```
(Select count(*) from vehicule where proprietaire_id = personne_id)
```

permet de compter le nombre de véhicules dont une personne est propriétaire et de stocker cette information dans la base cible.

Personne		
personne_id	nom	prenom
1	Durand	Pierre
2	Dupond	Alexandre
3	Rougemont	Yves
4	Martin	Simon

Tableau 3.20 : table *Personne* de la base source

Vehicule			
vehicule_id	proprietaire_id	marque	couleur
1	2	Peugeot	rouge
2	2	Citroën	grise
3	2	Audi	noire
4	4	Chrysler	blanche

Tableau 3.21 : table *Vehicule* de la base source

Personne			
personne_id	nom	prenom	nombre_de_vehicule
1	Durand	Pierre	0
2	Dupond	Alexandre	3
3	Rougemont	Yves	0
4	Martin	Simon	1

Tableau 3.22 : table *Personne* de la base cible après exécution

6- Conception du système

Dans cette partie on va exposer la conception du système de collecte automatique de données.

6-1- Les cas d'utilisation

Les cas d'utilisation sont définis par une description textuelle, décrivant les objectifs et interactions entre le système et ses acteurs. Le format de présentation textuelle des cas d'utilisation est libre, mais il existe quelques propositions reconnues dans le domaine [16].

Un cas d'utilisation représente un ensemble de séquences d'actions réalisées par le système et produisant un résultat observable intéressant pour un acteur particulier [17].

- **La liste des cas d'utilisation**

Numéro du cas d'utilisation	**Nom du cas dutilsation**
Cas d'utilisation 1	Création compte utilisateur
Cas d'utilisation 2	Connexion a un compte
Cas d'utilisation 3	Transformation du fichier Excel en XML
Cas d'utilisation 4	Gestion du formulaire de saisie
Cas d'utilisation 5	Extraction de la structure d'une base de données.
Cas d'utilisation 6	Etablissement des correspondances.

Cas d'utilisation 7	Collecte des données.

Tableau 3.23 : la liste des cas d'utilisation

- **Diagramme de cas d'utilisation**

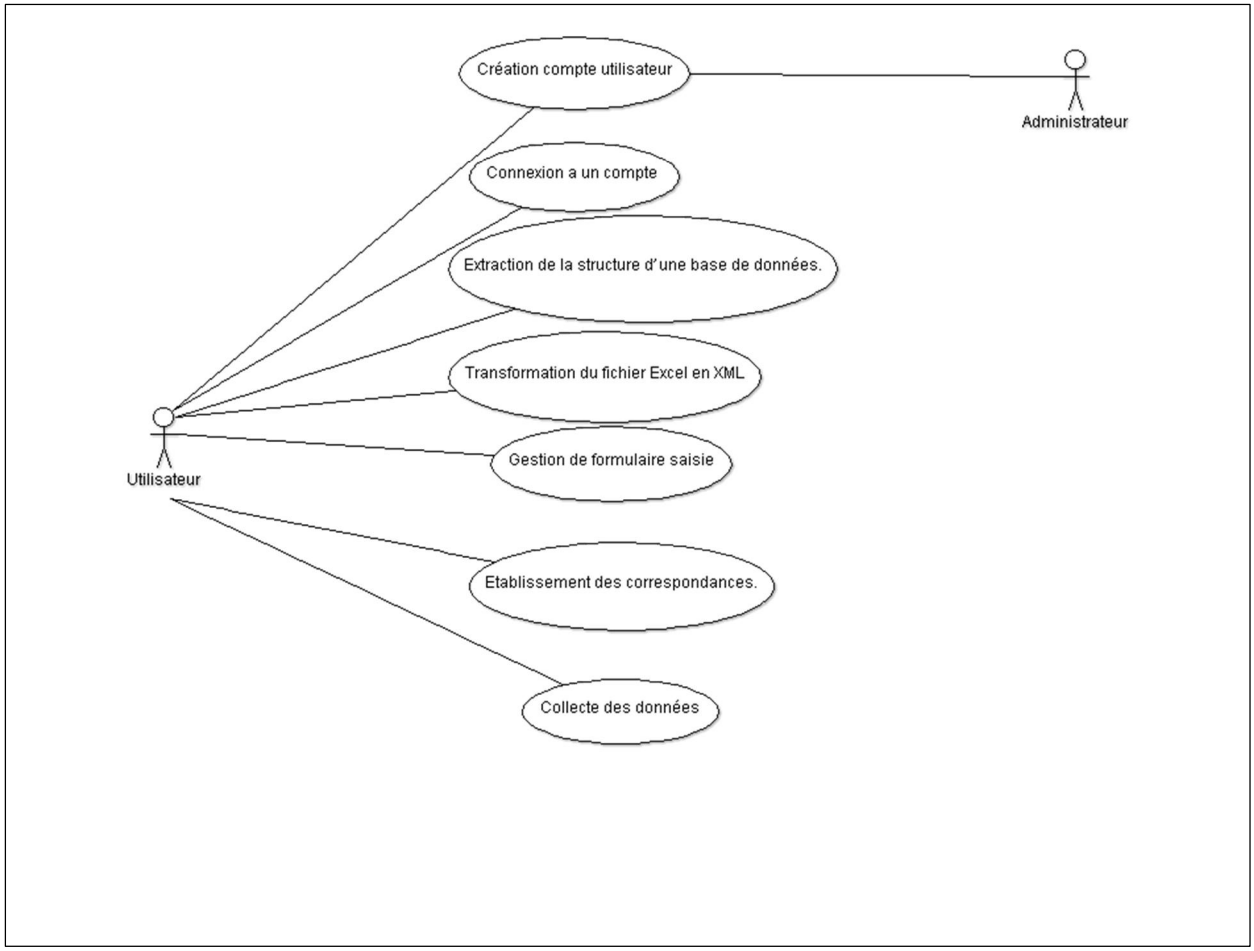


Figure 3.11 : Diagramme de cas d'utilisation

- **Les cas d'utilisation en détail**

Cas d'utilisation 1
Titre : créer un compte **Résumé** : le but de ce cas d'utilisation est de permettre à l'administrateur de créer un compte utilisateur sur le système. **Acteur** : l'utilisateur, administrateur.

Pré condition : l'utilisateur demande la création d'un compte.

Enchaînements nominaux :

1- Le système affiche le formulaire de création
2- L'utilisateur saisie ses informations.
3- L'administrateur vérifie les informations saisies et valide
4- Le système envoie le message de succès

Exceptions

1. l'administrateur vérifie et ne valide pas le formulaire

2. Retour à l'étape numéro 1 du scénario principal

Cas d'utilisation 2

Titre : connexion à un compte (authentification)

Résumé : le but de ce cas d'utilisation est de permettre à l'utilisateur de se connecter à son compte.

Acteur : l'utilisateur.

Pré condition : l'utilisateur demande de se connecter au s'système.

Enchaînements nominaux :

1- L'utilisateur saisit son login et mot de passe
2- L'utilisateur valide
3- Le système vérifie les données saisies

Exceptions : Un message d'erreur est affiché

Cas d'utilisation 3
Titre : ***Transformation du fichier Excel en XML***   **Résumé** : convertir un fichier Excel sous un format XML   **Acteurs :** utilisateur.   **Pré conditions** : le fichier Excel est accessible.
Enchaînements nominaux :   - l'utilisateur indique le fichier Excel à convertir,   - le système lit le fichier Excel,   - le système créer un fichier XML à partir de fichier Excel.

Cas d'utilisation 4
Titre : ***Gestion de formulaire de saisie***   **Résumé** : le choix d'un formulaire de saisie existant sinon la création d'un nouveau formulaire.   **Acteurs :** utilisateur.   **Pré conditions** : l'utilisateur s'est connecté à son compte.
Enchaînements nominaux :   - l'utilisateur demande de consulter les formulaires existant sur le système,   - le système affiche tous les formulaires déjà crée par l'utilisateur.   - l'utilisateur choisis un formulaire de saisie,   -le système lui affiche le formulaire vierge,   -l'utilisateur remplis le formulaire,   -le système crée un fichier XML contenant les données saisies par l'utilisateur.   **Enchaînements alternatifs :**   -le formulaire n'existe pas sur le système,   - l'utilisateur crée un nouveau formulaire de saisie,   - le système enregistre le nouveau formulaire de saisie.

Cas d'utilisation 5
Titre : ***Extraction de la structure d'une base de données***

Résumé : L'extraction de la structure des sources de données va permettre de représenter les schémas des bases sources et cibles à l'utilisateur afin que celui-ci puisse définir les correspondances qu'il souhaite mettre en place entre les bases sources et la base cible. **Acteurs :** utilisateur. **Pré conditions** : les bases sont accessibles
Enchaînements nominaux : - l'utilisateur indique la base de données à laquelle il souhaite se connecter, - le système créé un fichier XML contenant les informations de connexion à la base, - l'utilisateur indique qu'il souhaite extraire le schéma physique de la base, - le système établit la connexion à la base en utilisant les informations du fichier XML de connexion, - le système extrait le schéma physique de la base et le stocke dans un fichier XML.

Cas d'utilisation 6
Titre : ***Etablissement des correspondances*** **Résumé :** En se basant sur les schémas générés précédemment, l'utilisateur définit des correspondances entre les bases sources et la base cible. La définition de ces correspondances est stockée dans un fichier XML. **Acteur :** utilisateur. **Préconditions :** tous les fichiers XML de définition de schémas physiques sont disponibles.
Enchaînements nominaux : - l'utilisateur demande de créer une correspondance entre les bases sources et la base cible. - le système demande à l'utilisateur de choisir les fichiers XML source et cible. - l'utilisateur faire les correspondances entre les éléments source et cible, -le système crée un fichier XML de correspondance et il l'enregistre.

Cas d'utilisation 7

Titre : ***Collecte des données.***   **Résumé :** l'utilisateur souhaite extraire les données sources en y appliquant les transformations déduites des correspondances qu'il a définies.   **Acteur** : utilisateur.   **Préconditions** : les sources sont accessibles et les fichiers XML de définition de correspondances existent.
Enchaînements nominaux :   - l'utilisateur demande à procéder à l'extraction et à la transformation des données,   - le système demande à l'utilisateur de choisir les fichiers sources et les fichiers XML de définition de correspondances,   - le système génère les requêtes SQL nécessaire,   - le système exécute les requêtes SQL généré,   - le système stocke le résultat dans le fichier XML de données extraites.

6-2- Les diagrammes de séquence :

Avec les diagrammes de séquence, l'UML fournit un moyen graphique pour représenter les interactions entre objets à travers le temps, ces diagrammes vont montrer typiquement un utilisateur et les objets et composants avec lesquels il interagit au cours de l'exécution des cas d'utilisations, un diagramme de séquence représente généralement un scénario. **[18]**

Les diagrammes de séquence servent à décrire les scénarios nominaux des cas d'utilisation d'un système. Ils fournissent des détails que les diagrammes de cas d'utilisation ne puissent pas les fournir. Ils montrent les interactions entre le système et ses acteurs pour chaque cas d'utilisation **[18]**

Dans cette section, nous exposons des diagrammes de séquence pour chacun des cas d'utilisation mentionnés dans la section précédente.

- **Diagramme de séquence 1 « Création compte utilisateur »**

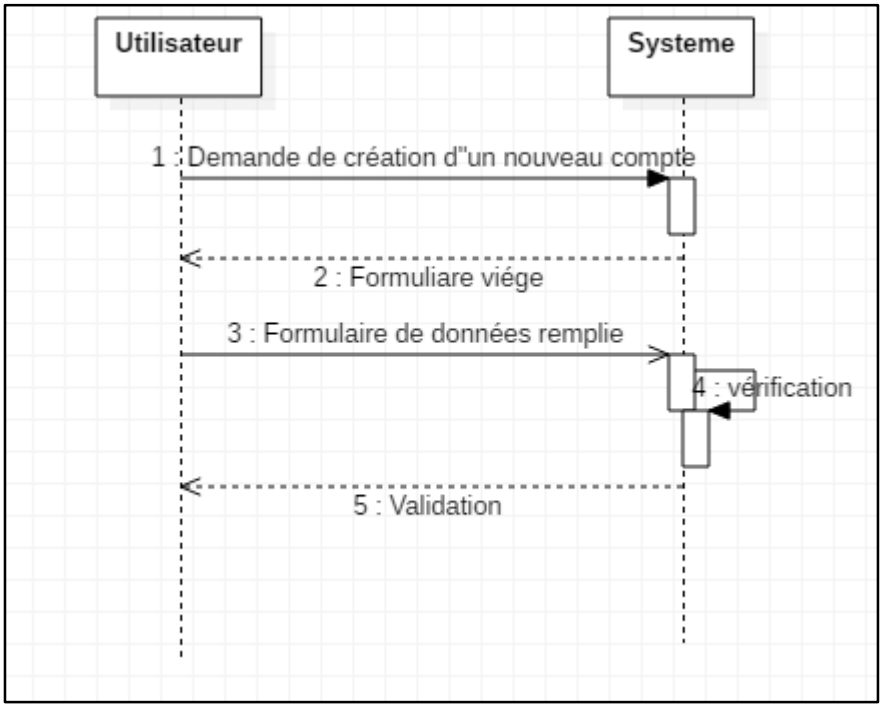


Figure 3.12 : Diagramme de séquence « Création compte utilisateur »

- **Diagramme de séquence 2 « connexion à un compte (Authentification) »**.

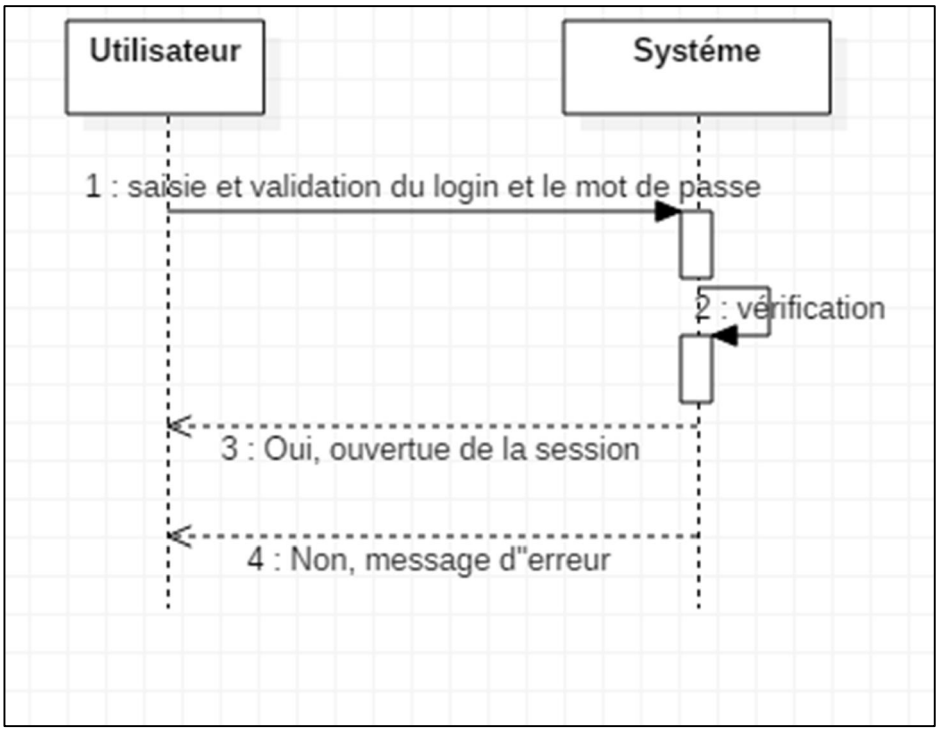


Figure 3.13: Diagramme de séquence « connexion à un compte (Authentification) ».

- **Diagramme de séquence 3 « Transformation du fichier Excel en XML »**

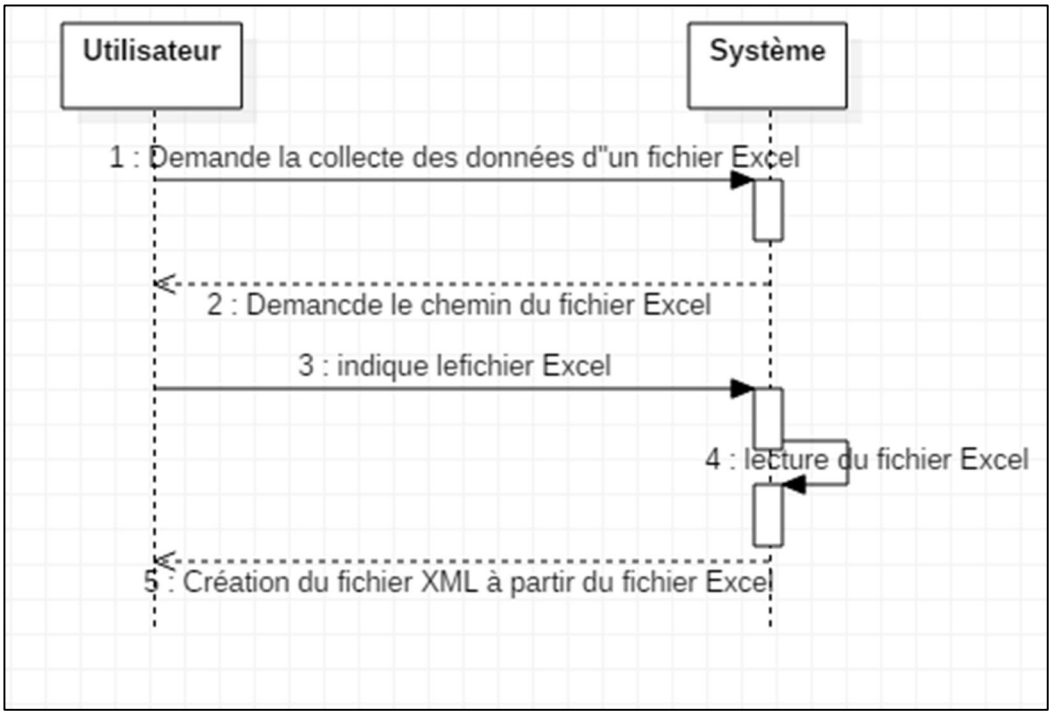


Figure 3.14 : Diagramme de séquence « Transformation du fichier Excel en XML »

- **Diagramme de séquence 4 « Gestion du formulaire de saisie »**

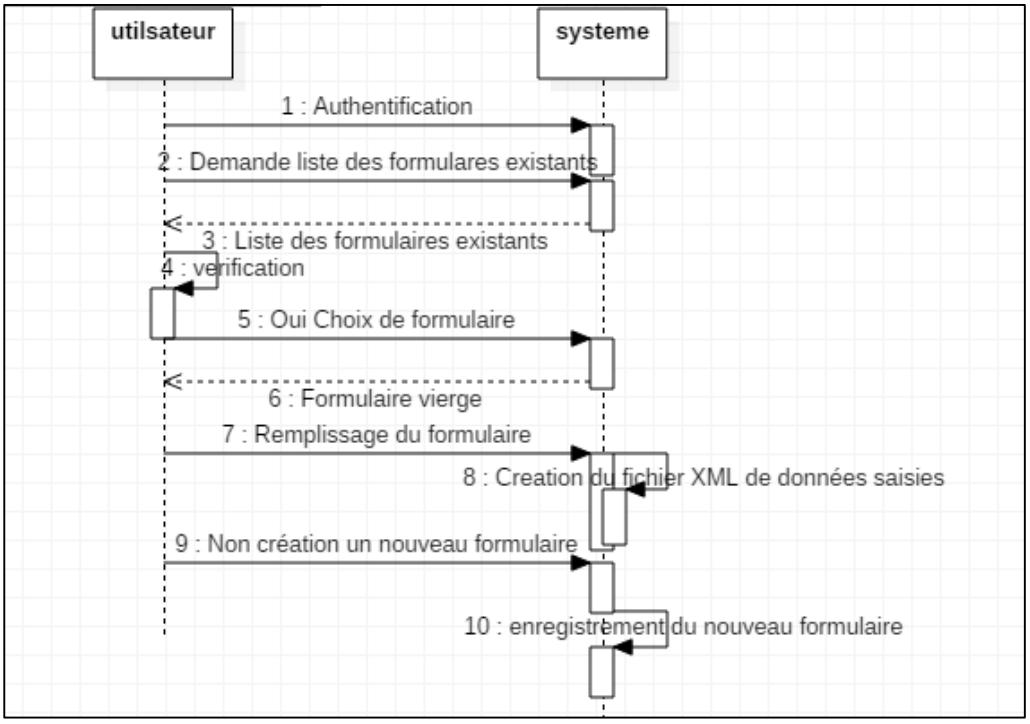


Figure 3.15 : Diagramme de séquence « Gestion du formulaire de saisie »

- **Diagramme de séquence 5 « Extraction de la structure d'une base de données »**

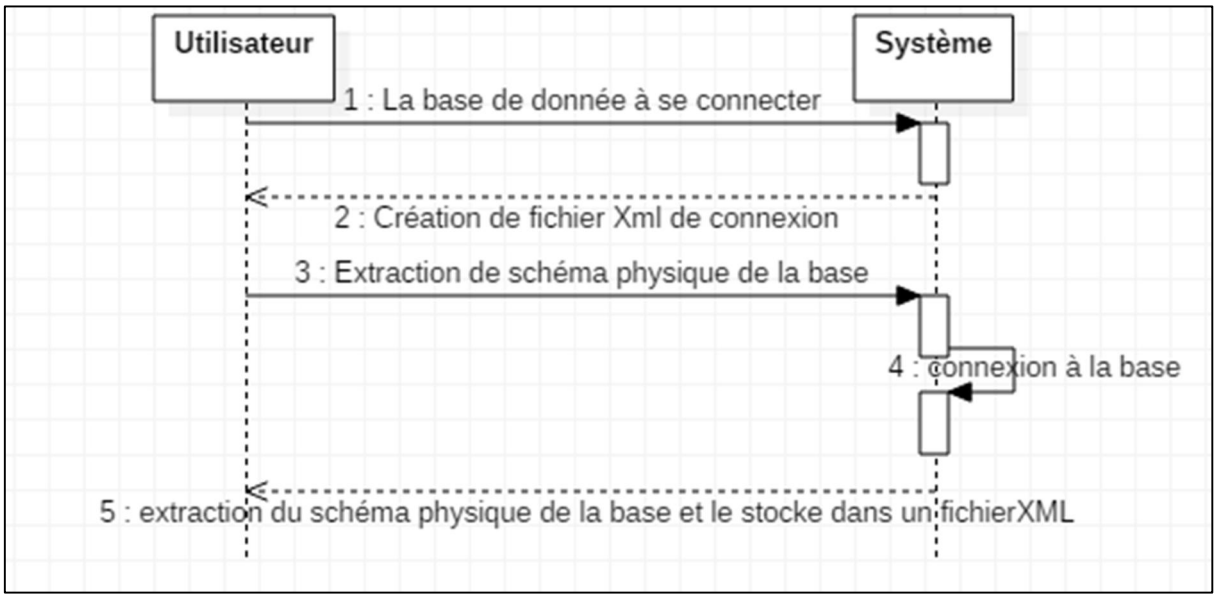


Figure3.16 : Diagramme de séquence « Extraction de la structure d'une base de données ».

- **Diagramme de séquence 6 « Etablissement des correspondances »**

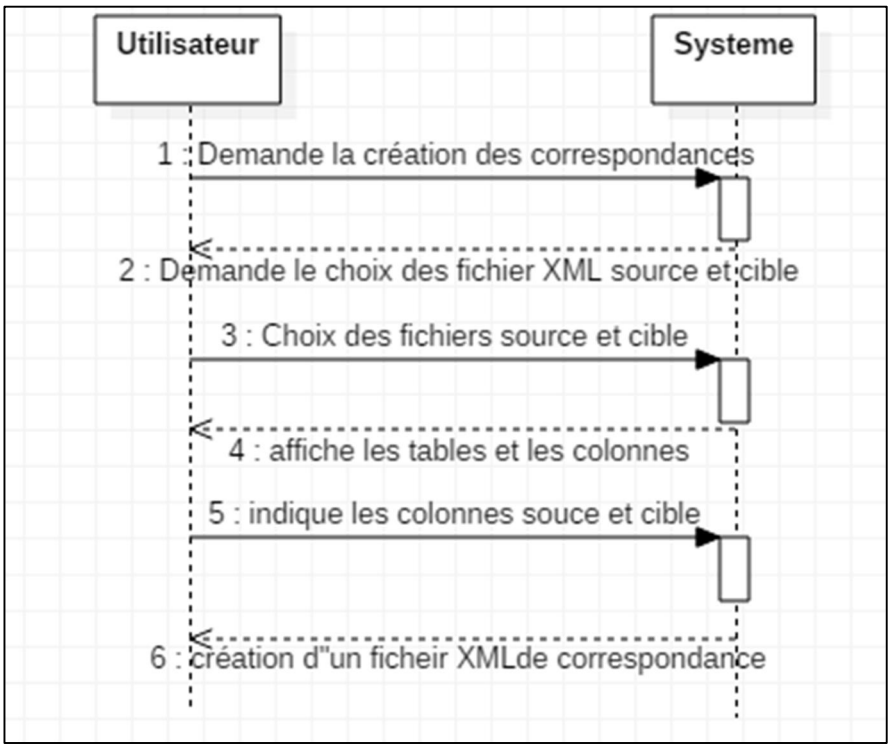


Figure 3.17 : Diagramme de séquence « Etablissement des correspondances »

- **Diagramme de séquence 7 « Collecte des données »**

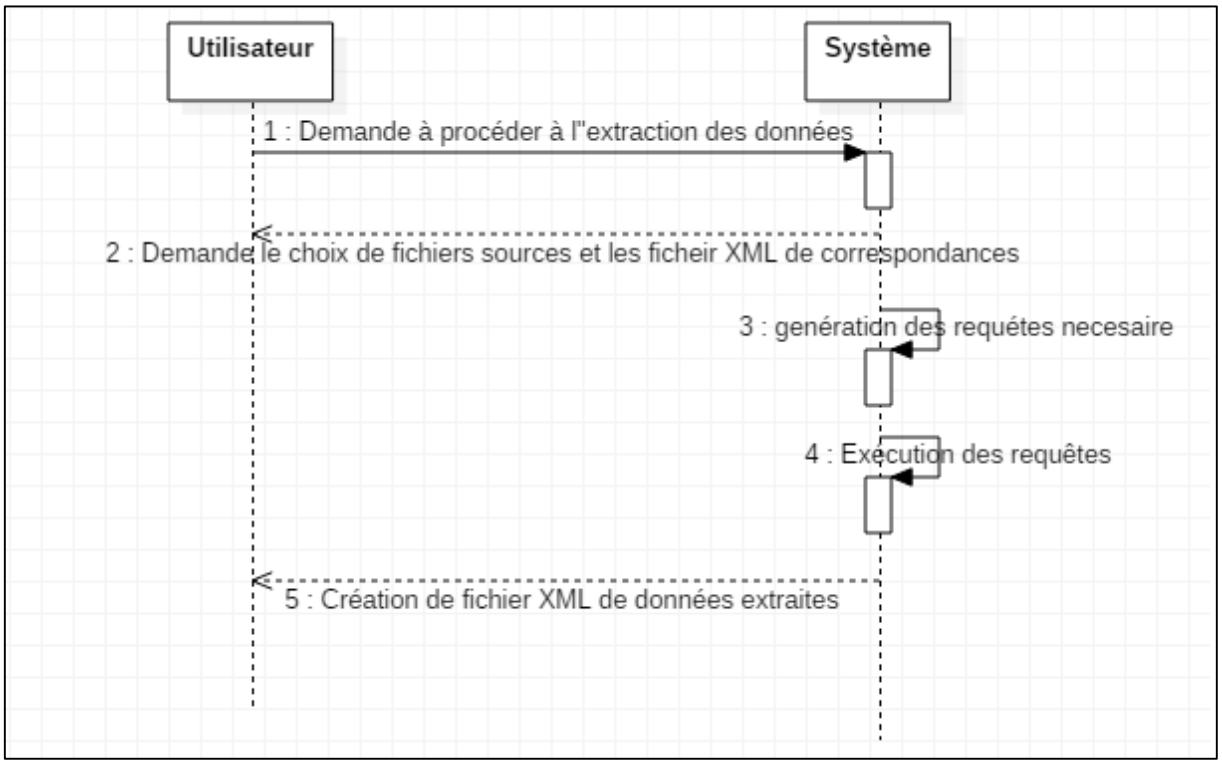


Figure 3.18 : Diagramme de séquence « Collecte des données »

Chapitre IV: Implémentation

1- Le choix technique

1-1- Langage de programmation

L'HTML est le format de données conçu pour représenter les pages web. C'est un langage de balisage permettant d'écrire de l'hypertexte, d'où son nom. HTML permet également de structurer sémantiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie, et des programmes informatiques. Il permet de créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web. Il est souvent utilisé conjointement avec des langages de programmation (JavaScript) et des formats de présentation (feuilles de style en cascade). HTML est initialement dérivé du Standard Generalized Markup Language(SGML) [22].

PHP est un langage de programmation web côté serveur, ce qui veut dire que c'est le serveur qui va interpréter le code PHP et générer du code qui pourra être interprété par votre navigateur. Pour décrire une page PHP, on pourrait dire que c'est un fichier avec l'extension .php, lequel contient une combinaison de balises HTML et de scripts qui tournent sur un serveur web [19].

JavaScript est un langage de programmation principalement utilisé pour créer des pages web interactives. Ce langage, incorporé dans un document HTML, n'est pas visible dans la fenêtre du navigateur. Il sert à améliorer le Langage html en effet, il permet d'exécuter des commandes du côté client (c'est-à-dire au niveau du navigateur et non du serveur web). Ce code qui est exécuté par le navigateur Web est utile pour toutes les interactions du client sur la page Web. Ce langage permet de manipuler des objets au sens informatique : créer des fenêtres spécifiques, contrôler les données saisies dans les formulaires, redimensionner certains objets, rediriger des liens… [20].

1-2 XML

Le système créé et analyse des documents XML :

- fichiers de métadonnées,

- fichiers de données extraites et transformées,
- fichiers de définitions des correspondances.

On peut manipuler XML de différentes manières

- A la main
- Avec XMLWriter/XMLReader
- Avec DOM
- Avec SimpleXML

ou des combinaisons de ces méthodes.

- DOM ou Document Object Model est une méthode qui recrée toute l'arborescence d'un document XML sous forme d'objets PHP. Son utilisation est simple mais elle est couteuse en ressources, en temps d'exécution et un peu verbeuse.
- XMLWriter et XMLReader traitent les fichiers XML à plus bas niveau mais leur utilisation exclusive rend parfois le code délicat à implémenter surtout pour la lecture de fichiers complexes.
- SimpleXML représente une sorte de compromis Simplicité/Performance.
- Traiter des fichiers xml « à la main » est généralement à éviter sauf pour créer des fichiers très simples.

2- Les fonctionnalités du système

2-1- Module de transformation du fichier Excel en XML

Afin de convertir un fichier de format Excel en format XML on doit passer par le format CSV qui est un format d'échange de données situés au même niveau qu'un tableur et permettant de transmettre des informations d'une application ou d'un serveur à l'autre.

- **Conversion d'un XLSX en CSV**

Pour convertir un fichier XLS en CSV on utilise la bibliothèque PHPEXCEL qui est une bibliothèque open source permet de lire et d'écrire dans des tableurs, XLS et XLSX. Mais il peut aussi générer des CSV, des PDF, et des HTML.

Elle comprend toute sorte de fonctions de manipulations de tableurs, telles que le changement de couleur des champs, l'ajout de graphiques et de filtres, la protection de feuilles...

```php
function xlsXmlFile($namefile) {

//conveert xls to xml
//conveert xls to csv

 $objReader = PHPExcel_IOFactory::createReader('Excel2007');
$objPHPExcel = $objReader->load('../FileExcel/'.$namefile.'.xlsx');
$xlsx = PHPExcel_IOFactory::load('../FileExcel/'.$namefile.'.xlsx');
$writer = PHPExcel_IOFactory::createWriter($xlsx, 'CSV');
$writer->setDelimiter(";");
$writer->setEnclosure("");
$writer->save('../FileExcel/'.$namefile.'.csv');
```

- **Conversion d'un CSV en XML**

Bien que le CSV soit en format ASCII, il en demeure assez difficile à lire, toutefois on peut le convertir sans problème majeur en format XML pour rendre plus compréhensible les données. L'ennui avec le format CSV c'est qu'il n'est pas vraiment standard, ainsi parfois le séparateur, est un caractère «,», «;», «|». De plus, parfois les chaines de caractères sont situées entre guillemets, d'autre fois ce sont uniquement le texte qu'il l'est. Le script suivant permet de convertir un fichier CSV (source.csv) en fichier XML (dest.xml) :

```
$Header = true;
$Separator = '|';
$I = 0;
if($fileRead = fopen("../FileExcel/".$namefile.".csv", 'r')) {
    $fileWrite = fopen('../FileExcel/'.$namefile.'.xml', 'w');
    fwrite($fileWrite, '<?xml version="1.0" encoding="ISO-8859-15"?>'."\n");
    fwrite($fileWrite, '<root>'."\n");
    $ColumnName = array();
    while($CurrLine = fgets($fileRead)) {
        if($Header && ($I == 0)) {
            for($Cellule = '', $K = 0,$J = 0; $J < strlen($CurrLine); $J++) {
                if($CurrLine[$J] == $Separator) {
                    $ColumnName[$K] = trim($Cellule);
                    $Cellule = '';
                    $K++;
                } else {
                    if(preg_match('/^[a-zA-Z0-9]+$/', $CurrLine[$J])) $Cellule .= $CurrLine[
                }
            }
            $ColumnName[$K++] = trim($Cellule);
            $I++;
        } else if($CurrLine != '') {
            fwrite($fileWrite, '<entry>'."\n");
            for($Cellule = '', $K = 0,$J = 0; $J < strlen($CurrLine); $J++) {
                if($CurrLine[$J] == $Separator) {
                    if($Header) {
                        fwrite($fileWrite, '<'.$ColumnName[$K].'>'.$Cellule.'</'.$ColumnName
                    } else {
                        fwrite($fileWrite, '<cellule'.$K.'>'.$Cellule.'</cellule'.$K.'>'."\n
                    }
                    $Cellule = '';
                    $K++;
                    ",
                } else {
                    if(ord($CurrLine[$J]) >= 32) $Cellule .= $CurrLine[$J];
                }
            }
            if($Header) {
                fwrite($fileWrite, '<'.$ColumnName[$K].'>'.$Cellule.'</'.$ColumnName[
            } else {
                fwrite($fileWrite, '<cellule'.$K.'>'.$Cellule.'</cellule'.$K.'>'."\n"
            }
            fwrite($fileWrite, '</entry>'."\n");
            $I++;
        }
    }
    fwrite($fileWrite, '</root>'."\n");
    fclose($fileWrite);
    fclose($fileRead);
}
}
```

2-2- Gestion des formulaires :

Le système permet de consulter l'ensemble des formulaires existant sur le serveur, la création des nouveaux formulaires de saisies ainsi que la collecte des données du formulaire.

- **Consulter les formulaires existants**

```php
$dirname = '../formulaire';
$dir = opendir($dirname);

while($file = readdir($dir)) {
if($file != '.' && $file != '..' && !is_dir($dirname.$file))
{
echo '<a href="'.$dirname.'/'.$file.'">'.$file.'</a>'.'<br /><br />';

}
}

closedir($dir);
```

- **Création des nouveaux formulaires**

```php
if (isset($_POST['submit']))

{
    $c=isset($_POST['textarea']) ? $_POST['textarea'] : NULL;
    $nomf=isset($_POST['nomFor']) ? $_POST['nomFor'] : NULL;
    $h=".php";
    $ch= "../formulaire/";
    $codes=" codec();";
    $nf=$ch."".$nomf."".$h  ;
    //echo $nf;
    //création du fichier
$f = fopen($nf, "x+");
// écriture
fputs($f, '<html> <head> </head> <body>');
fputs($f, $c);
//fputs($f, $codes);
fputs($f, '</body> </html>');
// fermeture
fclose($f);
```

- **Collecter les données saisies par l'utilisateur et l'enregistrer sous format XML**

```
$xml_doc = new DOMDocument('1.0', 'utf-8');
 $xml_doc->formatOutput=TRUE;
 $racine=$xml_doc ->createElement($nomF);
 $xml_doc->appendChild($racine);
 $a=$xml_doc->createElement($nomF);
 $racine->appendChild($a);
  $i = 0 ;
  $i =$i+1;
   while ($i <= $nbre){
      $champs= "champs"."".$i;
       $Nchamps="nchamps"."".$i;
$nomChamps=isset($_POST[$Nchamps]) ? $_POST[$Nchamps] : NULL;
$valChamps=isset($_POST[$champs]) ? $_POST[$champs] : NULL;
  $aa =$xml_doc->createElement($nomChamps, $valChamps);
  $a->appendChild($aa);
  //$a->appendChild($xml_doc->createElement($nomChamps, $valChamps));
  $i++;}
  $dossier ='../'."".$nomF."";
if(!is_dir($dossier)){
   mkdir($dossier);}
$chemin = $dossier.'/'."".$nomF."".'.xml';
  //$chemin='../Vue/'."".$nomF."".'.xml';
  $xml_doc->save( $chemin);
```

2-3- Gestionnaire de base de données

Le gestionnaire de base de données est chargé de représenter en mémoire le schéma physique des tables sources ou cibles concernées.

Ce module est appelé à deux niveaux :

- lors de l'extraction des schémas physiques des bases,
- lors de la phase de définition des correspondances.

Extraction des schémas physiques

A ce niveau, les informations qui nous intéressent sont celles qui définissent le schéma physique des bases : liste des tables, caractéristiques de ces tables (nom, encodage, …), liste des colonnes des tables, paramètres de ces colonnes (nom, type, taille, …). Dans le monde des systèmes de gestion de bases de données, ces informations sont communément appelées métadonnées.

Une fois que l'utilisateur a défini une connexion à une base de données, le système en extrait les métadonnées et en déduit les bases, tables, et colonnes qui la composent. Lors de cette phase, une instance de la classe BDD est instanciée pour chaque base de données. Le système crée pour chaque table de la base de données un objet Table lié à l'objet BDD concerné et pour chaque colonne un objet Colonne lié à l'objet Colonne concerné.

Dans un second temps, le schéma extrait est stocké dans des fichiers XML.

Le gestionnaire de représentations s'appuie alors sur le gestionnaire de documents XML.

Un extrait de fichier XML représentant le schéma physique d'une base de données généré par

Le système est présenté sur la figure ci-dessous :

```
<table name="centre">
<colonne name="centre_id" type="VARCHAR" size="10" nullable="0"
isAutoIncrement="NO"/>
<colonne name="nom" type="VARCHAR" size="20" nullable="1"
isAutoIncrement="NO"/>
<colonne name="adresse" type="VARCHAR" size="255" nullable="1"
isAutoIncrement="NO"/>
</table>
<table name="medecin">
<colonne name="medecin_id" type="VARCHAR" size="10" nullable="0"
isAutoIncrement="NO"/>
<colonne name="centre_id" type="VARCHAR" size="20" nullable="1"
isAutoIncrement="NO"/>
<colonne name="nom" type="VARCHAR" size="20" nullable="1"
isAutoIncrement="NO"/>
<colonne name="prenom" type="VARCHAR" size="20" nullable="1"
isAutoIncrement="NO"/>
</table>
```

Figure : **extrait d'un schéma d'une base de données généré par le système**

Définition des correspondances

Lors de la phase de définition des correspondances, le gestionnaire de base de données parcourt le fichier XML du schéma physique des bases de manière séquentielle et les objets (Bdd, Table, Colonne) créés au fur et à mesure des éléments rencontrés, ensuit il gère les correspondances définies par l'utilisateur. En fonction de ces correspondances, le système va créer et appliquer des transformations sur les données.

Un objet de type correspondance est créé pour chaque correspondance définie par l'utilisateur entre les schémas source et cible. Dans le cas d'une correspondance complexe, plusieurs objets de type correspondances peuvent être instanciés, le système stocke la description de l'ensemble des correspondances définies sous la forme d'une fichier XML.

Extraction des données / transformation

Le système établit des ordres SQL nécessaires. Ces ordres SQL sont des requêtes de type *SELECT* qui vont permettre d'extraire et de transformer les données des bases sources selon les correspondances définies par l'utilisateur.

- **Exemple de requêtes simples**

Considérons l'ordre SQL suivant :

```
SELECT liste_de_colonnes FROM liste_de_tables
```

liste_de_colonnes liste les colonnes du résultat de l'exécution de l'ordre et *liste_de_tables* liste les tables sur lesquelles porte cet ordre.

L'exécution de cette simple requête permet d'extraire d'une base de données les données des colonnes *liste_de_colonnes* des tables *liste_de_tables*.

Il est possible de définir explicitement un nom pour la, ou les, colonne(s) de ces tables de résultat en utilisant l'opérateur *AS* comme ci-dessous :

```
SELECT nomColonne AS nomColonneResultat FROM table
```

nomColonne désigne la colonne à interroger dans la table *table* de notre base de données.

nomColonneResultat désigne le nom de la colonne que nous donnons au résultat obtenu.

Dans la plupart des SGBD *nomColonne* peut être écrit sous sa forme absolue, ou étendue, soit : *nomSchema.nomTable.nomColonne*. La colonne *nomColonne* se trouve dans la table

nomTable qui est elle-même définie dans le schéma *nomSchéma*.

Ces requêtes utilisent l'opérateur *AS* pour définir pour chaque colonne résultat un nom unique et ainsi éviter d'obtenir deux colonnes de même nom (qui pourraient provenir de deux tables différentes) dans un même résultat.

Ce type de requête, dite requête simple, est généré par le système lorsque la transformation à effectuer est de type atomique.

Une correspondance simple est présentée ci-dessous :

```
<correspondance>
<id>2</id>
<lotId>0</lotId>
<dataSrcName>bdd_src</dataSrcName>
<tableSrcName>centre</tableSrcName>
<colonneSrcName>nom</colonneSrcName>
<tableDestName>centre</tableDestName>
<colonneDestName>nom</colonneDestName>
</correspondance>
```

Cette correspondance génère la requête suivante :

```
SELECT bdd_src.centre.nom
```

```
AS bdd_src_centre_nom
FROM bdd_src.centre
```

- **Exemple de requête de type calcul**

Il est possible en SQL d'effectuer certains calculs mathématiques de base sur les données des colonnes, comme ci-dessous :

```
SELECT libellé, prix*1.3508 FROM produit
```

Supposons que `1,3508` soit le taux de conversion actuel de l'euro vers le dollar.
En exécutant cette requête sur une table `produit` contenant le `libellé` du produit associé à un `prix` en euros, nous obtiendrions l'équivalent en dollars US.
Cette correspondance est présentée ci-dessous.

```
<correpondance>
<id>1</id>
<lotId>0</lotId>
<dataSrcName>bdd_src</dataSrcName>
<tableSrcName>produit</tableSrcName>
<colonneSrcName>prix</colonneSrcName>
<tableDestName>produit</tableDestName>
<colonneDestName>prix</colonneDestName>
<calc>
<type>mutliply</type>
<value>1.3508</value>
</calc>
</correspondance>
```

Cette correspondance génère la requête suivante :

```
SELECT bdd_src.produit.prix*1.3508
AS bdd_src_produit_prix
FROM bdd_src.produit
```

- **Exemple d'une requête avec jointure**

Une jointure est un sous-ensemble d'un produit cartésien de deux ou plusieurs tables.

SQL permet d'effectuer des jointures entre tables, voire, pour certains SGBD, entre schémas.

Considérons les deux schémas de tables suivants :

```
Patient (nom_patient varchar(255), nationalite_id int(10))
Pays (pays_id int(10), nom_pays varchar(10))
```

Supposons que `nationalite_id` est une clé étrangère référençant `pays_id`.

Nous pouvons alors écrire la requête suivante :

```
SELECT nom_patient, nom_pays
FROM Patient LEFT JOIN pays ON nationalite_id=pays_id
```

Cette requête permet d'obtenir une liste des noms des patients de la table patient avec, pour chacun, le nom du pays de leur nationalité.

Voici un exemple de déclaration d'une correspondance de type jointure

```
<correspondance>
<id>1</id>
<lotId>0</lotId>
<dataSrcName>bdd_src</dataSrcName>
<tableSrcName>patient</tableSrcName>
<colonneSrcName>nom_patient</colonneSrcName>
<tableDestName>patient</tableDestName>
<colonneDestName>nom</colonneDestName>
</correspondance>
<correspondance>
<id>2</id>
<lotId>0</lotId>
<dataSrcName>bdd_src</dataSrcName>
<tableSrcName>pays</tableSrcName>
<colonneSrcName>nom_pays</colonneSrcName>
<tableDestName>patient</tableDestName>
<colonneDestName>nationalite</colonneDestName>
</correspondance>
<reference>
<refOrigSchema>bdd_src</refOrigSchema>
<refOrigtable>patient</refOrigtable>
<refOrigColonne>nationalite_id</refOrigColonne>
<refCibleSchema>bdd_src</refCibleSchema>
<refCibletable>pays</refCibletable>
<refCibleColonne>nom_pays</refCibleColonne>
</reference>
```

Cette correspondance génère la requête suivante :

```
SELECT bdd_src.patient.nom AS bdd_src_patient_nom, bdd_src.pays.nom
AS bdd_src_pays_nom
FROM bdd_src.patient
LEFT JOIN bdd_src.pays
on bdd_src.patient.nationalite_id= bdd_src.pays.pays_id;
```

3- Représentation de l'interface graphique du système

Dans ce qui suit nous allons présenter l'interface graphique du système de collecte automatique de données.

3-1- Page d'accueil

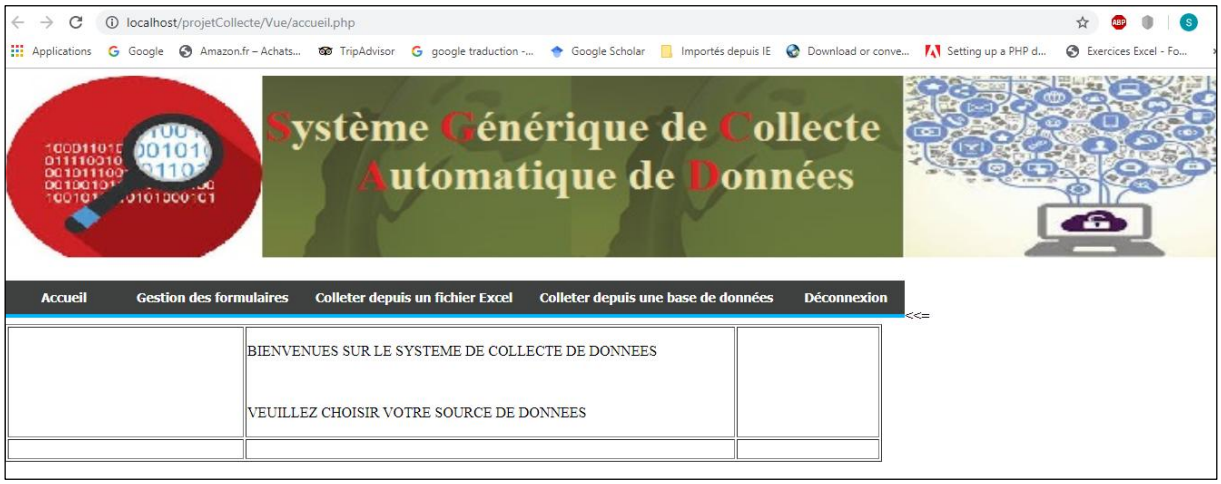


Figure 4.1 : La page d'accueil du système

3-2- Gestion des formulaires

- **Créer un nouveau formulaire**

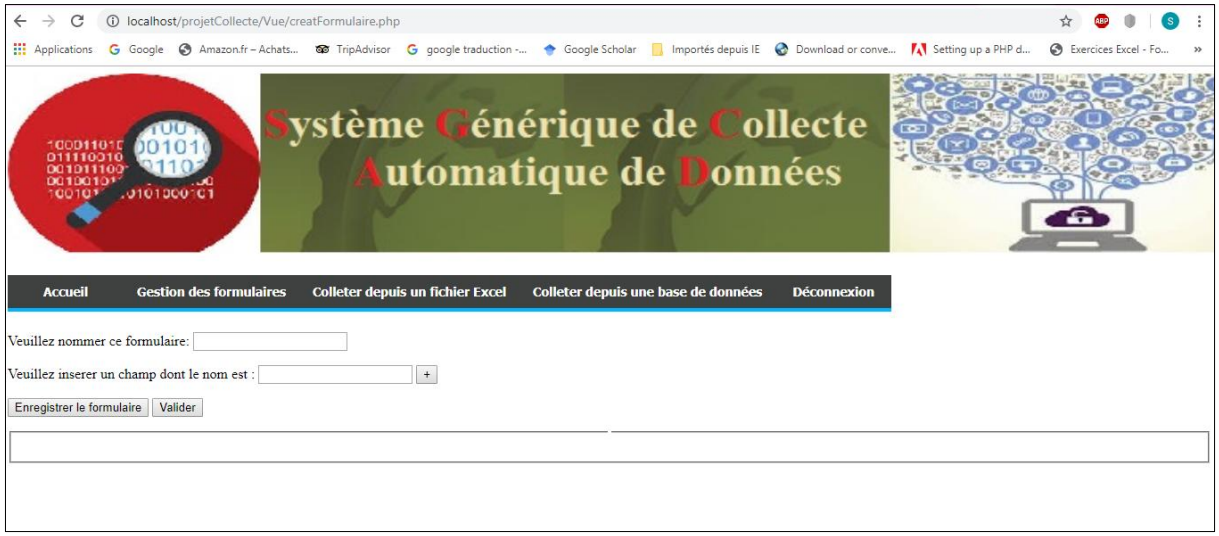


Figure 4.2 : interface de création d'un formulaire

Pour créer un nouveau formulaire on doit suivre les étapes suivantes :

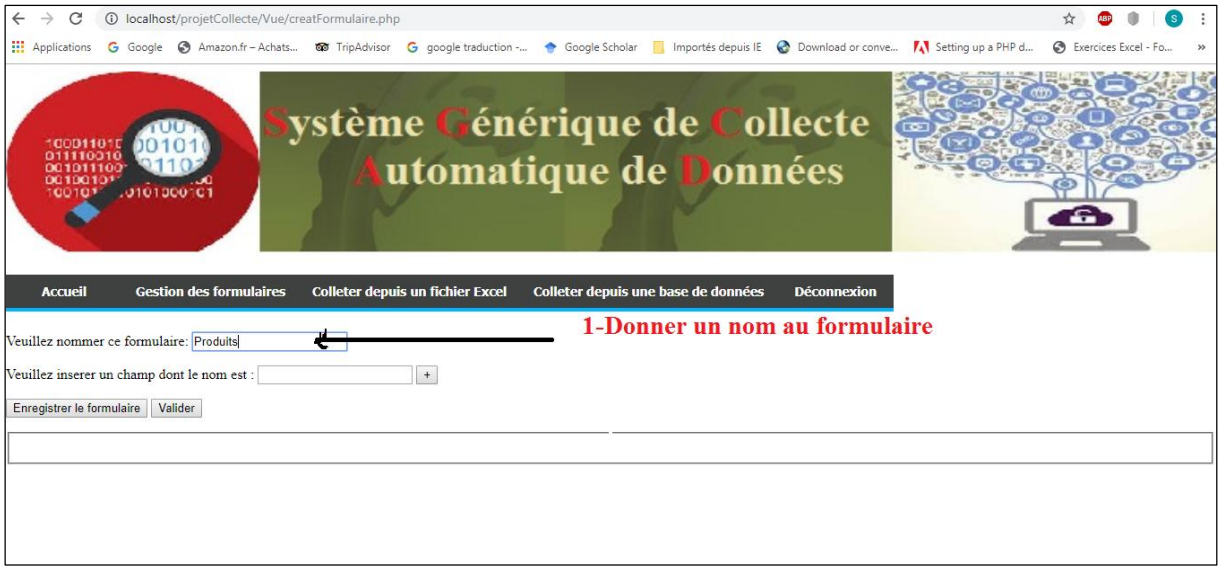


Figure 4.3 : Nommer un formulaire

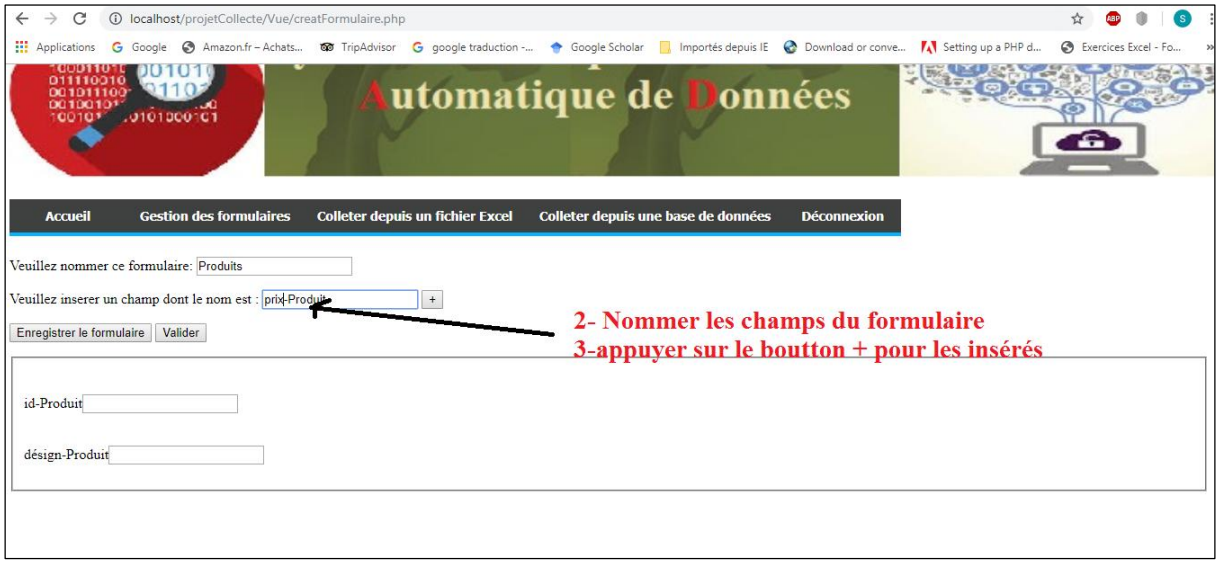


Figure 4.4 : Insertion des champs

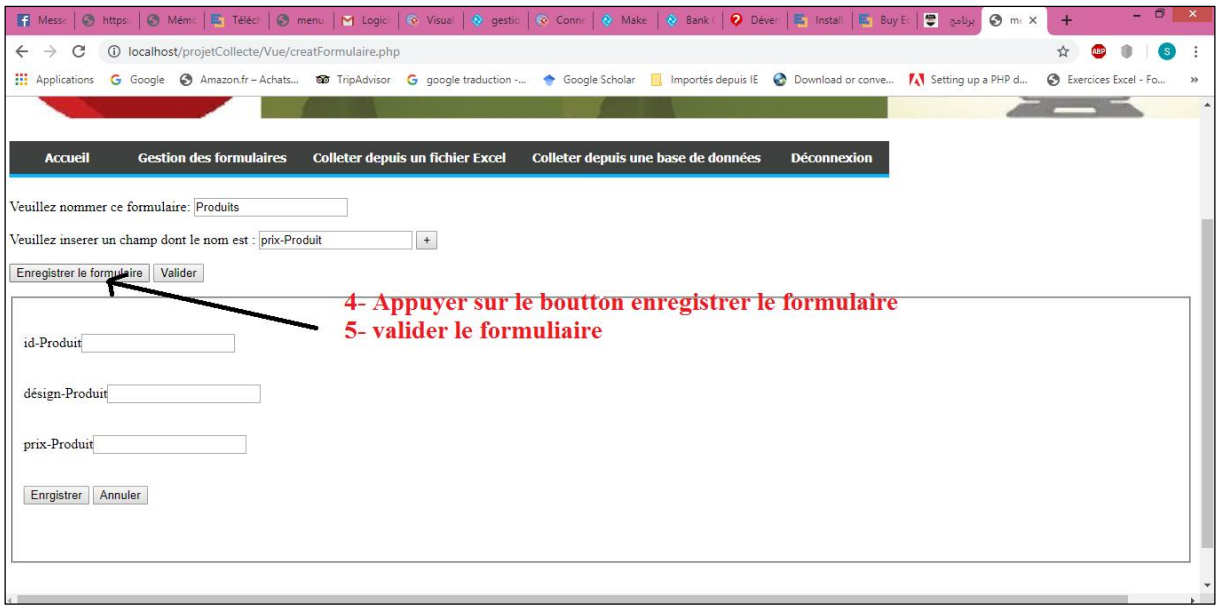


Figure 4.5 : interface de validation

Après la validation le nouveau formulaire va être créé, on peut le consulter dans la liste des formulaires existants.

- **Consulter les formulaires existants**

Figure 4.6 : La liste des formulaires existants sur le système

En appuyant sur le nom du formulaire, on aura :

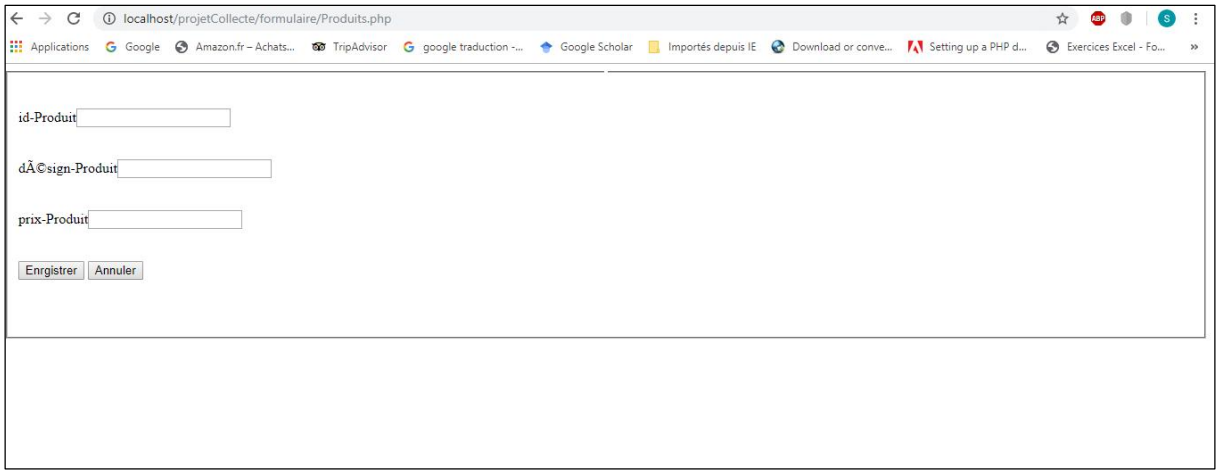


Figure 4.7 : un formulaire de saisie

L'utilisateur remplis ensuite les champs du formulaire :

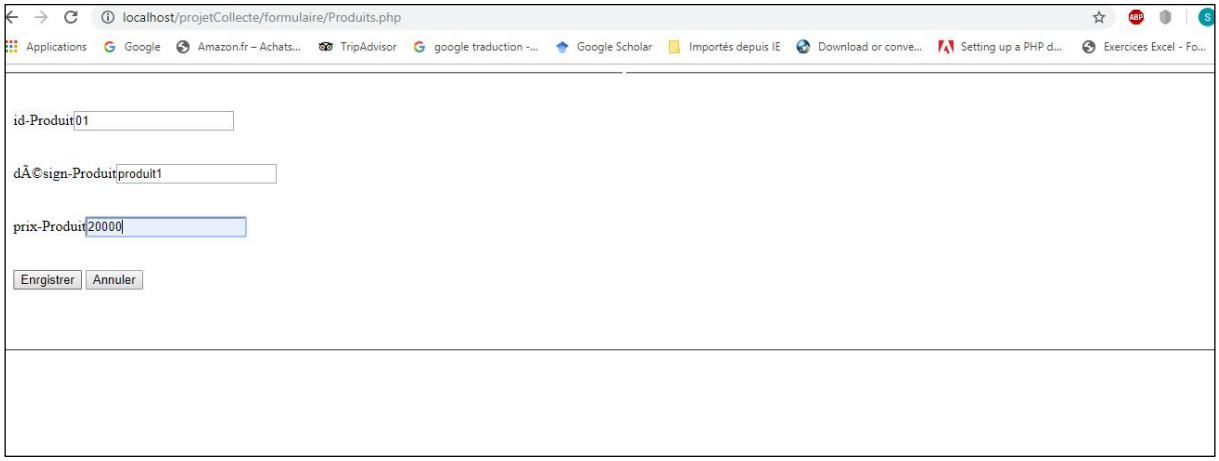


Figure 4.8 : le remplissage du formulaire

Puis il appuie sur le bouton Enregistrer , le système génère automatiquement un fichier XML

Figure 4.9 : le fichier XML contenant les données du formulaire

```
<?xml version="1.0" encoding="utf-8"?>
<Produits>
  <Produits>
    <id-Produit>01</id-Produit>
    <dÃ©sign-Produit>produit1</dÃ©sign-Produit>
    <prix-Produit>20000</prix-Produit>
  </Produits>
</Produits>
```

4-3-1- Collecter les donner depuis un fichier Excel

L'utilisateur doit choisir un fichier Excel en appuyant sur le bouton choisir un fichier

Figure 4.10 : L'interface de collecte de donnée d'un fichier Excel

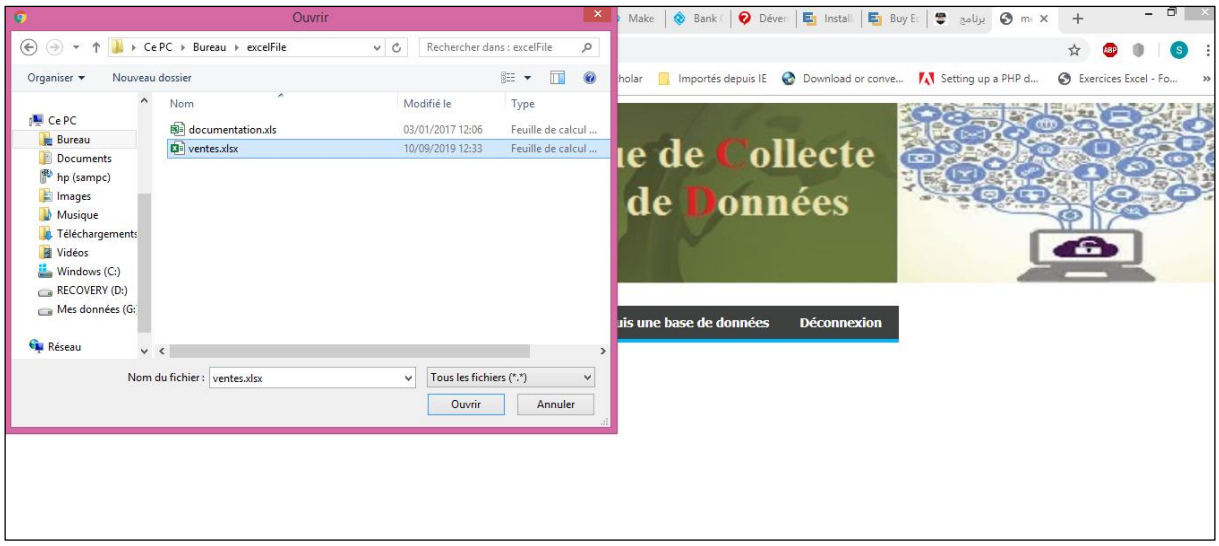


Figure 4.11 : Le choix du fichier Excel

Le système va télécharger le fichier Excel

Figure 4.12 : le téléchargement du fichier Excel par le système

En appuyant sur le bouton Envoyer le fichier le système va convertir automatiquement le fichier Excel en XML

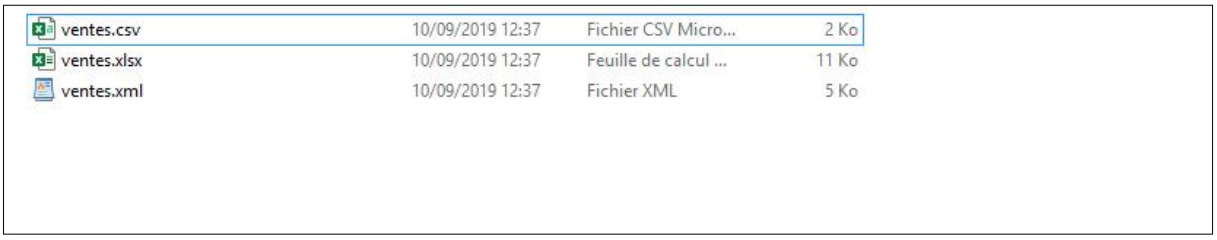


Figure 4.13 : Les fichiers .CSV et . XML crée à partir d'un fichier Excel

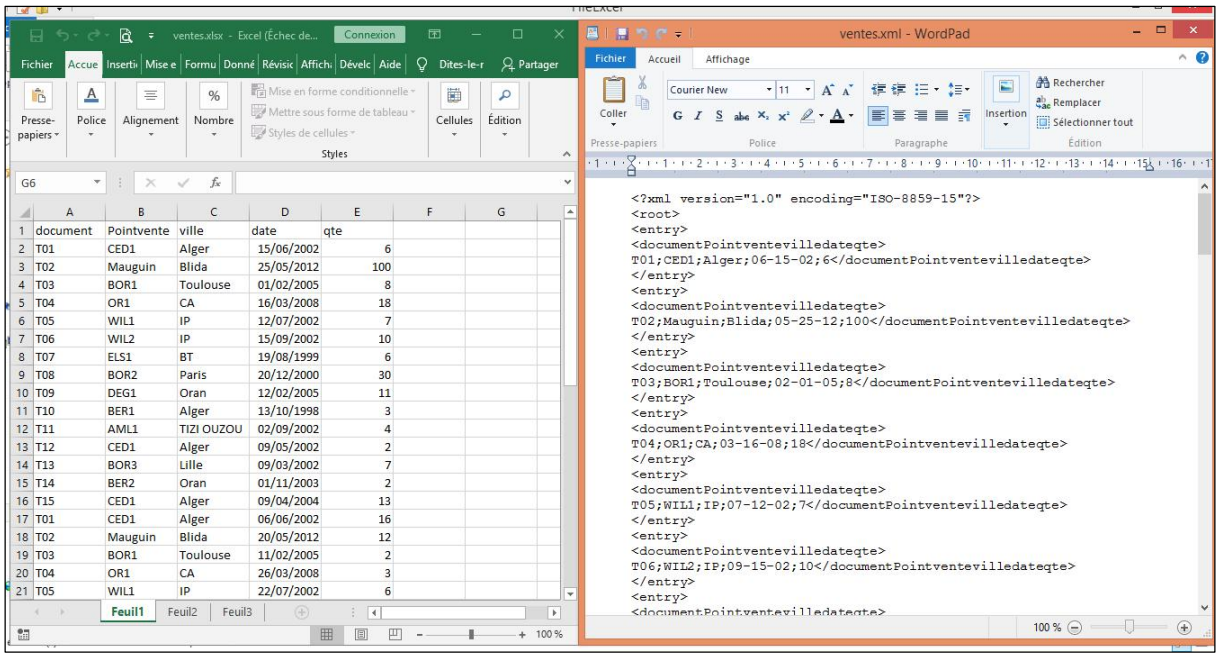
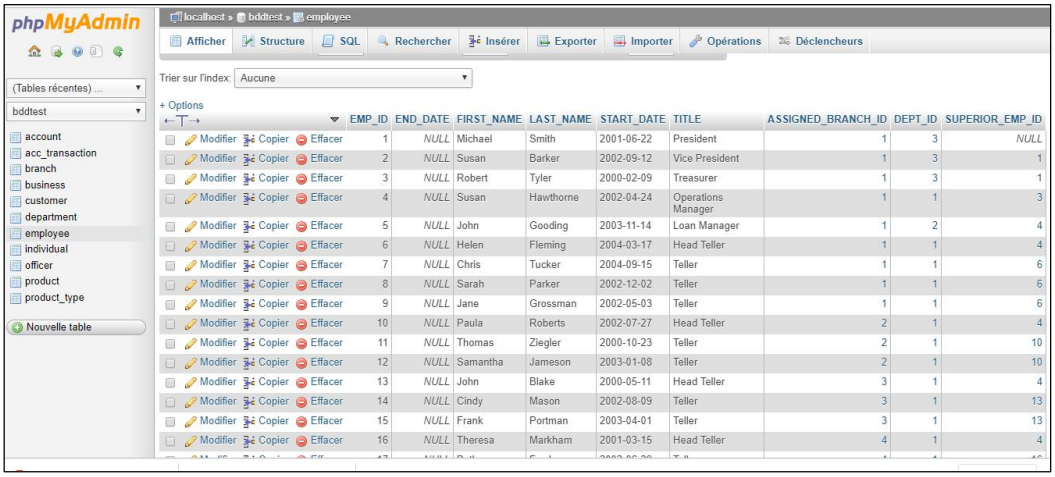
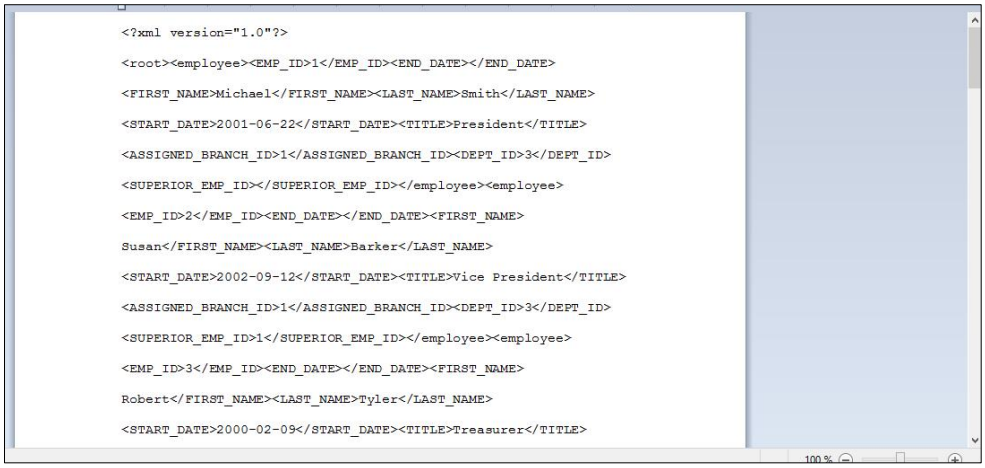


Figure 4.14 : Un aperçu de fichier Excel et son correspondant en XML

3-3- Collecter les donner depuis une base de données

Dans cette partie on a fait des tests sur une BDD (bddtest) mais sur une table de la base (table employee) le système à créer un fichier XML à partir de cette table :





```
<?xml version="1.0"?>
<root><employee><EMP_ID>1</EMP_ID><END_DATE></END_DATE>
<FIRST_NAME>Michael</FIRST_NAME><LAST_NAME>Smith</LAST_NAME>
<START_DATE>2001-06-22</START_DATE><TITLE>President</TITLE>
<ASSIGNED_BRANCH_ID>1</ASSIGNED_BRANCH_ID><DEPT_ID>3</DEPT_ID>
<SUPERIOR_EMP_ID></SUPERIOR_EMP_ID></employee><employee>
<EMP_ID>2</EMP_ID><END_DATE></END_DATE><FIRST_NAME>
Susan</FIRST_NAME><LAST_NAME>Barker</LAST_NAME>
<START_DATE>2002-09-12</START_DATE><TITLE>Vice President</TITLE>
<ASSIGNED_BRANCH_ID>1</ASSIGNED_BRANCH_ID><DEPT_ID>3</DEPT_ID>
<SUPERIOR_EMP_ID>1</SUPERIOR_EMP_ID></employee><employee>
<EMP_ID>3</EMP_ID><END_DATE></END_DATE><FIRST_NAME>
Robert</FIRST_NAME><LAST_NAME>Tyler</LAST_NAME>
<START_DATE>2000-02-09</START_DATE><TITLE>Treasurer</TITLE>
```

Conclusion générale

Les technologies réseau actuelles (Intranet, Internet, etc.) permettent l'accès à une immense quantité de données stockées à différents emplacements physiques, le principal intérêt de ces technologies pour l'utilisateur est la possibilité d'accéder à une multitude de sources et de les combiner afin d'obtenir l'information désirée.

Un système de collecte de données a pour but de combiner les données réparties dans différentes sources, qui ont été conçues indépendamment les unes des autres, et de fournir à l'utilisateur une vue unifiée de ces données.

Le but fixé au début de ce travail était de développer un système de collecte automatique de données à partir d'un formulaire de saisie, un fichier Excel et une base de données et ce n'est pas le développement d'un ETL (la partie chargement de données n'est pas pris en charge).

Au début de ce travail on a étudié quelques travaux connexes ayant abordé la conception du processus ETL et on a adopté une approche qui se base essentiellement sur la correspondance entre la source et la cible de données où l'utilisateur définit des correspondances entre les schémas physiques des bases sources et celui de la base cible, résolvant ainsi l'hétérogénéité sémantique des données et en utilisant des fichier XML comme support d'échange. Notre contribution consiste à généraliser cette contribution et l'appliquer sur des fichiers Excel et des formulaires de saisie et non seulement sur les bases de données relationnels et pour réaliser cette généralisation on a développé un module qui convertit les fichiers Excel en XML et on a donné la main aux utilisateurs du système à créer leurs propres formulaires de saisie et d'enregistrer les donner sous format XML.

La réalisation de système de collecte automatique de données m'a permis de me perfectionner dans l'utilisation des technologies PHP, javascript, et XML et j'essaie maintenant, dans le cadre de mon travail, de pousser à l'utilisation de ces technologies lorsque celles-ci sont adaptées aux besoins. J'ai aussi amélioré ma maîtrise de différents outils de développement.

Ce travail ouvre des perspectives à court et à long terme. Je souligne dans ce qui suit les perspectives qui me semble pertinentes pour l'évolution du système :

- Premièrement, il serait intéressant de développer un module de gestion d'erreurs avancé qui permettra d'identifier les données qui n'auraient pas été correctement traitées lors de l'exécution du processus du collecte de données.
- Deuxièmement, je propose aussi en perspectives de développer un module de chargement de données collectés dans une base de données cible.
- Troisièmement, puisque ce système utilise des fichiers XML comme support d'échange de données il est envisageable de traiter les requêtes exprimées en XQuery qui est un langage de requêtes permettant d'extraire des informations d'un document XML ou d'une collection de documents XML.

BIBLIOGRAPHIE

[1] M. Ben Taher, H. Ben-Abdallah, *Approche semi automatique pour la génération de procédures ETL*, Troisième Atelier sur les Systèmes Décisionnels ASD 2010, Sfax, November 2010.

[2] D. Skoutas, A. Simitsis, *Designing ETL Processes Using Semantic Web Technologies*, In DOLAP, 2006.

[3] Z. Zhang, S. Wang, *A Framework Model Study for Ontology-driven ETL Processes*, Wireless Communications, Networking and Mobile Computing, 2008. WiCOM08.4th International Conference on, October 2008.

[4] A. Simitsis, D. Skoutas, M. Castellanos, *Natural Language Reporting for ETL Processes,* In DOLAP, 2006.

[5] M. Bala, O. Mokeddem, O. Boussaid, Z. Alimazighi , « *Une Plateforme ETL parallèle et distribuée pour l'intégration de données massives* » , 2015.

[6] H.Laifa, I. Tabiou, *Modélisation du processus ETL au niveau conceptuel, logique et physique,* Université de Larbi Tébessi ,Tébessa, Algérie, Juin 2016.

[7] W.Bakari, M. Ali, H. Ben-Abdallah, *Approche Automatique de Génération des opérateurs*
ETL, Université de Sfax,Tunisia,,……..

[8] C.Gueydan, *XeuTL : « un outil ETL pour l'intégration de données »,* CENTRE D'ENSEIGNEMENT DE GRENOBLE, France, Juin 2010.

[9] F.Z.MARHOUMI ; « *Entrepôts de données XML* » ; Université Libre de Bruxelles, Juin2006.

[10] Le Moigne J.L., « *La théorie du système général, théorie de la modélisation* », P.U.F., 1977.

[11] Abdenour Bouzghoub ; « *Modélisation des Entrepôts de données XML : Application au domaine de la sécurité sociale* » ; Thèse de Magistère Option : SISCSD ; Institut National de Formation en Informatique (I.N.I) 2008.

[12] W. H. Inmon ; « *Building the Data Warehouse Third Edition*» ; Wiley Computer Publishing 2002.

[13] B. Inmon; What is a Data Warehouse; Article; http://www.billinmon.com; 2000.

[14] R. Kimball et M. Ross ; « *Entrepôts de Données : Guide Pratiqu de Modélisation Dimensionnelle 2ème édition* » ; Vuibert 2002.

[15] R. Kimball et J. Caserta ; « *The Data warehouse ETL Toolkit»* ;Wiley Publisshing, INC 2004

[16] O.Capuozzo ; Cas d'utilisation, Article ; https://www.reseaucerta.org; 2004

[17] P.Roques, F. Vallée, « *UML 2 en action, de l'analyse des besoins à la conception J2EE*» Eyrolles, 1ère édition, 2004.

[18] K.Bouzerfrane, « *Extension d'un outil web d'annotation par l'ajout de nouvelles annotations riches et collaboratives »,* Ecole nationale Supérieure d'Informatique (E.S.I),2009 /2010.

[19] http://creer-un-site.fr/sous-categorie-5-le-langage-php.php;

[20] http://www.gralon.net/articles/internet-et-webmaster/creation-site-internet/article-javascript- - presentation-et-applications-1776.htm;

[21] http://fr.wikipedia.org/wiki/JQuery ;

[22]Wikipidia ; JQUIRY. ; http://fr.wikipedia.org/wiki/Hypertext_Markup_Language;

[23] J.F. Goglin; « La Construction du Datawarehouse : du Datamart au Dataweb »;

Hermes 1998.

[24] Les ETL pour les entreprises ; https://www.renaud-dhoker.fr/;

[25] http://sig2010.esrifrance.fr/SIG_et_datawarehouse.aspx

[26] M.BOUNEGAB et F.GHEZAL *« Mise En Œuvre d'une Base de Données OLAP Pour le Décisionnel* », Universite Kasdi Merbah Ouargla,2015-2016