Université Saad DAHLEB de Blida

Faculté des sciences
Département d'Informatique

Mémoire de fin d'études présenté par :

SEDDI Asma & ZIDANE Dahbia

En vue d'obtenir le diplôme de Master en informatique

Option : Ingénierie du logiciel

Thème

Mise en œuvre d'un framework destiné à la modélisation d'un workflow de type ETL.

Soutenu le 06/07/2011 devant le jury constitué de :

Mme L.Ouahrani, Maître Assistante, USDB Présidente

Mme I.Azouz, Maître Assistante, USDB Examinatrice

M. M.R. Sidoumou, Maître Assistant, USDB Examinateur

M. M.Bala, Maître Assistant, USDB Rapporteur

Promotion : 2010-2011

RESUME

Le processus ETL constitue l'usine qui produit l'information pertinente ayant la valeur et la qualité requises exigée par la suite décisionnelle. Toutes les phases qui suivent constituant la chaine décisionnelle consomment l'information livrée par ce processus d'extraction, de transformation et de chargement.

Pour comprendre et maîtriser le flux de données ainsi que la séquence des tâches de filtrage, nettoyage, conversions, intégration, agrégation et chargement, nous nous intéressons dans le cadre de ce projet de fin d'études à la modélisation de tous les éléments constituants ce processus qu'ils soient statiques (sources de données, zone de préparation, entrepôt de données) ou dynamiques (tâches d'extraction, nettoyage, filtrage, fusion, agrégation, … et chargement)

Parmi les approches les plus intéressantes dans ce domaine, nous avons étudié l'approche de modélisation de Vassiliadis et *al.* En se basant sur celle-ci, nous avons mis en œuvre un framework destiné pour la modélisation des processus ETL.

Mots clés

Système décisionnel, ETL, Datawarehouse, Workflow, qualité des données, framework, intégration.

ABSTRACT

The ETL process is the plant that produces relevant information with the value and quality requirements required by the BI suite. All phases that follow up the decision chain consume information delivered by this process of extraction, transformation and loading.

To understand and control the flow of data and the sequence of tasks of filtering, cleaning, conversion, integration, aggregation and loading, we are interested in this project graduation to the modeling of all components that processes, both static (data sources, staging area, data warehouse) or dynamic (extract tasks, cleaning, filtering, merging, aggregation, ... and loading).

Among the most interesting approaches in this field, we studied the modeling approach of VASSILIADIS and *al*. Based on this, we have implemented a framework for modeling for ETL processes.

Keywords

Decision-making system, ETL, data warehouse, workflow, data quality, framework, integration.

REMERCIEMENTS

Remerciements

Louange à Dieu le tout puissant de nous avoir accordé force, courage et volonté d'achever ce travail.

Quelques lignes ne pourront jamais exprimer la reconnaissance que nous éprouvons envers tous ceux qui, de près ou de loin, ont contribué, par leurs conseils, leurs encouragements ou leurs amitiés à l'aboutissement de ce travail.

Nos vifs remerciements accompagnés de toute notre gratitude vont tout d'abord à notre promoteur Mr BALA Mahfoud, pour nous avoir proposé ce sujet, pour les conseils qu'il n'a cessé de nous prodiguer et surtout pour la confiance qu'il nous a accordé pour la réalisation de ce projet.

Nos remerciements vont également aux membres du jury pour nous avoir honoré par l'évaluation de ce travail.

Notre reconnaissance va aussi à tous ceux qui ont contribué à notre formation depuis l'école primaire jusqu'à l'université.

....Merci à tous.

DEDICACES

Dédicaces

Je dédie ce travail

A toute personne qui sa présence dans ma vie m'est source de joie

Et particulièrement

...

Mes chers parents

Dahbia

Dédicaces

Aux témoignages d'affection, d'amour et de grande reconnaissance, aux êtres les plus chers que j'ai dans la vie, qui m'ont toujours été présent dans tous les moments, et qui m'ont soutenu avec tous ce qu'ils ont , Que dieu me les protègent «in chae Allah» .

…Mes parents,

A mon binôme Dahbia,

A tous ce que j'aime

Asma

TABLE DES MATIERES

PARTIE III : MISE EN ŒUVRE DU SYSTEME

CHAPITRE V : SPECIFICATION DES BESOINS & CONCEPTION

CHAPITRE VI : IMPLEMENTATION ET VALIDATION

CONCLUSION ET PERSPECTIVES

LISTE DES TABLEAUX

LISTE DES FIGURES

INTRODUCTION GENERALE

INTRODUCTION GENERALE

CONTEXTE GENERAL

La clé de survie dans le monde de l'entreprise est la capacité d'analyser, de planifier et de réagir aux conditions changeantes aussi vite que possible. Avec l'avènement des nouvelles technologies de communication et plus précisément Internet, les entreprises recueillent des masses de données de plus en plus importantes. Ces données étant généralement hétérogènes car elles proviennent de différentes sources. Les entrepôts de données ont apparu afin de présenter une solution pour les systèmes d'intégration et de prise de décision.

En 1992, Bill Inmon, considéré comme un des précurseurs du concept d'entrepôt de données fournit un guide pratique de construction des entrepôts dans son ouvrage de référence « Building the data warehouse [INM 92] ». Les entrepôts de données ont été introduits comme des systèmes fournissant une vue transversale et analytique sur un grand ensemble de données hétérogènes et dispersées dans le but de fournir un support d'aide à la décision.

Les introductions de Bill Inmon ont ouvert une nouvelle voie de recherche portant sur la conception des projets d'entreposage. Comme pour tout système informatique, cette conception passe généralement par les trois niveaux de conception habituels: conceptuel, logique et physique.

Dans le jargon du décisionnel, les systèmes ETL constituent une part importante d'un projet d'entreposage. Ces systèmes connus sous l'acronyme ETL (Extract-Transform-Load), sont des outils logiciels permettant d'extraire les données à partir des systèmes sources, les préparer et les transporter vers l'entrepôt.

Les tâches du processus ETL sont très complexes du fait de la variété et de la diversité des systèmes sources qui participent dans l'alimentation de l'entrepôt de données, la suite des tâches permettant la préparation des données (extraction, nettoyage, filtrage, agrégation, …etc.), la planification du processus pour le rafraichissement de l'entrepôt de données, etc.

Vu la complexité qui caractérise le processus ETL, le nombre de tâches avec toute leur complexité, leur synchronisation, leur dépendance afin de livrer à l'entrepôt, des données pertinentes ayant les qualités requises pour l'analyse, il serait nécessaire de zoomer et d'analyser davantage le fonctionnement de ce processus. En effet, Il devient nécessaire de procéder à la représentation de ce processus avec des notations formelles afin de spécifier toutes les tâches qui devront être accomplies par ce processus et d'éviter les éventuels risques et problèmes pouvant surgir dans le fonctionnement du processus.

PROBLÉMATIQUE

Bien que beaucoup d'avancées ont été réalisées dans le domaine des entrepôts de données, de nos jours il n'y a pas de méthode standard ou modèle de données pour la conception des entrepôts. D'autres parts, plusieurs rapports mentionnent qu'environ 40-50% des projets d'entrepôts de données échouent [MOR 05].

Encore moins, la modélisation des premières phases du projet d'entreposage de données qui consistent à capturer le flux de données entre les sources et l'entrepôt de données, n'a pas connu de standards. Or, Cette phase chargée d'un exercice d'intégration devient une tâche délicate et complexe, dans laquelle le concepteur doit gérer les différents conflits syntaxiques et sémantiques entre les sources. Les conflits sémantiques, qui sont les plus difficiles à gérer, peuvent être de différentes natures : conflits de représentations, conflits de noms, conflits de contexte et conflits de mesures. Cependant, des travaux sur l'ETL ont vu le jour à partir de 2002 et se sont intéressés à la modélisation du processus ETL. Nous citerons principalement :

VASSILIADIS, SIMITSIS, SKIADOUPOLOS [VAS 02], PhD thesis de SIMITSIS [SIM 04], PhD thesis de S.Lujan MORA [MOR 05].

OBJECTIF

L'intérêt dans ces travaux est le fait de donner une perception du processus ETL sur le niveau logique où le concepteur est amené à détailler les aspects techniques du processus ETL, mais notamment, sur le niveau conceptuel qui représente le mappage des données entre les sources et l'entrepôt dans un niveau plus élevé du fait que celui-ci doit avoir lieu dans les premières phases du projet d'entreposage de données.

L'objectif dans ce mémoire est de mettre en œuvre un framework permettant la modélisation d'un processus ETL basé sur le formalisme et le méta modèle de VASSILIADIS et al.

Le framework à développer devra présenter des interfaces très simples, conviviales permettant au concepteur ETL de modéliser graphiquement le processus à partir du niveau conceptuel puis de générer, grâce à d'autres informations fournies par le concepteur, le modèle correspondant au niveau logique. Pour ce faire, le framework mettra à la disposition du concepteur une boite à outils contenant les différents symboles proposés par le formalisme (source de données, attribut, Primary key, Surrogate key, transformation, note, jointure, fonctions d'agrégation, ...).

ORGANISATION DU MEMOIRE

Afin de répondre à la problématique présentée précédemment, ce mémoire a été organisé comme suit :

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

Cette partie est composée de deux chapitres qui définissent les concepts de base du domaine décisionnel.

Chapitre I : Les systèmes d'information décisionnels.

Ce chapitre présente les fondamentaux des systèmes d'information décisionnels.

Chapitre II : Etude du processus ETL

Nous présentons dans ce chapitre les différentes tâches d'un processus ETL avec tous les aspects liés à celui-ci.

Partie 2 : Etat de l'art sur la modélisation du processus ETL.

Cette partie expose un parcours de la littérature de modélisation du processus ETL. Elle se compose de deux chapitres.

Chapitre III : Etude des approches de modélisation du processus ETL.

Dans ce chapitre nous présentons une étude comparative entre différentes approches de modélisation des systèmes ETL.

Chapitre IV : Le méta modèle de VASSILIADIS et al.

Nous présentons dans ce chapitre l'approche retenue après l'étude comparative menée dans le chapitre 3. Il sera présenté le méta modèle de VASSILIADIS et al.

Partie 3 : Mise en œuvre du système.

Cette partie présente la démarche suivie pour la mise en œuvre de notre outil 'Master-ETL-Designer'.

Chapitre V : Spécification des besoins & conception

Nous présentons dans ce chapitre -de manière formelle- la spécification des besoins pour la mise en œuvre de notre outil et la conception de celui-ci.

Chapitre VI : Implémentation et validation

Ce chapitre présente les méthodes d'implémentation et les résultats obtenus au terme de ce travail.

Nous clôturons ce manuscrit par une conclusion et des perspectives pour des travaux futurs.

PARTIE I

ETAT DE L'ART SUR LES SYSTEMES DECISIONNELS ET LE PROCESSUS ETL.

Partie I

CHAPITRE I

LES SYSTEMES D'INFORMATION DECISIONNELS

1. INTRODUCTION

Dans cette ère de mondialisation que connait l'entreprise, l'efficacité est un souci majeur pour les chefs d'établissements. La grande concurrence que connaissent les marchés impose la rapidité dans l'analyse, la planification et la réaction face aux nouvelles conditions sur ces marchés pour assurer la survie de l'entreprise.

L'entreprise possède l'information. Cependant, dans la contrainte du temps qui s'impose aux décideurs, le grand volume d'information devra être réduit en des informations stratégiques pour répondre aux besoins d'analyse et de prise de décision.

Ces deux dernières décennies ont souligné une nouvelle génération dans les systèmes d'information : Les systèmes d'information décisionnels. Ces nouveaux systèmes d'informations permettent de donner une vue transversale sur un grand nombre de données issues de différentes sources pour aboutir à l'information stratégique.

Le noyau du système décisionnel est l'entrepôt de données. Les entrepôts de données gèrent des données identiques, partagées entre toutes les organisations, orientées par sujet ou par thème, qui ne sont plus cantonnées à une seule sphère fonctionnelle mais valent pour l'entreprise toute entière. Ils offrent une vue unique des données, une information plus complète, plus précise qui facilite d'autant la prise de décision.

2. LES SYSTEMES D'INFORMATION DECISIONNELS

Les systèmes d'information décisionnels se présente en un ensemble de moyens, d'outils et de méthodes permettant de collecter, consolider, modéliser et de restituer les données de l'entreprise dans le but d'apporter une aide à la prise de décision.

2.1. Architecture décisionnelle

C'est une base de données triée, classée, afin d'en sortir, grâce à des outils de restitution et d'analyse, une information stratégique.

La mise en place d'un système décisionnel consiste donc à récupérer les données de la base opérationnelle de l'entreprise afin de constituer l'entrepôt de données souhaité. Par la suite, l'accès aux données de l'entrepôt peut se faire par des outils de restitution et d'analyse.

La figure-1- explique la chaîne de déploiement d'un système décisionnel.

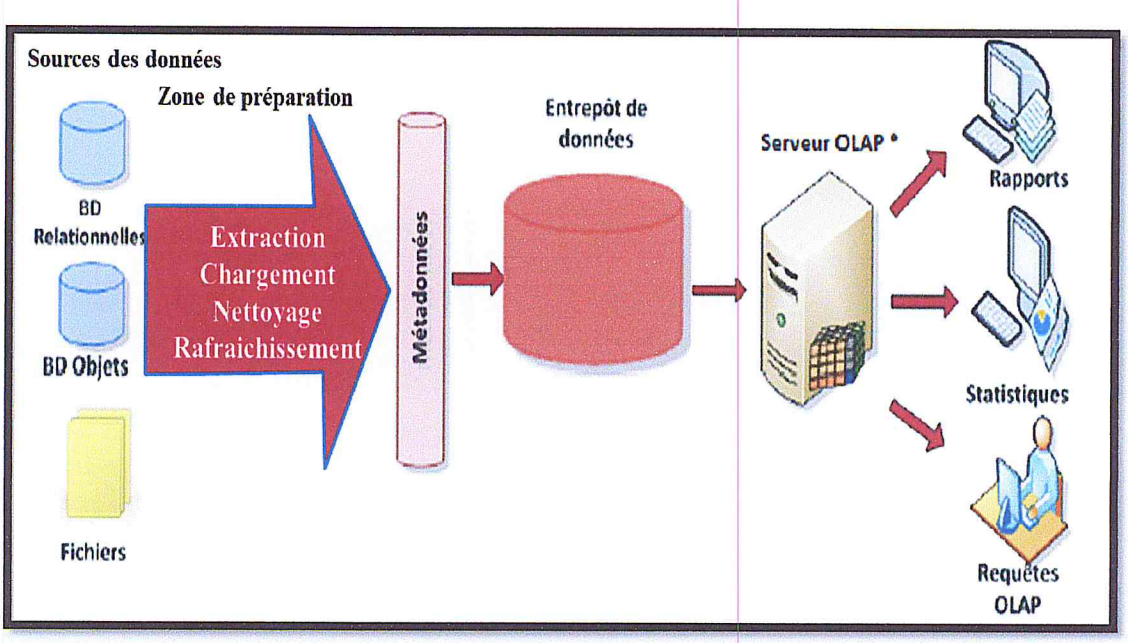


Figure 1 : L'architecture d'un système décisionnel [KHO 09].

Comme illustré dans la figure -1-, l'environnement décisionnel comprend plusieurs composants, chacun avec sa propre série de techniques, outils, produits et méthodes de conception. Néanmoins, aucun de ces composants ne constitue à lui seul un environnement décisionnel. L'architecture d'un système décisionnel peut être structurée en cinq composants essentiels :

2.1.1. Les sources de données

L'entrepôt de données stocke des données provenant de différentes sources d'informations hétérogènes et distribuées. Ces sources peuvent être des bases de données, des fichiers de données, des sources internes (bases de production) ou externes à l'entreprise…etc.

2.1.2. La zone de préparation

Avant leur chargement vers l'entrepôt de données, les données extraites des sources passent par la zone de préparation afin de subir tous les traitements nécessaires avant la population de l'entrepôt de données. Cette zone regroupe l'ensemble des processus qui se chargent de l'extraction, le nettoyage, le chargement et le rafraichissement de l'entrepôt. Ces processus sont réalisés par des outils dits ETL (Extract, Transform and Load).

Les systèmes ETL sont basés sur trois outils :

- Des **connecteurs** servant à exporter les données de différentes sources (Ex : connecteur Oracle ou SAP...).
- Des **transformateurs** qui manipulent les données (agrégations, filtres...).
- Des outils de mises en correspondance (**mappages**).

Dans le chapitre suivant, nous allons zoomer sur le processus ETL, on va étudier principalement les différentes étapes d'extraction, de transformation et de chargement.

2.1.3. Les métadonnées

Les métadonnées ou le dictionnaire des données sont au cœur du système décisionnel. En effet, les métadonnées contiennent les informations nécessaires à l'administration et à la gestion de système décisionnel.

2.1.4. L'entrepôt de données

L'entrepôt de données constitue le composant de stockage pour un grand volume de données historisées destinées à l'analyse.

2.1.5. Les outils de restitution

Les outils de restitution permettent de manipuler les données suivant des axes d'analyse. Ils permettent de présenter l'information sous différentes formes pour répondre aux besoins de toutes les catégories d'utilisateurs (souvent non informaticiens). En effet, l'information peut être sous formes de tableaux, courbes, rapports, statistiques… etc.

2.2. SI décisionnel versus SI classique

Les systèmes d'information classiques sont conçus pour supporter le traitement au jour le jour de l'entreprise. Les systèmes décisionnels de par leur nature sont dédiés à offrir une vue transversale de l'entreprise afin de fournir un support d'aide à la décision.

Afin de se rendre compte des différences entre les deux systèmes, décisionnel et classique, le tableau récapitulatif suivant englobe les caractéristiques de chacun des deux systèmes:

Caractéristiques	*SI Classique*	*SI Décisionnel*
Données	-Exhaustives -Courantes -Dynamiques -Orientées applications -Petite quantité traitée, en Gigaoctet	-Résumées -Historiques -Statiques -Orientées sujets (d'analyse) -Grande quantité traitée, en Téraoctets
Accès aux données	-Nombreux utilisateurs -Différentes catégories d'utilisateurs (employés, directeurs,…) -Mises à jour et interrogations -Requêtes prédéfinies -Réponses immédiates -Accès à peu d'information	-Utilisateurs peu nombreux -Uniquement les décideurs et analystes -Interrogations -Requêtes imprévisibles et complexes -Réponses moins rapides -Accès à de nombreuses informations

Tableau 1 : Comparaison entre les SI Classiques et les SI Décisionnels.

3. LES ENTREPOTS DE DONNEES

3.1. Définition

Le terme entrepôt de données a été exprimé pour la première fois dans les débuts des années 90 par Bill INMON, fondateur des entrepôts de données (Data warehouses). Il a défini entrepôt de données (ED) comme :

> *« Une collection de données thématiques, intégrées, non volatiles et historisées, organisées pour la prise de décision. »* [INM 92]

3.2. Caractéristiques d'un entrepôt de données

Selon la définition de Bill INMON, on peut extraire les principales caractéristiques d'un entrepôt de données.

3.2.1. Données thématiques

Les données des systèmes d'informations opérationnelles sont organisées selon les applications d'une entreprise. Ces données sont orientées applications. Les données d'un ED sont organisées par thèmes ou par sujets (Exemple : Production, Vente, Marketing…). Cette organisation permet de rassembler toutes les informations relatives à un thème précis afin de faciliter la prise de décision.

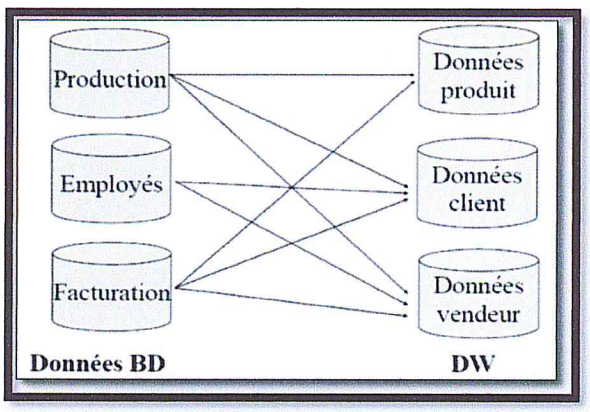


Figure 2: Données orientées sujet dans un ED. [KHO 09].

3.2.2. Données intégrées

Les données d'un ED proviennent de différentes sources hétérogènes. Cette hétérogénéité donne lieu à des conflits syntaxiques et sémantiques (conflits de représentation, de noms, de contexte et de mesure) entre les données des sources. L'intégration des données permet d'éliminer l'ensemble de ces conflits afin d'avoir une représentation uniforme et cohérente des données lors de leur chargement au niveau de l'ED. Selon [INM92], de toutes les caractéristiques d'un ED, l'intégration est l'aspect le plus important.

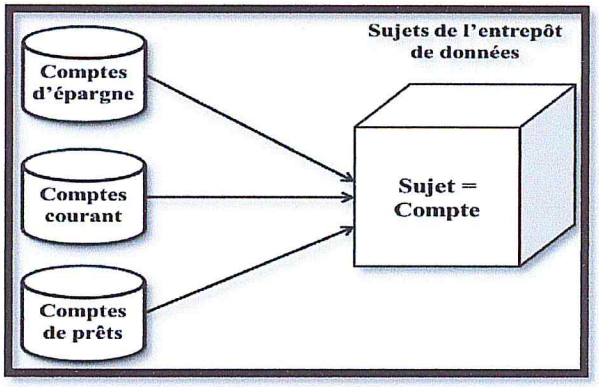


Figure 3: Données intégrées dans un entrepôt de données [PON 01].

3.2.3. Données non-volatiles

Les données d'un entrepôt sont généralement utilisées en mode consultation. Elles peuvent être interrogées mais ne sont ni modifiées, ni supprimées (sauf dans les cas de rafraichissement de l'entrepôt). Ceci permet de conserver la traçabilité des informations afin de pouvoir effectuer des analyses sur une longue période.

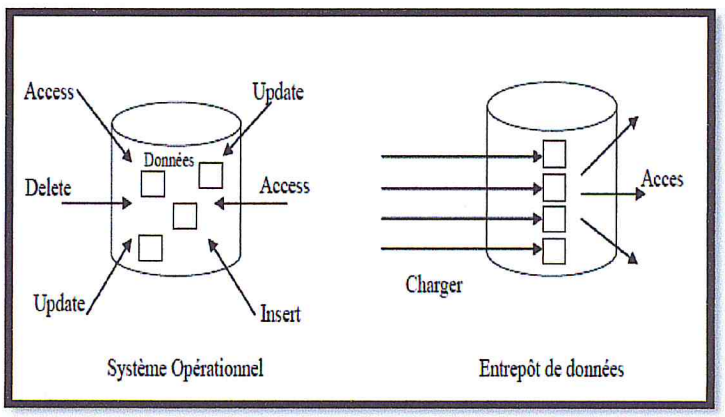


Figure 4: Données non-volatiles dans un entrepôt de données [INM 92].

3.2.4. Données historisées

Les données d'un entrepôt doivent être associées à un référentiel temps afin de refléter l'activité de l'entreprise sur une période. L'historique des données peut s'étendre sur plusieurs années. Ceci permet de suivre et d'analyser les variations des données et d'effectuer des analyses comparatives dans le temps.

3.2.5. Support d'un processus d'aide à la décision

Les données provenant des sources doivent être agrégées afin de faciliter leur analyse. Ces données peuvent être consultées à travers des outils (requêtes, outils OLAP, outils de datamining, outils de statistiques …) permettant leur manipulation et leur analyse.

3.3. Les magasins de données

Les magasins de données sont des extraits de l'entrepôt orientés sujet. Les données sont organisées de manière adéquate pour permettre des analyses rapides à des fins de prise de décision [TES 00].
Selon Bill INMON [INM 92] :

> *« Un magasin de données est issu d'un flux de données provenant de l'entrepôt de données. Contrairement à ce dernier qui présente le détail des données pour toute l'entreprise, il a vocation à présenter la donnée de manière spécialisée, agrégée et regroupée fonctionnellement. »*

3.4. Restitution d'un entrepôt de données

Les outils de restitution forment un modèle de présentation sous lequel l'utilisateur voit les données. On distingue plusieurs façons de représenter ces données.

3.4.1. Les requêtes ad hoc

Les rapports ad hoc sont préétablis par l'administrateur de l'entrepôt.

3.4.2. Les outils OLAP

OLAP (On-Line Analytical Processing) est défini comme étant le nom donné à l'analyse dynamique requise pour créer, manipuler, animer et synthétiser l'information par des modèles d'analyse de données exégétiques, contemplatifs et selon des formules.

En d'autres termes, il s'agit d'applications de modélisation descriptive et d'analyse exploratoire des données, conçues à des fins de prise de décision. Nigel PENDSE récapitule la définition d'OLAP en : « Fast Analysis of Shared Multidimensional Information (FASMI) » traduit en français comme: « Analyse Rapide d'Information Multidimensionnelle Partagée ».

Un serveur OLAP permet d'accéder à l'ED, il convertit les requêtes des clients en requêtes d'accès à l'ED et fournit des vues multidimensionnelles des données à des outils d'aide à la décision [KHO 09].

3.4.3. Datamining

Le datamining est un ensemble d'outils d'analyse d'entrepôts de données et de cubes apportant aux décideurs des éléments supplémentaires de prise de décisions qui ne sont pas forcément visibles aux premiers abords. Il met en œuvre un ensemble de techniques composites telles que de la statistique, de l'intelligence artificielle, de l'analyse des données, et des interfaces de communication homme-machine. Le résultat du datamining peut se présenter sous différents formats : texte plat, tableau, graphique… etc.

4. LA MODELISATION MULTIDIMENSIONNELLE

Les modèles de conception des systèmes transactionnels OLTP ne sont pas adaptés aux systèmes OLAP dont les requêtes sont souvent très complexes, utilisent beaucoup de jointures, demandent beaucoup de temps de calcul et sont de nature ad hoc.

Pour ce type d'environnement OLAP, une nouvelle approche de modélisation a été proposée : la modélisation multidimensionnelle. Popularisée par Ralph KIMBALL dans les années 90, cette modélisation est aujourd'hui reconnue comme la modélisation la plus appropriée aux besoins d'analyse et de prise de décision.
La modélisation multidimensionnelle consiste à considérer un sujet analysé comme un point dans un espace à plusieurs dimensions. Les données sont organisées de manière à mettre en évidence le sujet analysé et les différentes perspectives de l'analyse. [TES 00]

4.1. Modélisation conceptuelle

4.1.1. Les bases de la modélisation conceptuelle

Conceptuellement, le modèle multidimensionnel renferme les deux concepts fondamentaux de ***fait*** et de ***dimension***.

- Un ***fait*** représente le sujet ou le thème analysé. Il présente un centre d'intérêt de l'entreprise et est considéré comme un concept clé sur lequel repose le processus de prise de décision. Un fait est formé de « mesures » ou attributs du fait qui correspondent aux informations liées au thème analysé. Par exemple pour la gestion des commandes, les mesures peuvent être la quantité du produit commandé et le montant de la commande.

- Une **dimension** représente un contexte d'analyse d'un fait, par exemple la gestion des commandes peut être analysée selon les dimensions *Client*, *Magasin*, et *Temps*. Les dimensions se présentent sous forme d'une liste d'éléments organisés de façon *hiérarchique*. Une hiérarchie organise les paramètres d'une dimension selon une relation "*est_plus_fin*" conformément à leur niveau de détail. [TES 00]
Par exemple, pour la dimension *temps*, nous pouvons avoir la hiérarchie suivante : *année*, *semestre*, *trimestre*, *mois*, *semaine* et *jour*.
Le principal constructeur des modèles multidimensionnels est le *cube* de données. Un cube de données est composé de cellules qui représentent les mesures (les attributs du fait). Le cube ci-dessous permet d'analyser les mesures selon les différentes dimensions : produit, site et temps. Les hiérarchies définies sur une dimension peuvent être simples ou multiples.

4.1.2. Le schéma en étoile

Chaque dimension du cube est représentée par une table de dimension, et les mesures par une table de faits qui référence les tables de dimension en utilisant une clé étrangère pour chacune d'elles. La table de faits est normalisée, les tables de dimension sont généralement non normalisées. Cette représentation facilite l'analyse selon différentes perspectives.

4.1.3. Le schéma en flocon de neige

Le schéma en flocon de neige est une extension du schéma en étoile. Dans un schéma en étoile, les informations associées à une hiérarchie de dimension, sont représentées dans une seule table, même si les différents niveaux de la hiérarchie ont des propriétés différentes. Le schéma en flocon est le résultat de la décomposition d'une ou plusieurs dimensions en plusieurs niveaux formant une hiérarchie. Les tables de dimensions sont ainsi éclatées en plusieurs tables, ce qui peut être vu comme une normalisation des tables de dimensions.

4.1.4. Le schéma en constellation

Les schémas en constellation sont des schémas où plusieurs modèles dimensionnels se partagent les mêmes dimensions, c'est-à-dire les tables de faits ont des tables de dimensions en commun.

5. CONCLUSION

L'avènement des systèmes d'information dédiés à la prise de décision, et notamment les entrepôts de données a donné une plus grande souplesse dans l'exploration des données d'une entreprise vu qu'ils permettent une meilleure analyse guidée par l'utilisateur. Le domaine des entrepôts de données est encore très récent. Il ouvre la voie vers différents axes de recherches. Nous citons principalement :

- La modélisation des entrepôts et des magasins ainsi que leur langage d'interrogation.
- La modélisation et l'implémentation d'un processus ETL.

Partie I

CHAPITRE II

ETUDE DU PROCESSUS ETL

1. INTRODUCTION

Les besoins des utilisateurs dans un environnement décisionnel diffèrent des besoins opérationnels. Pour répondre aux mieux aux exigences des décideurs et analystes, l'information contenue dans l'entrepôt de données doit respecter certaines règles de l'entreprise, et donner une vue transversale sur l'ensemble des données de cette dernière. Le processus ETL a pour mission de fournir la matière première pour servir au mieux l'utilisateur du système décisionnel.

L'alimentation d'un entrepôt de données détient la plus grande importance dans un projet d'entreposage de données. En effet, différents rapports ont mentionné que dans la majeure des cas, le processus ETL est construit avec des procédures internes à l'entreprise et qui consomment jusqu'à 70% des ressources consacrées à un projet d'entreposage [STR 02].

Au cours de ce chapitre nous allons voir de près les mécanismes, les fonctionnalités, et les responsabilités d'un processus d'alimentation d'entrepôt de données.

2. PROBLEME D'INTEGRATION DES DONNEES

Le véritable défi dans un projet d'entreposage est la collection des données source à partir d'une variété de systèmes dispersés et hétérogènes. La plupart de ces systèmes sources ne respectent pas le même ensemble de règles métier. Très souvent, ils suivent différentes conventions de nommage et des normes de représentation des données variantes.

L'intégration des données est le regroupement de toutes les données opérationnelles pertinentes dans des structures de données cohérentes pour être prêtes au chargement dans l'entrepôt de données. Il est nécessaire de normaliser les noms et les représentations de données et de résoudre les anomalies d'une façon où une même donnée est représentée de manière unique même si celle-ci provient de différents systèmes sources. Bien que beaucoup de temps soit consommé, de nombreuses tâches d'intégration de données doivent être gérées.

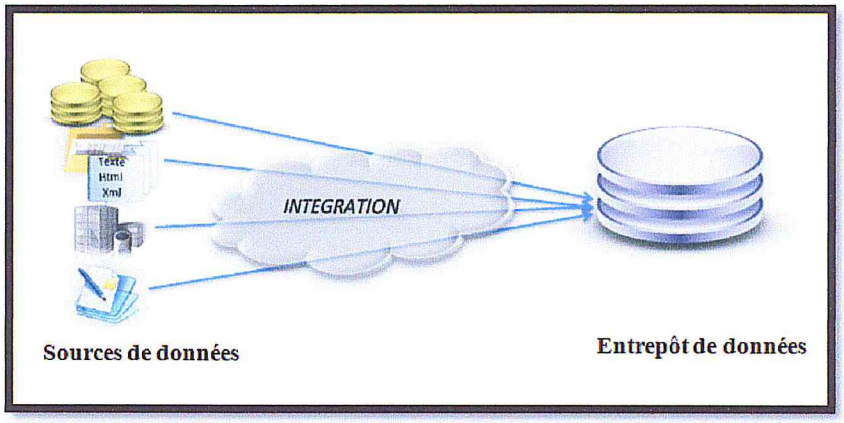


Figure 5: Intégration des données.

3. FLUX DE DONNEES ENTRE LES SOURCES ET L'ENTREPOT DE DONNEES

L'alimentation de l'entrepôt de données est très complexe. En effet, L'intégration de données émanant de plusieurs sources hétérogènes nécessite le passage de ces dernières par plusieurs tâches avant d'être chargées dans l'entrepôt.

Le processus ETL se charge de l'extraction des données de leurs emplacements sources, leurs déversement vers une zone de traitement où elles subiront des tâches de nettoyage et de conformité, et enfin leur chargement dans l'entrepôt de données.

La figure -6- illustre l'environnement du processus ETL.

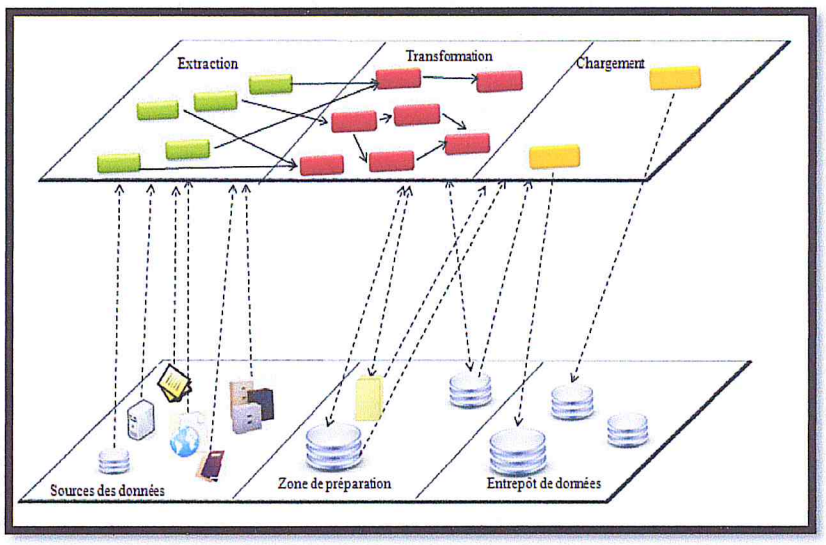


Figure 6: Environnement d'exécution du processus ETL [VAS 02].

La capture du flux de données entre les sources et l'entrepôt de données nécessite la définition de toutes les opérations à effectuer au niveau de chaque étape dans le processus ETL. Dans ce qui suit, nous présentons les principales étapes du processus ETL, ainsi que l'environnement d'exécution du processus.

3.1. Zone des traitements (Data staging area)

Les sources et l'entrepôt de données sont physiquement, logiquement et administrativement disjoints. En effet, dans la plupart des cas, l'entrepôt de données et les différentes sources qui l'alimentent sont sur des machines différentes, dépendent de différentes structures de données et sont gérés par des administrateurs différents [KIM 04]. La définition d'une zone dédiée aux traitements et à la préparation des données s'avère indispensable pour préparer les données ainsi que pour administrer le flux et gérer toutes les exceptions sans perturber le fonctionnement des systèmes sources.

En effet, la zone de traitement représente le préparatoire de toutes les tâches ETL. Elle constitue le composant physique qui héberge temporairement les données extraites à partir des systèmes sources et qui fournit la plateforme pour les différentes transformations à faire subir à ces dernières avant de devenir conformes aux schémas de l'entrepôt. La réservation d'une zone dédiée pour les tâches du processus ETL permet entre-autres de garantir la récupération en cas de problèmes (problème matériel, troubles d'alimentation…) des données à différents niveaux de traitement et épargner des problèmes de performances et blocages pour les systèmes sources.

3.2. Extraction des données -Extract-

L'extraction des données dans un système décisionnel s'avère différente d'une simple extraction dans un système de migration de données par exemple (migration d'un système de fichiers plats vers un système de base de donnes relationnelles). Les principales différences entre la fonction d'extraction dans les deux systèmes se résument comme suit :

- Pour un entrepôt de données, la fonction d'extraction considère plusieurs sources distribuées.

- Les changements dans les sources de données provoquent un rafraichissement continu et incrémental pour un processus décisionnel.
- Dans un système de migration, la fonction d'extraction s'effectue une seule fois, à partir d'une seule source et ne nécessite que quelques opérations de conversion de types.

3.2. 1. Les différents types de systèmes sources

Les sources de données se présentent sous différentes formes. La compréhension de la structure de chaque source est nécessaire afin de pouvoir effectuer l'extraction des données à partir de ces dernières. Ainsi, selon la structure de la source de données, des programmes spéciaux (les extracteurs[1], les wrappers[2]) sont définis afin de lire et d'extraire l'information à partir de cette source. Les sources de données peuvent être :

- **Structurées :** Les données dans les sources structurées respectent un schéma prédéfini. Les bases de données relationnelles et les bases de données objet adhérent à ce type de systèmes sources. L'extraction à partir des sources structurées est relativement simple grâce à l'utilisation des SGBDs.
- **Semi-structurées :** Dans ce type de systèmes sources, la structure n'est pas strictement prédéfinie. Cependant, les éléments de données portent leurs propres définitions sémantiques sous formes de labels. Les fichiers XML, HTML sont des exemples de sources semi-structurées.
- **Non-structurées :** La représentation de l'information dans ce type de sources n'adhère à aucun format. Les fichiers texte, les images, les fichiers internet et les documents non numériques sont des exemples de sources de données non-structurées.

3.2. 2. Complexité d'extraction de données

L'extraction des données est une clé pour la réussite d'un projet d'entreposage. Il est nécessaire de formuler une stratégie pour la fonction d'extraction de données.

[1] Les extracteurs sont des programmes développés en procédures internes à l'entreprise.
[2] Les wrappers sont des outils qui existent sur le marché exécutant la fonction d'extraction à partir de différents types de structures de données.

La complexité d'extraction des données dans un projet d'entreposage impose une attention particulière aux problèmes suivants :

- **Identification de la source** : identifier les applications et les structures des différentes sources de données et définir la source de données adéquate pour chaque élément de données dans l'entrepôt.
- **Méthodes d'extraction** : pour chaque source de données, définir si la fonction d'extraction se fait par un outil dédié ou par des procédures internes à l'entreprise.
- **Fréquence d'extraction** : pour chaque source de données, définir la fréquence d'extraction. L'extraction des données peut se faire quotidiennement, une fois par semaine, tous les trois mois… etc.
- **Temps d'exécution** : pour chaque source de données, on note le temps d'exécution de la fonction d'extraction.
- **Séquence des tâches d'extraction** : déterminer la séquence des tâches d'extraction et leur synchronisation.
- **La gestion des exceptions** : permet de déterminer la façon de traiter les données qui ne peuvent pas être extraits à partir des sources.

3.2.4. Méthodes d'extraction des données

Avant de présenter les différentes techniques d'extraction de données, il est important de bien comprendre la nature de la source de données à extraire.

Les données dans les systèmes sources évoluent dans le temps. Cependant, le traitement des MAJ d'une donnée dans un système opérationnel implique des considérations particulières dans l'entrepôt de données. En effet, lorsque la donnée en question connait des MAJ et est considérée pertinente pour l'analyse (comme dimension ou mesure), le système décisionnel préserve son historique en inscrivant la nouvelle valeur accompagnée de la date de MAJ.

La capture de l'historique à partir des sources de données dépend de la manière dont les données sont stockées dans les systèmes sources.

i. Le stockage des données dans les systèmes opérationnels

Les données opérationnelles dans le système source peuvent être classifiées en deux grandes catégories. Le type de la technique d'extraction de données utilisé dépend de la méthode de stockage.

- **Stockage de la valeur courante (transitoire)**

La majorité des attributs dans les systèmes sources appartiennent à cette catégorie. Dans ce cas, la valeur stockée pour un attribut représente la valeur de l'attribut en cours. Dans cette catégorie, la valeur d'un attribut reste constante jusqu'à ce qu'une transaction métier la change.

- **Stockage d'état périodique**

Dans cette catégorie, la valeur précédente de l'attribut est préservée chaque fois qu'un changement se produit. La valeur de l'attribut est stockée avec référence au temps où la nouvelle valeur devient effective. Ainsi, cette catégorie inclut des événements stockés avec référence au moment où chaque événement se produit.

La figure -7- montre la différence entre les deux catégories de stockage de données.

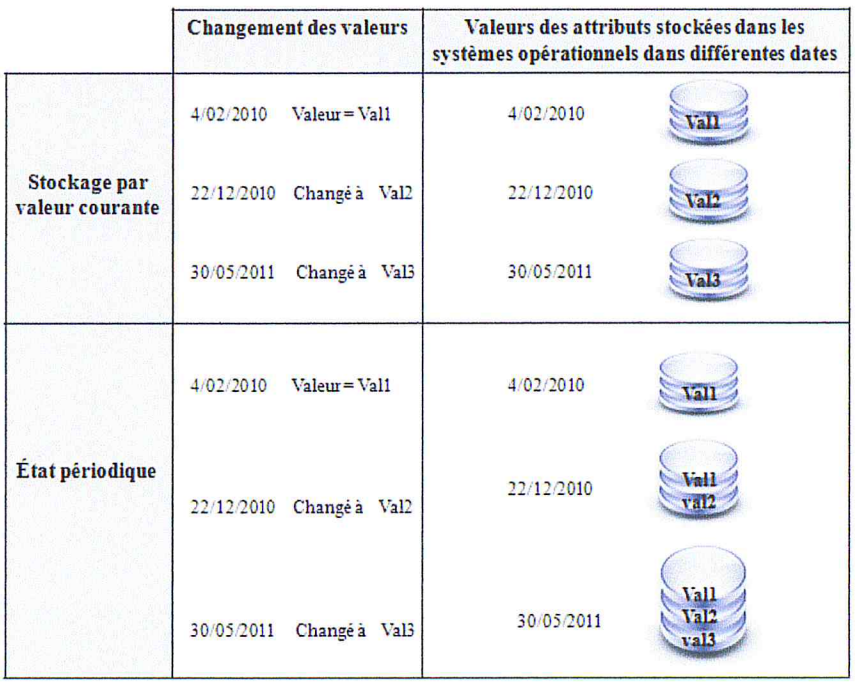


Figure 7 : Catégories de Stockage de données.

i. Les techniques d'extraction des données

Il existe deux grands types d'extraction des données à partir des systèmes sources : extraction des données statiques et extraction par capture de données incrémentale.

- **Extraction des données statiques**

Cette technique consiste en la capture d'une image sur les sources de données dans un moment donné. Dans les systèmes de stockage de données courantes, cette capture inclurait toutes les données courantes identifiées dans l'extraction. Dans les systèmes de stockage de données périodiques, cette capture de données doit inclure les valeurs et événements disponibles dans les systèmes opérationnels sources.

La capture des données statiques est utilisée principalement pendant le chargement initial de l'entrepôt de données et parfois dans le rafraichissement complet de l'entrepôt.

- **Extraction par capture de données incrémentale**

Consiste en la révision du contenu de l'entrepôt depuis la dernière opération d'extraction. Si le stockage dans les systèmes sources est de type 'stockage par valeur courante', la capture des modifications n'est pas facile. Pour les données de valeurs périodiques, cette technique consiste à extraire les valeurs et événements qui ont été enregistrés depuis la dernière date d'extraction.

La capture des données incrémentale peut être immédiate ou différée. Dans la capture des données immédiate il y a trois options distinctes. Deux options sont disponibles pour la capture de données différée.

a. L'extraction de données immédiate

L'extraction de données est en temps réel. Elle se produit lors de la survenue de transactions dans les bases de données et les fichiers sources. L'extraction des données immédiate se base sur les journaux de transactions ou les déclencheurs de bases de données pour détecter les changements dans les systèmes sources.

b. L'extraction de données différée

Les techniques d'extraction de données différées ne capturent pas les modifications en temps réel. La capture se produit ultérieurement. Cette méthode utilise des techniques de capture basée sur la date et l'horodatage (timestamping)[3], et des techniques de capture différentielle (comparaison de fichiers) pour détecter les modifications apportées aux données dans les systèmes sources.

3.3. Transformation des données -Transform-

Les données sources sont généralement brutes et ne disposent pas de la valeur et qualités requises par l'entrepôt de données. Disposer d'une information de valeur pour la prise de décision est le principe indispensable de l'entrepôt de données. L'extraction des données opérationnelles à partir de nombreux systèmes sources influe sur la qualité des données au niveau de l'entrepôt. La qualité des données doit être enrichie et améliorée avant le chargement des données dans l'entrepôt.

Avant de déplacer les données extraites des systèmes sources vers l'entrepôt de données, il est indispensable d'effectuer divers types de transformations de données afin de les rendre conformes aux normes de l'entrepôt.

3.3.1. Les tâches de base

Indépendamment de la variété et de l'hétérogénéité des systèmes opérationnels sources et quelle que soit la complexité de l'entrepôt de données, un certain nombre de tâches de base reviennent très souvent et constituent la plupart des fonctions de transformation. Nous résumons, dans ce qui suit, les principales tâches de base :

a. Sélection (Restriction en Algèbre relationnelle)

Ceci a lieu au début du processus de transformation de données. La sélection concerne des enregistrements entiers ou des parties de plusieurs enregistrements à partir des systèmes sources. La tâche de sélection représente habituellement une partie de la fonction d'extraction elle-même. Cependant, dans certains cas, la

[3] L'horodatage (en anglais timestamping) est un mécanisme qui consiste à associer une date et une heure à un événement, une information ou une donnée informatique. Il a généralement pour but d'enregistrer l'instant auquel une opération a été effectuée.

structure des sources de données ne permet pas l'extraction des parties nécessaires. La sélection permet d'extraire les enregistrements nécessaires dans la suite du processus ETL.

b. Eclatement/jointure

Cette tâche permet la manipulation des enregistrements sélectionnes. Parfois, les parties sélectionnées doivent être séparées/fusionnées lors des transformations des données. La jointure des enregistrements sélectionnés à partir de nombreux systèmes sources est très utilisée dans un processus ETL.

c. Conversion

Cette tâche comprend une grande variété de fonctions de conversions élémentaires. Ces conversions ont pour but de normaliser les données provenant des systèmes sources hétérogènes pour les mettre dans un format adéquat et approprié.

d. Agrégation

L'entrepôt de données se caractérise et se distingue principalement des sources de données par son contenu agrégé. Le détail d'informations sur un métier particulier de l'entreprise est la vocation des systèmes sources. Dans un processus ETL, Les fonctions de transformations sont responsables, entre autres, de laisser le détail et procéder à des calculs pour ramener vers l'entrepôt un niveau d'informations et d'agrégation plus élevé. Ceci est justifié par le fait qu'aucun utilisateur n'ait besoin des données d'un niveau de détail (granularité basse) pour l'analyse.

3.3.2. Principaux types de transformations

Les fonctions de transformations à faire subir, après extraction, à un ensemble de données sources sont définies généralement à base des tâches élémentaires présentées ci-dessus. Voici un résumé des transformations les plus courantes :

a. Révisions de format

Ces révisions concernent les changements des types de données et de la longueur des champs. Il est nécessaire de normaliser et de changer les types de données pour fournir des valeurs dans un format standard aux utilisateurs.

b. Décodage des champs

Dans différents systèmes sources, un même objet peut être décrit de manière différente. L'exemple classique est celui de la donnée sexe. Dans un système S1, on utilise « 1 » et « 2 » pour désigner respectivement « Masculin », « Féminin » alors que dans S2 on utilise « M » et « F ».

c. Valeurs calculées et dérivées

Ce type de transformation s'occupe de calculer ou de dériver à partir des données extraites du système source de nouvelles valeurs, avant d'être chargées dans l'entrepôt de données.

d. Division des champs uniques

Il est important de diviser les champs structurés en plusieurs composants plus élémentaires. Tout d'abord, ceci permet d'améliorer la performance opérationnelle par l'indexation sur les composants élémentaires. Deuxièmement, les utilisateurs peuvent avoir besoin d'effectuer une analyse en utilisant des composants.

e. Conversion des unités de mesures

De nombreuses entreprises ont aujourd'hui plusieurs succursales dans le monde. Si l'entreprise exerce ses activités à l'étranger, on doit convertir les mesures pour que les chiffres puissent tous être dans une unité de mesure standard.

f. Génération de clé de substitution (Surrogate key)

Les clés dans les tables de faits et de dimensions sont générées en fonction des clés dans les systèmes sources. Cependant, les clés dans ces systèmes sont structurées pour avoir un sens particulier, or, l'utilisation de ces clés comme clé dans les tables de l'entrepôt de données génère des problèmes. Si une donnée est déplacée vers un entrepôt, une partie de la clé devra être changée. Transformer ces clés en clés générées par le système lui-même, est ce qui est appelé génération des clés de substitution.

Exemples de transformations

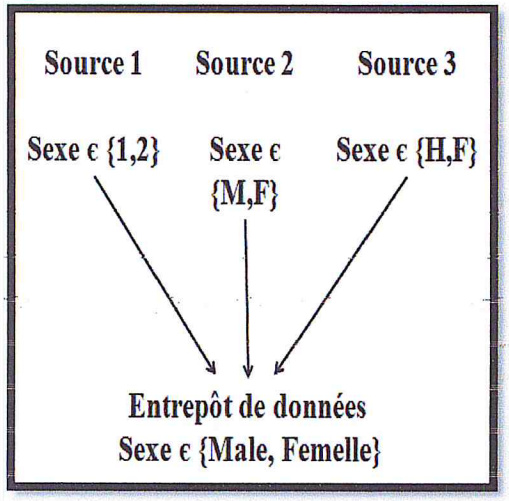


Figure 8: Décodage des champs.

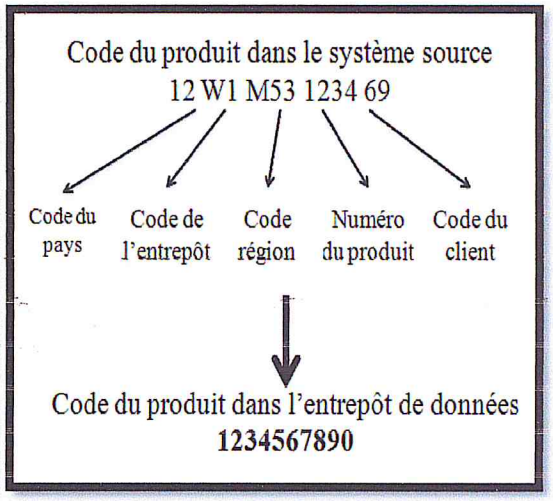


Figure 9: Génération de clé de substitution.

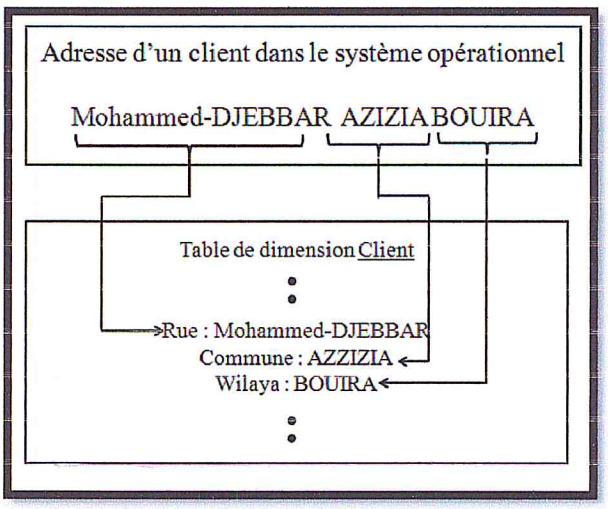


Figure 10: Division des champs uniques.

3.4. Chargement des données -Load-

Il est évident que les fonctions de transformations s'achèvent dès que les images de chargement soient créées. Les principales fonctions qui viennent par la suite sont celles qui prennent les données préparées, les appliquent à l'entrepôt de données, et les stockent dans la base de données de ce dernier, on parle des fonctions de chargement.

inconditionnellement les données entrantes, en préservant les données existantes dans la table cible.

c. Destruction de fusion

Dans ce mode, on applique les données entrantes vers les données cibles. La correspondance d'une clé primaire de l'enregistrement entrant avec une clé d'un enregistrement existant implique la mise à jour de l'enregistrement cible correspondant. Si l'enregistrement entrant est un nouvel enregistrement sans correspondance, celui-ci sera ajouté à la table cible.

d. Construction de fusion

Ce mode est légèrement différent par rapport au mode destruction de fusion. Si la clé primaire d'un enregistrement entrant correspond à une clé d'un enregistrement existant, laisser l'enregistrement existant, ajouter l'enregistrement entrant, et marquer l'enregistrement ajouté comme remplaçant de l'ancien enregistrement.

La figure ci-dessous montre les différents modes d'application des données.

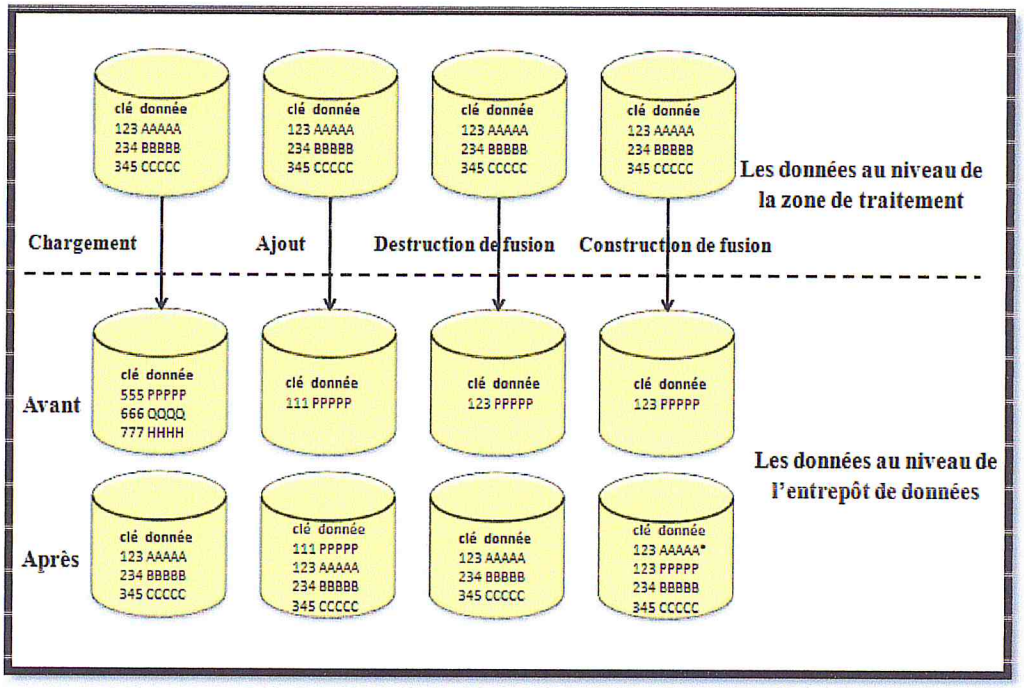


Figure 11: Modes d'application des données [PON 01].

Le chargement de données vers l'entrepôt pose plusieurs problématiques, telles que :

- Le chargement de l'entrepôt de données peut prendre beaucoup de temps. Il détient généralement une grande partie des préoccupations.
- Pendant le chargement, l'entrepôt de données doit être hors ligne.
- Il est difficile d'estimer le temps d'exécution des chargements, en particulier le chargement initial ou le rafraichissement complet de l'entrepôt de données.
- Il faut trouver les intervalles de temps adéquats qui permettent d'exécuter le chargement sans affecter l'utilisateur final de l'entrepôt.

Afin de régler les problèmes cités ci-dessus, le processus de chargement est divisé en plusieurs petits segments. Chaque segment est responsable du chargement d'une partie de l'entrepôt de données. De cette solution, découle des avantages considérables :

- La possibilité d'exécuter les segments de chargement en parallèle.
- Certaines parties de l'entrepôt deviennent fonctionnelles au cours du chargement des autres parties.
- La possibilité de faire des tests d'estimation du temps d'exécution des chargements.

3.4.1. Modes d'application des données [PON 01]

Les données peuvent être appliquées à l'entrepôt dans quatre modes différents : chargement, ajout, destruction de fusion et construction de fusion.

a. Chargement

Si la table cible à charger existe déjà et il existe des données dans la table, le processus de chargement efface les données existantes et applique les données du fichier entrant. Si la table est vide avant le chargement, le processus de chargement applique simplement les données du fichier entrant.

b. Ajout

L'ajout peut être vu comme une extension du mode d'application 'chargement'. Si les données existent déjà dans la table, le processus d'ajout additionne

3.4.2. Types de chargement des données

Trois types de chargement de données dans l'entrepôt de données existent : le chargement initial, le chargement progressif, et le rafraîchissement complet [PON 01].

a. Le chargement initial

Le chargement initial permet de peupler toutes les tables d'entrepôt de données pour la première fois.

L'entrepôt de données peut être chargé complètement dans une seule exécution. Le chargement peut aussi être divisé en plusieurs segments exécutant chacun un sous-chargement. Chaque exécution d'un segment génère le chargement complet d'une table de l'entrepôt de données. Dans ces cas, le mode d'application du chargement utilisé est le mode 'chargement'.

b. Le chargement progressif

Le chargement progressif applique les modifications en cours dans les systèmes opérationnels dans l'entrepôt de données de manière périodique. Autrement dit, dans un processus de chargement progressif ou incrémental, les enregistrements apportés à partir des systèmes sources existent déjà dans l'entrepôt de données avec des différences dans les valeurs de certains champs dues à la modification de ces valeurs dans les systèmes opérationnels. Les modes d'applications 'construction de fusion' et 'destruction de fusion' correspondent aux cas où les enregistrements d'entrées existent déjà dans l'entrepôt avec des valeurs différentes.

c. Le rafraichissement complet

Ce mode d'application efface complètement le contenu d'une ou de plusieurs tables et les recharge avec de nouvelles données. Ce type de chargement implique de reconstruire périodiquement l'entrepôt de données tout entier ou de reconstruire seulement certaines tables dans l'entrepôt.

Comme dans le cas du chargement initial, les modes 'chargement' et 'ajout' sont applicables dans le rafraichissement complet.

3.4.3. Procédures de chargement des données

a. Chargement des tables de dimensions

La procédure de chargement des données pour les tables de dimension détient deux considérations :

- La première concerne la génération de clés de substitution (surrogate key). En effet, les clés primaires dans les systèmes sources ne sont pas toujours susceptibles d'identifier de façon unique un enregistrement dans l'entrepôt de données.
- La deuxième est liée à la stratégie d'application du chargement. Chaque attribut dans l'entrepôt de données est associé avec une valeur SCD (Slowly Changing Dimension). Le champ SCD prend trois valeurs 1, 2 et 3 interprétées respectivement : remplacement de l'ancienne valeur, création d'un nouvel enregistrement, marquage des valeurs modifiées comme 'ancienne' et ajout de la nouvelle valeur [PON 01].

b. Chargement des tables de faits

La clé d'une table de faits est la concaténation des clés des tables de dimensions. Le chargement des tables de faits doit prendre en compte deux conditions :

- Le chargement des tables de dimensions concernées doit précéder le chargement de la table de fait.
- Le chargement de chaque enregistrement de la table de faits, doit être précédé par la création d'une clé concaténée pour cet enregistrement à partir de clés des enregistrements de dimensions correspondants.

4. LA QUALITE DES DONNEES

La qualité des données est le facteur faisant la différence entre les outils ETL. Il est prouvé que des milliards se perdent chaque année à cause de données de mauvaise qualité, erronées, …etc. Derrière la mauvaise affectation des ressources, ce sont des clients insatisfaits, une érosion de la crédibilité et l'incapacité à prendre la bonne

décision au bon moment. L'objectif du décisionnel est d'apporter aux utilisateurs des données de qualité. Il est donc nécessaire d'évaluer, d'analyser et de corriger si nécessaire, l'information, dès le départ dans le processus ETL.

4.1. Indicateurs de la qualité des données

On entend par *«qualité»* l'ensemble des caractéristiques des données qui conduisent à leurs capacité à satisfaire des besoins définis et implicites. La liste suivante définit quelques indicateurs pour la qualité de données [PON 01].

- **Intégrité du domaine** : la valeur d'un attribut se situe dans le domaine défini et permis de valeurs pour cet attribut

  Exemple : domaine de définition pour le sexe = {male, femelle}
- **Type de données** : la valeur d'un attribut est enregistrée selon le type de données défini pour l'attribut.

  Exemple : Le type de données *numérique* pour un attribut implique que toutes les valeurs de cet attribut doivent être des valeurs numériques et ne contiennent pas du texte.
- **Cohérence** : la forme et le contenu d'un champ de données est le même à travers les différents systèmes sources.

  Exemple : Si le code du produit pour le produit 'ABC' dans un système est '1234', alors le code de ce produit doit être '1234' dans tous les systèmes sources.
- **Complétude** : pas de valeurs manquantes pour un attribut donné dans le système.
- **La conformité aux règles métiers** : La valeur de chaque élément adhère aux règles métiers prédéfinies.

  Exemple : Dans un système de prêt bancaire, le solde du prêt doit toujours être positif ou zéro.
- **Structures définies** : Partout où un élément de données peut être structuré en composants individuels, l'élément doit respecter cette structure comme définie.

  Exemple : L'adresse d'un fournisseur se compose de numéro d'immeuble, rue et ville. L'adresse pour un fournisseur donné doit contenir le numéro d'immeuble, la rue et la ville, dans l'ordre définie.

- **Clarté** : Les conventions de nommage aident les utilisateurs à mieux comprendre le sens de l'élément.
- **Temps** : Les utilisateurs déterminent le temps de validité d'un élément.

  Exemple : Si les utilisateurs définissent des données de dimension de clients qui ne doivent pas dépasser un jour, alors le changement dans les données des clients dans les systèmes sources doit être appliqué sur l'entrepôt de données tous les jours.
- **Utilité** : Un élément qui ne rapporte aucune valeur pour l'utilisateur, alors il est inutile de le charger dans l'entrepôt de données.
- **Adhérence aux règles d'intégrité des données** : Les enregistrements dans les bases de données relationnelles doivent adhérer aux règles d'intégrité référentielles, et d'intégrité d'entité.

  Exemple :
  - Une table qui contient des lignes avec comme clé primaire *« null »* ne respecte pas les règles d'intégrité d'éléments.
  - L'intégrité référentielle force l'étlissement des relations père-fils correctement.

4.2. Types de problèmes de la qualité de données

- **Valeurs factices dans les attributs** : Ce problème est généralement dû au fait de remplir des champs avec de fausses valeurs, en attendant que la valeur du champ soit disponible.
- **Absence de valeurs pour les données** : C'est très courant dans les données des utilisateurs, où de données importantes pour l'analyse de l'entrepôt de données ne figurent pas dans les systèmes opérationnels. Généralement, les SGBD affectent par défaut la valeur NULL pour des colonnes non renseignées
- **Utilisation de champs non-officielle** : Utilisation d'un champ pour servir à stocker d'autres informations que celle prévue (ajouter des commentaires sur le client dans le champ *contact du client* à cause de la non réservation d'un champ *commentaires*)

- **Valeurs contradictoires** : Il y a des champs reliés dans les systèmes sources pour lesquelles les valeurs doivent être compatibles. (Le code postal pour une ville *V* (42415) et le nom de cette ville *V* (Bou Ismail) doivent être compatibles.
- **Violation des règles du métier** : Dans un système de paie et du personnel, une règle dicte que le nombre de jours travaillés dans un an, plus le nombre de jours de vacances, le nombre de jours fériés et le nombre de jours de maladie ne peut dépasser 365 ou 366. Tout dossier d'employé avec le nombre de jours de plus que 365 ou 366 viole cette règle de base.
- **Identificateurs non-unique** : Ce problème se pose lorsque pour une même entité de données qui apparait dans des systèmes sources différents, ces entités apparaissent avec des identificateurs différents dans chaque système.
- **Valeurs incompatibles** : Les codes pour le type de politiques dans les systèmes existants dans une compagnie d'assurance pourraient avoir des valeurs incompatibles. Par exemple, A = Auto, C = Accueil, I = Inondation, E = Employé dans un système, et 1, 2, 3 et 4, respectivement dans un autre.

Comme vu dans cette section, les systèmes sources peuvent contenir des données de mauvaise qualité. Ceci ajoute de la complexité au processus ETL. Il est nécessaire – dans les premières phases du projet d'entreposage – de faire une étude détaillée sur le contenu des systèmes sources, et de prévoir tous les problèmes qui peuvent nuire à la qualité des données à charger vers l'entrepôt.

5. LES METADONNEES

Différents processus lors de la construction et l'administration de l'entrepôt de données génèrent des parties de métadonnées de l'entrepôt. Les parties de métadonnées générées par un processus peuvent être utilisés par un autre [PON 01]. Dans un entrepôt de données, les métadonnées jouent un rôle clé, et permettent la communication entre les différents systèmes. Un système ETL communique et génère des métadonnées tout au long de son cycle de vie. On peut regrouper ces

dernières en deux catégories : métadonnées sur l'environnement, et métadonnées générées à partir du processus ETL.

5.1. Métadonnées de l'environnement

Les métadonnées de l'environnement servent à décrire et documenter les structures de données (les sources, l'entrepôt, et la zone de traitement) ainsi que les règles du métier.

Ces métadonnées sont catégorisées dans [KIM 04] comme suit :

5.1.1. Les métadonnées métiers

Les métadonnées métier sont comme une carte de mappage, ou un répertoire facile d'utilisation de l'information, indiquant le contenu et la façon d'y arriver [PON 01]. Chaque attribut dans l'entrepôt de données doit avoir une définition métier qui lui est associée. Si le métier ne peut définir l'attribut, ceci indique que l'attribut n'a aucune valeur analytique et n'y a probablement pas besoin de le charger dans l'entrepôt. Les définitions métiers consistent en une ou deux phrases qui décrivent le sens métier d'un attribut. Les définitions métiers contiennent aussi les interprétations des noms des tables et des noms de colonnes.

Par exemple : l'attribut CD-STT-EMP désigne Code de statut de l'employé.

5.1.2. Métadonnées techniques

Les métadonnées techniques ressemblent à un guide d'assistance pour l'équipe technique pour construire, entretenir et administrer l'entrepôt de données [PON 01].

Ces métadonnées décrivent les aspects techniques d'un élément, incluant la structure, le format, et l'emplacement.

Les métadonnées techniques contiennent les règles métier[4] qui peuvent comprendre toute règle à partir des valeurs permises aux valeurs par défaut jusqu'aux calculs de champs dérivés [KIM 04].

[4] Selon [KIM 04], les règles du métier peuvent être catégorisées dans les métadonnées métiers ou les métadonnées techniques. Cependant, il est préférable de les classer dans la deuxième catégorie du fait qu'elles constituent l'essence du processus ETL [KIM 04].

5.2. Métadonnées générées à partir du processus ETL

Cette catégorie traite les métadonnées générées par l'équipe ETL et utilisées soit par l'équipe pour gérer le processus ETL, par les utilisateurs finaux ou d'autres membres de l'entrepôt de données pour mieux comprendre les données dans celui-ci. Ces métadonnées sont classifiées en cinq classes [KIM 04].

5.2.1. Traçabilité des données

Appelé aussi le mappage logique des données, celui-ci illustre la traçabilité des données à partir des sources jusqu'à l'entrepôt de données. Chaque tâche ETL – de l'extraction au chargement avec toutes les transformations entre l'extraction et le chargement – est capturée par ce type de métadonnées.

5.2.2. Métadonnées sur les transformations

Ce type de métadonnées contient des informations sur la construction du processus ETL. Il contient la définition des fonctions, procédures et requêtes impliquées dans les transformations qui s'exécutent au cours du processus.

5.2.3. Métadonnées sur l'exécution des lots[5]

Ces métadonnées comprennent des informations sur l'ordonnancement des tâches du processus ETL. L'ordre d'exécution des tâches, ainsi que les dépendances entre ces dernières doivent être bien définies.

5.2.4. Métadonnées sur les évènements liés à la qualité

Les métadonnées doivent contenir des informations sur la gestion des exceptions dans le processus ETL. Les points d'injection des contrôles de qualité et le traitement des exceptions doivent être définis.

5.2.5. Métadonnées sur l'exécution du processus

Contient des statistiques sur le processus :

- Les statistiques sur le chargement des tables de de l'entrepôt.
- Statistiques sur le chargement des tables de la zone de préparation.

[5] Un lot est un ensemble de tâches regroupées dans un ordre d'exécution donné pour effectuer le nécessaire de préparation des données et leur chargement dans l'entrepôt.

- Statistiques sur les succès et échecs pour chaque tâche.
- Fréquence de chargement des tables.
- …etc.

6. LA GESTION DU PROCESSUS ETL

6.1. ETL vu comme workflow

Comme vu dans les sections précédentes, le processus ETL est très complexe par ses tâches d'extraction, de transformation et de chargement. La gestion d'un tel processus nécessite de considérer plusieurs paramètres de gestion et d'administration. Des approches de modélisation du processus ont traité le processus ETL comme workflow [SIM 04].

Un workflow dans son contexte général est défini comme la modélisation et la gestion informatique de l'ensemble des tâches à accomplir et des différents acteurs impliqués dans la réalisation d'un processus métier.

Un workflow de type ETL est une suite de tâches synchronisées pour s'exécuter dans un ordre donné dans le but d'accomplir les fonctions d'extraction, de transformation et de chargement tout en respectant, au niveau de chaque tâche, des règles de validation correspondant aux indicateurs de qualité énoncés précédemment.

6.2. Plans de gestion du workflow

La considération du processus ETL comme un workflow implique la définition de trois plans de gestion distincts pour ce dernier [SIM 04].

6.2.1. Plan d'exécution (Execution plan)

La définition de ce plan inclut la définition de trois perspectives

- **Séquence d'exécution (Execution sequence)**

Spécifie l'ordonnancement des activités. Quelle activité s'exécute en premier, en deuxième, et ainsi de suite. Indique les activités qui s'exécutent en parallèle. Cette perspective définit aussi la synchronisation des activités à un point de rendez-vous.

- **Ordonnancement de l'exécution (Execution schedule)**

Cette perspective définit les évènements ou bien les dates qui déclenchent le traitement des lots du processus.

- **Plan de récupération (Recovery plan)**

Spécifie la séquence d'étapes à entreprendre en cas d'échec d'une activité (re-lancement de l'activité, ou annulation des résultats intermédiaires).

6.2.2. Plan d'administration (Administration plan)

- **Surveillance ou journalisation (Monitoring or logging)**

Cette fonctionnalité implique la notification de l'administrateur, soit en ligne (surveillance) ou hors ligne (journalisation) pour le statut d'une activité exécutée.

- **Sécurité et gestion de l'authentification (Security and authentication management).**

6.2.3. Relation avec les données (Relationship with data)

- **Flux de données primaire (Primary data flow)**

Décrit la traçabilité des données à partir des sources vers la destination finale dans l'entrepôt de données telles qu'elles passent à travers le workflow.

- **Flux de données secondaire (Data flow for logical exceptions)**

Définit le flux pour les données problématiques, c.à.d. les lignes qui présentent des problèmes de qualité.

La sémantique du workflow ETL est générée par la combinaison des flux de données et de la séquence d'exécution. La figure suivante illustre les différents plans d'un workflow ETL.

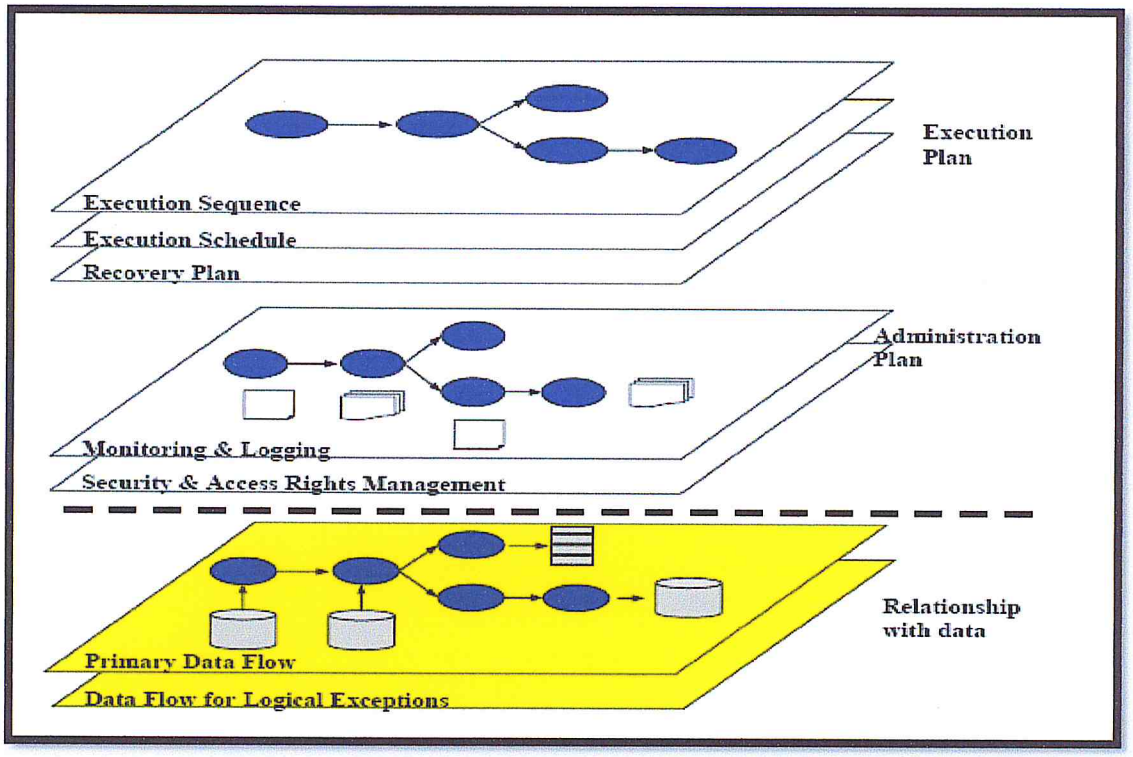


Figure 12: Les différentes Perspectives d'un workflow ETL.

7. CONCLUSION

Au cours de ce chapitre, nous avons souligné les grandes fonctionnalités du processus ETL. Aussi, Il a été mis l'accent sur l'environnement d'exécution et les paramètres relatifs au bon fonctionnement et au succès d'un projet d'entreposage. En effet, il devient évident après cette étude, que le processus ETL dans sa totalité nécessite d'être étudié et documenté avant toute tentative de réalisation d'un projet d'entreposage. Dans la suite de ce mémoire, nous allons présenter les différentes approches de modélisation de ce processus, suivi d'une étude comparative qui nous guidera vers le choix judicieux d'une approche à retenir pour la construction du Framework de modélisation.

PARTIE II

ETAT DE L'ART SUR LA MODELISATION DU PROCESSUS ETL.

Partie II

CHAPITRE III

ETUDE DES APPROCHES DE MODELISATION DU PROCESSUS ETL

1. INTRODUCTION

Un projet décisionnel se réalise principalement en deux grandes phases à savoir (1) phase amont qui s'occupe de préparer la matière première (données) pour la construction de l'ED (2) phase aval qui présente/restitue les données aux utilisateurs finaux. Cependant, le plus grand intérêt dans le domaine de la modélisation des EDs au niveau conceptuel s'est penché beaucoup plus sur la partie aval (exploitation de l'ED).

En effet, des travaux de modélisation ont été élaborés dans le cadre de la modélisation dimensionnelle d'un ED, on cite principalement le modèle en étoile et ses variantes proposés par [KIM 04] basé sur les notions de dimension, faits et mesures. Une autre approche utilisant une extension du standard UML a été proposée par [MOR 05] pour une modélisation dimensionnelle orientée objet.

Le processus ETL détient une très grande importance dans l'alimentation et le rafraîchissement de l'ED, notamment dans la phase amont de construction de ce dernier. Par conséquent, un système ETL mal conçu implique une complexité considérable dans le projet ED, mais aussi d'importants problèmes de fonctionnement, et de qualité de données.

Dans ce chapitre, nous allons, dans un premier temps, aborder l'importance que détient le processus ETL dans un projet ED. Par la suite, une section sera dédiée à l'étude des différentes approches et propositions dans le cadre de la modélisation du processus ETL.

2. POURQUOI MODELISER LE PROCESSUS ETL

Les projets de construction d'entrepôts de données et la maintenance de celui-ci sont couteux financièrement, et très exigeants en matière de qualité de données. Le processus d'alimentation de ces entrepôts est très complexe par ses tâches d'extraction, de transformation et de chargement. L'adoption d'une solution ETL est motivée par un grand besoin de données analytiques, mais surtout par des données décisionnelles qui peuvent jouer un rôle important dans le sort de l'entreprise.

Les outils ETL disposent d'une grande considération dans un projet entrepôt de données. En effet, ils détiennent 30% du budget réservé au projet, 55% dans les coûts du temps d'exécution de celui-ci et 80% dans le temps de développement [VAS 02].

Plusieurs facteurs participent dans la complexité de ce processus. D'une part les tâches complexes que doit résoudre le système ETL, et d'une autre, les exigences en matière de qualité de données qu'il doit fournir. De tels facteurs font que ce processus doit être bien étudié et documenté avant sa mise en œuvre afin d'éviter des pertes considérables : pertes de crédibilité ou pertes financières pour l'entreprise.

Quoi modéliser dans un système ETL ?

Il est nécessaire dans un projet d'entrepôt de données de pouvoir capturer le flux de données à partir des sources jusqu'au chargement dans les tables de l'entrepôt. Ce flux décrit les principales tâches que doit réaliser le système. En effet, la modélisation d'un processus ETL doit spécifier :

- Les différentes sources de données candidates pour l'extraction,
- Les tâches de transformation qui doivent s'effectuer sur les données,
- L'ordonnancement des différentes tâches,
- Le mappage entre les données sources et les données cibles.
- Etc.

Dans ce qui suit, nous allons voir de près certaines propositions dans le cadre de la modélisation du processus ETL

3. ETAT DE L'ART SUR LA MODELISATION ETL

3.1. Modélisation à base d'ontologies

Dans cette approche, un entrepôt de données est similaire à un système d'intégration dans son processus de construction [KHO 09]. En effet, un système d'intégration des sources a ainsi pour but de fournir une interface uniforme et transparente aux données pertinentes des différentes sources via un schéma global [GOM 05]. Les

travaux se basant sur cette réalité, ont adopté les ontologies afin de permettre une intégration sémantique et de manière automatique [KHO 09] où l'ontologie intégrante est considérée comme base et référence de conception.

Dans ce cadre, un premier travail [ROM 07] a proposé de générer le modèle conceptuel d'un ED à partir d'une ontologie couvrant les sources de données. Une deuxième proposition [KHO 09] a été basée sur le constat que la première proposition [ROM 07] soumet l'hypothèse d'existence d'une ontologie couvrant les sources de données. De ce fait, [KHO 09] ont proposé la méthode SISROM2C qui construit dans un premier temps une ontologie intégrante des sources de données au niveau sémantique, une ontologie locale est générée automatiquement à partir de cette ontologie, le modèle conceptuel est défini à partir de l'ontologie locale.

3.2. Modélisation de KIMBALL

3.2.1. Principe

Dans [KIM 04], les auteurs recommandent l'élaboration d'une documentation descriptive sur le processus ETL logique, contenant une description détaillée sur les sources de données et le modèle de l'entrepôt, ainsi que les différentes transformations qui doivent s'effectuer sur les sources. En effet, dans [KIM 04], les auteurs proposent de capturer un mappage logique des données qui décrit la relation entre les points extrêmes de départ et d'arrivée.

La description textuelle des sources de données et celle de l'entrepôt doit contenir : « Le nom de la table cible, le nom de la colonne cible, le type de la table[6], le type de SCD (Slowly Changing Dimension) type[7], la base de données source, le nom de la table source, le nom de la colonne source, la transformation ».

[6] Le type de la table peut être soit table de fait ou table de dimension

[7] Cet indicateur varie entre 1-2-3 pour chaque colonne dans la table de dimension. La valeur '2' exprime la nécessité de garder l'historique sur cette colonne, la valeur '1' indique la permission de réécriture de la valeur en cas de changement.

3.2.2. Points forts

Le contenu des sources de données et de l'entrepôt est bien détaillé.

Le type de SCD permet de gérer la stratégie d'historisation des dimensions dans l'entrepôt.

3.2.3. Point faibles

Cette contribution donne une vue technique sur le processus ETL, et ne propose pas de vue globale sur le processus dans les premières étapes du projet d'entrepôt de données.

3.3. Modélisation avec UML

3.3.1. Principe

Les auteurs [MOR 05] ont proposé la modélisation du processus ETL comme partie d'une approche globale et intégrante de modélisation d'un ED. En effet, les auteurs ont élaboré une méthode complète pour la modélisation de l'entrepôt de données depuis les sources de données jusqu'à la présentation de l'information à l'utilisateur final.

La méthode DWEP (Data Warehouse Engineering Process) proposée est divisée en cinq parties, et chacune comprend trois niveaux de conception. Le tableau suivant montre les différents diagrammes utilisés dans chaque partie.

	Source (S)	Integration	Data Warehouse (DW)	Customization	Client (C)
Conceptual	SCS Class diagram *Standard UML*	DM Class diagram *Data Mapping Profile*	DWCS Class diagram *Standard UML* *Multidimensional Profile*	DM Class diagram *Data Mapping Profile*	CCS Class diagram *Standard UML* *Multidimensional Profile*
Logical	SLS Class diagram *Different data modeling profiles*	ETL Process Class diagram *ETL Profile*	DWLS Class diagram *Different data modeling profiles*	Exporting Process Class diagram *ETL Profile*	CLS Class diagram *Different data modeling profiles*
Physical	SPS Comp. & deploy. diagrams *Database Deployment Profile*	Transportation Diagram Deployment diagram *Database Deployment Profile*	DWPS Comp. & deploy. diagrams *Database Deployment Profile*	Transportation Diagram Deployment diagram *Database Deployment Profile*	CPS Comp. & deploy. diagrams *Database Deployment Profile*

LEGEND: CS: Conceptual Schema, LS: Logical Schema, PS: Physical Schema, Comp. & deploy: Component and deployment

Tableau 2: Plateforme de modélisation d'un ED [MOR 05].

Cette approche utilise principalement les diagrammes UML. Toutefois, des profiles[8] ont été proposés pour apporter plus de perfectionnement aux diagrammes standards de ce langage.

Le processus ETL a été traité dans la partie d'intégration. Un diagramme de mappage a été proposé dans le *niveau conceptuel* afin de modéliser les relations entre les différentes sources de données impliquées et les tables de l'ED.

Au *niveau logique*, les auteurs ont proposé un modèle basé sur UML pour la modélisation du processus ETL qui traite avec certains problèmes communs du processus. On peut citer les principaux mécanismes proposés:

« Aggregation, conversion, filter, incorrect, join, loader, log, merge, surrogate, wrapper »

Dans le *niveau physique*, le diagramme de transport intégré (Transportation diagram) définit la structure physique du processus ETL utilisée pour le transport des données depuis les sources jusqu'à l'ED. Ce diagramme modélise les différentes plateformes utilisées, ainsi que les protocoles de communication entre ces plateformes. Les diagrammes utilisés sont principalement les diagrammes de paquetage et de déploiement.

3.3.2. Les points forts

Les auteurs se sont penchés sur l'utilisation d'un modèle standard, héritant ainsi de tous les avantages qui découlent de cette notation. En effet, le langage UML est connu et familier pour une grande communauté de concepteurs, ainsi, les efforts d'apprentissage d'un nouveau modèle sont réduits à la familiarisation avec les nouveaux profiles définis par cette méthode. Un autre point fort dû à l'utilisation d'UML est l'utilisation des diagrammes de paquetage. En effet, les diagrammes de paquetage décomposent le modèle en plusieurs pièces facilement gérées séparément. Ils donnent ainsi plusieurs niveaux de détails sur le modèle. Ils permettent ainsi de regrouper des éléments qui sont en forte adhésion dans des paquetages, ce qui

[8] Dans le jargon UML, une collection de raffinements qui étendent d'un type de diagramme existant est appelée 'PROFILE'

simplifiera davantage la vue globale du système, notamment dans les diagrammes de mappage et les diagrammes de transport intégrés.

3.3.3. Les points faibles

Dans cette proposition, la modélisation du processus ETL n'est effectivement traitée qu'après s'être attaqué au niveau logique de modélisation. Dans le niveau conceptuel, les diagrammes de mappage proposés capturent le flux de données entre les éléments de la source et ceux de la cible, mais ne mettent pas en évidence les transformations qui doivent être établies pour y répondre au besoin d'intégration.

3.4. Le méta modèle de VASSILIADIS et *al.*

3.4.1. Principe

Dans les travaux de [VAS 02] et [SIM 04], les auteurs ont proposé une nouvelle méthode de conception, de développement et d'optimisation du processus ETL. Cette méthode utilise un formalisme ad hoc, graphique, simple et intuitif dans la modélisation du processus ETL, notamment sur le niveau conceptuel.

L'objectif principal des auteurs dans ces travaux était de définir un niveau abstrait qui englobe un ensemble de constructeurs suffisamment génériques pour définir tous les cas qui peuvent se présenter. D'autres parts, cette méthode permet d'instancier à partir du méta-modèle générique une palette des éléments les plus usuels grâce au mécanisme d'instanciation.

La modélisation d'un processus ETL dans le cadre de cette approche commence par la définition d'un niveau conceptuel très simple du fait que cette modélisation a lieu dans les premières phases du projet d'entreposage. Le modèle logique, est généré par la suite avec le mappage du modèle conceptuel vers un modèle plus approfondi.

3.4.2. Points forts

La création d'une nouvelle méthode permettant de traiter les attributs d'un concept comme des éléments de première classe permettant ainsi d'arriver jusqu'au niveau de granularité le plus fin dans la modélisation des transformations et des relations

inter-attributs. Ainsi, les auteurs ont proposé la description d'un graphe d'architecture qui permet de décrire les détails du workflow.

D'autres parts, les mécanismes de généralisation et d'instanciation ont permis d'aboutir à un modèle générique et extensible.

3.4.3. Points faibles

La contribution des travaux de [VAS 02] et [SIM 04] s'inscrit dans les phases amont d'un projet d'entreposage (alimentation), et ne prend pas en considération le schéma de l'entrepôt de données. D'autres parts, le nouveau formalisme graphique proposé confronte une opposition par certains concepteurs classiques.

4. CONSTAT

L'étude des différents travaux proposés dans la littérature nous a permis de constater que peu de propositions traitent le processus ETL comme un workflow composite. En effet, quelques tentatives ont traité ce processus comme un problème d'intégration, et se sont penchés vers la modélisation du processus à base d'ontologies qui sont d'ailleurs considérées comme le *seul* artefact permettant de réconcilier des données hétérogènes au niveau sémantique [GOM 05]. D'autres travaux ont considéré le processus ETL comme un nouvel axe de recherche, et ont proposé des approches de modélisation du processus comme un workflow. Ces approches se sont intéressées principalement au problème de mappage inter-attributs, Cependant, dans la modélisation d'un processus ETL tant sur le niveau conceptuel ou le niveau logique, l'attribut représente le niveau de granularité le plus fin, et d'autres part, il représente l'unité principale des traitements les plus importants dans le contexte d'ETL (transformation et mappage). La considération d'un attribut en tant qu'élément de première classe n'a été effectivement proposée que dans le modèle de [VAS 02].

Le tableau suivant présente une comparaison entre les trois approches :

Approches / Critères	VASSILIADIS et al. [VAS 02], [SIM 04]	MORA [MOR 05]	KIMBALL [KIM 04]
Modèle	Ad-hoc	UML	Description textuelle + Modélisation dimensionnelle
Niveaux de modélisation	Conceptuel et logique	Conceptuel, logique et physique	Logique et physique
Aspects modélisés	processus ETL	Sources de données, processus ETL, entrepôt de données, présentation.	Processus ETL, Entrepôt de données
Modélisation en amont			
Extensibilité du modèle	Oui Mécanisme généralisation /instanciation	Non Ensemble fini de transformateurs	Non Représentation textuelle
Niveau conceptuel			
Type de modèle	Graphe	Diagramme de mappage	—
Mappage inter-attributs	Oui	Oui	Non
Représentation des transformations	Oui	Non	Non
Niveau logique			
Type de modèle	Graphe d'architecture	Diagramme du processus ETL	Tableau descriptif

Représentation des transformations	Activités, flux d'entrée, flux de sortie	Mécanismes prédéfinis (Filter, Aggregation, surrogate, join)	Langage naturel, langage de requêtes
Description de la sémantique des transformations	LDL : Langage de Description Logique	Utilisation des notes	Langage naturel, Langage de requêtes
Mécanismes de gestion	Gestion d'un flux d'exceptions	Les mécanismes : Log[9], incorrect[10]	—
Mappage entre niveau conceptuel et logique	Défini	Non défini	—
Niveau physique			
Type de modèle	—	Diagramme de transport intégré	—
Modélisation en aval			
Type de modèle	—	Data warehouse conceptual Schema, Multidimensional schema	Schéma en étoile, Etoile étendue

Tableau 3: Etude comparative entre les modèles de VASSILIADIS et al. Mora et KIMBALL.

En effectuant cette étude comparative entre les trois modèles de VASSILIADIS et *al.* MORA et KIMBALL, on a pu en tirer clairement la remarque que le méta modèle présenté par VASSILIADIS et *al.* s'est intéressé principalement aux phases en amont

[9] Le mécanisme « log » peut être connecté à n'importes quel autre mécanisme pour contrôler son activité

[10] Le mécanisme « incorrect » est utilisé pour rediriger les enregistrements rejetés et les exceptions vers une cible distincte.

de la construction d'un entrepôt de données. Effectivement, ce méta modèle a zoomé le processus d'alimentation, et il a proposé une méthode pour capturer le flux de données au niveau conceptuel et un mappage vers un modèle logique. Quand au modèle de KIMBALL et MORA, ils ont proposé la modélisation du processus ETL dans le cadre d'une approche globale de modélisation de l'entrepôt de données. En effet, les travaux de VASSILIADIS et al. ont proposé un modèle extensible en définissant leurs plateforme sur 03 couches, une couche générique, et une couche 'modèle' qui hérite de la première permettant ainsi au concepteur de définir de nouveaux éléments dans son modèle. Ceci n'étant pas permis dans la proposition de MORA qui définit un ensemble fermé de transformations.

Sur le niveau conceptuel, le diagramme de mappage de MORA modélise le mappage qui s'effectue entre les sources de données et l'entrepôt à plusieurs niveaux (mappage inter bases de données, mappage inter tables, et mappage inter attributs), la modélisation des transformations est faite sur le niveau logique ou le concepteur est doté d'un ensemble de transformateurs prédéfini. KIMBALL a contribué sur le niveau logique en proposant une représentation textuelle ou en requêtes des différentes transformations nécessaires lors d'un mappage. La seule contribution qui a représenté les transformations sur le niveau conceptuel a été celle de VASSILIADIS et al.

D'après cette étude, les avantages qu'on peut tirer sur le méta modèle de VASSILIADIS et al. sont en particulier :

- La proposition d'un méta modèle conceptuel qui répond aux besoins du concepteur dans les toutes premières phases de construction d'un entrepôt de données.
- Forte expressivité des transformations en raison de la valorisation d'un attribut comme composant élémentaire dans le modèle.
- La représentation du modèle logique par un graphe d'architecture qui permet la définition des activités, de différentes relations et ordonnancement entre les activités.

- Proposition d'un mappage de tous les éléments du modèle conceptuel vers ceux du modèle logique.

5. CONCLUSION

Au cours de ce chapitre nous avons pu constater que la proposition de VASSILIADIS et al. offre au concepteur les éléments nécessaires pour la modélisation tant sur le niveau conceptuel que logique. Dans le chapitre qui suit, nous allons présenter le méta modèle de VASSILIADIS et al. retenu pour l'implémentation de notre Framework.

Partie II

CHAPITRE IV

LE METAMODELE DE VASSILIADIS ET *al.*

1. INTRODUCTION

La motivation principale pour le modèle proposé dans [VAS 02] est le fait que, lors des premières étapes de la construction d'entrepôts de données, le concepteur intervient pour l'analyse de la structure et du contenu des sources de données existantes et leur mappage sur un modèle commun d'entrepôt de données. Par conséquent, un modèle formel pour cette tâche est nécessaire. Afin de modéliser le processus d'Extraction-Transformation-Chargement, les auteurs dans [VAS 02], [SIM 04] proposent deux niveaux de modélisation, conceptuel et logique et montrent le passage du modèle conceptuel vers le modèle logique.

2. LE META MODELE CONCEPTUEL

Le but principal de ce niveau de modélisation, est de fournir une description abstraite des entités importantes et des relations entre elles dans l'espace de problème en cours d'étude, indépendamment de toute technique d'implémentation. En effet, la modélisation du processus ETL dans le niveau conceptuel permet de faire une description rapide des sources de données et une définition des besoins d'analyse.

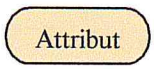
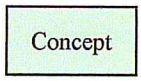
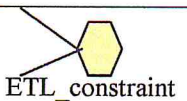
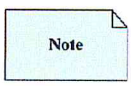
La majorité des contributions dans la modélisation conceptuelle d'ED est consacrée à la phase de restitution. Pour répondre à ce déficit, les auteurs dans [VAS 02] proposent un modèle conceptuel utilisable dans les premières étapes d'un projet ED basé sur la définition des objets manipulés dans le processus ETL (sources de données, transformations, contraintes, concepts, attributs,...etc.) ainsi que le mappage de ces objets vers ceux de l'ED. Il prend en charge les transformations fréquemment utilisées telles que l'assignation d'une clé primaire, les agrégations, ...etc.

2.1. Composants du modèle conceptuel

Un modèle conceptuel est caractérisé par le formalisme utilisé pour modéliser le schéma conceptuel. Le but de cette partie est de définir les entités utilisées pour capturer la sémantique du processus ETL proposé dans ce modèle. Ceci est introduit

par la proposition d'une notation graphique des entités et de les définir de manière formelle et détaillée.

Pour bien présenter notre travail, nous présentons les différents composants du modèle dans un tableau, ainsi que la définition et la représentation graphique de chaque composant.

Composant	Définition	Représentation graphique
Attribute	Un module granulaire de l'information. Le rôle des attributs est le même que dans le standard E/R	
Concept	Représente une entité dans la source de données ou dans l'ED	
Transformation	Abstraction d'un bout ou d'un module complet de code exécutant une tâche ETL : Nettoyage/ filtrage de données (violation de la contrainte PK/FK), transformations de données (agrégation).	
ETL-Constraint	Permet d'exprimer certaines contraintes sur le contenu de l'ED à travers les attributs de celui-ci (PK, FK, NOT NULL, …)	
Note	Permet d'expliquer des choix de conception, de préciser une sémantique ou une contrainte à vérifier en temps réel (temps d'exécution, événements, erreurs, …)	

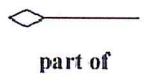
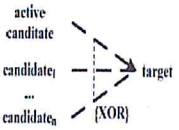
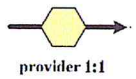
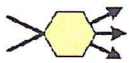
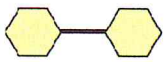
Les relations		
Part-Of	Permet de relier le concept à ses attributs	
Candidate	Exprime le fait que certains concepts cibles peuvent être alimentés par un ensemble fini de concepts sources candidats.	
Active candidate	Désigne le fait qu'un seul parmi plusieurs candidats est sélectionné pour l'alimentation du concept cible. Une relation *active candidate* est une spécialisation des relations candidates, avec la même structure et une sémantique raffinée.	
Provider	Transforme un ensemble d'attributs sources en un ensemble d'attributs cibles par une transformation. Le cas 1:1, la relation *provider* capture le fait qu'un attribut source alimente un attribut cible. Si les attributs sont compatibles, aucune transformation n'est nécessaire dans le mappage. Le cas N: M, la relation cache le mappage entre les attributs sources et attributs cibles.	 
Serial composition	Permet de modéliser une association « Provider » où les données sources passent par plusieurs transformations avant de donner naissance aux données cibles.	

Tableau 4 : Composants du méta modèle conceptuel.

2.2. Un exemple de processus ETL

Pour montrer l'utilisation de ce formalisme dans la construction d'un schéma conceptuel, la figure -13- présente un exemple de processus ETL qui traite sur un entrepôt de données pour la gestion de la scolarité.

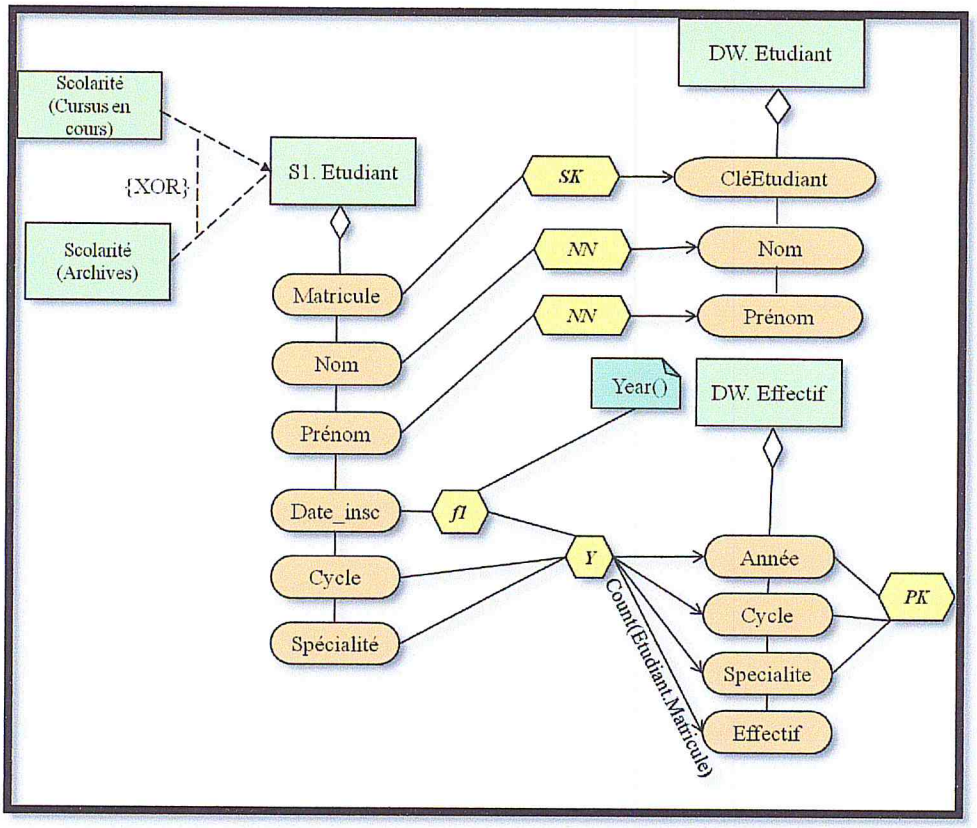


Figure 13: Exemple d'un schéma conceptuel du processus ETL.

L'exemple ci-dessus comporte une source de donnée S1 avec un concept Etudiant et un entrepôt de données DW avec deux concepts Etudiant et Effectif. Les concepts sources Scolarité (cursus en cours) et Scolarité (archives) sont candidats pour l'alimentation du concept cible Etudiant. Les attributs de DW.Etudiant sont alimentés depuis S1.Etudiant en utilisant les transformations Surogate key (SK), et Not Null (NN). Le Concept DW.Effectif reçoit ses données en exécutant les transformations f1 et une agrégation (γ).

2.3. Plateforme proposée pour le méta modèle conceptuel

Dans le méta-modèle proposé par VASSILIADIS, l'idée principale dans la modélisation conceptuelle est basée (a) sur l'identification d'un ensemble restreint de constructeurs génériques qui sont assez puissants pour capturer tous les cas possibles (généricité) et (b) sur un mécanisme d'extensibilité qui permet la construction d'une palette des types fréquemment utilisés [VAS 02].

La plateforme de modélisation du processus ETL est divisée en trois couches. La figure suivante illustre les mécanismes d'instanciation et de spécialisation entre ces dernières.

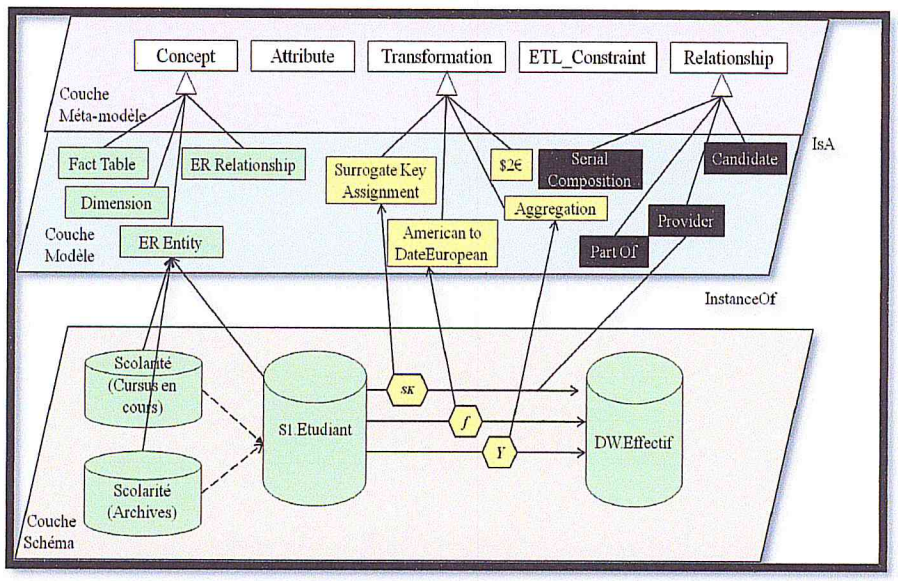


Figure 14: Plateforme de modélisation du processus ETL (niveau conceptuel).

2.3.1. La couche métamodèle (metamodel layer)

Cette couche comporte les méta-classes: *Concept, Attribute, Transformation, ETL_Constraint* et *RelationShip*. Ces méta-classes génériques permettent de modéliser tous les cas d'un processus ETL.

2.3.2. La couche modèle (template layer)

Afin de rendre le méta modèle d'une plus grande utilité, la couche modèle définit un ensemble de constructeurs spécifiques qui constitue un sous ensemble de la couche

méta modèle (relation IsA). En effet, dans cette couche, un mécanisme d'extensibilité permet la construction d'une palette de types et de fonctions fréquemment utilisés.

2.3.3. La couche schéma (Schema layer)

La couche *schéma,* comporte un scénario ETL spécifique. Toutes les entités de la couche schéma sont des instances des méta-classes de la couche méta-modèle ou de leurs sous classes de la couche modèle (relation InstanceOf).

3. LE META MODELE LOGIQUE

Le modèle logique est une représentation du système tel qu'il sera implémenté dans la machine. Il permet donc de décrire la structure des données utilisées sans faire référence à un langage de programmation. Il s'agit aussi de préciser le type des données utilisées lors des traitements. Le modèle logique proposé dans [SIM 04] capture les flux de données à partir de la source vers l'ED grâce à la combinaison des activités et des enregistrements de données.

3.1. Le graphe d'architecture

Dans [SIM 04], les auteurs ont proposé un méta modèle logique formel des activités pour un environnement ETL. La considération d'un processus ETL en tant que workflow a mené à la modélisation des activités, des enregistrements et des fonctions par un graphe appelé *graphe d'Architecture*. En effet, ce graphe d'architecture permet la représentation de la structure interne des activités mais aussi la modélisation du scénario ETL de façon à offrir différentes vues sur le processus (vue globale du système, zoom sur une activité).

3.1.1. Caractéristiques du graphe d'architecture

Le graphe d'architecture comporte des sommets et des arêtes. Les sommets représentent les types de données utilisés, les types des fonctions, les constantes, les attributs, les activités, les enregistrements, les paramètres et les fonctions. Les arêtes représentent les différents types de relations entre ces entités. Par conséquent, un graphe d'architecture couvre :

- Les activités et les enregistrements de données (RecordSet) du scénario, avec leurs composants.
- Les flux des données (Dataflow) dans l'environnement ETL.
- Le typage des entités (DataTypes) et la régulation de l'exécution (Execution sequence) d'un scénario, par des paramètres spécifiques.

3.1.2. Les sommets dans le graphe d'architecture

a. Les entités élémentaires

DataTypes: Il est supposé l'existence d'un ensemble fini, dénombrable de types de données. Chaque type de données T est caractérisé par : (a) un nom, et (b) un domaine, c.-à-d., un ensemble comptable de valeurs, appelé dom(T). Les valeurs des domaines sont référencées comme *des constantes*.

Attributes : Les attributs sont caractérisés par : (a) un nom, et (b) un type de données. Le domaine d'un attribut est un sous-ensemble du domaine de son type de données. Les attributs et les constantes sont référencés comme étant des termes.

Schema : est une liste finie d'attributs. Toute entité caractérisée par un ou plusieurs schéma est appelée *entité structurée*.

b. Les enregistrements (RecordSet)

Un enregistrement est défini comme l'instanciation d'un schéma à une liste de valeurs appartenant aux domaines des attributs respectifs au schéma. Un enregistrement est formellement caractérisé par : (a) un nom, (b) un schéma logique, (c) une extension physique.

c. Function

Il est supposé l'existence d'un ensemble fini *de types de fonctions*. Un type de fonction comporte : (a) un nom, (b) une liste finie *de types de données des paramètres*, et (c) un seul *type de retour de données*.

d. Activity

Les activités sont considérées comme des abstractions logiques représentant des parties ou des modules de codes complets. L'exécution d'une activité est assurée par un programme particulier. Les activités ETL sont représentées par un langage LDL[11] qui définit le code source d'une activité d'un côté, et d'un autre, évite le traitement des spécificités d'un langage particulier.

Une activité est formellement décrite par les éléments suivants :

(a) - *Un nom(Name)* : identificateur unique de l'activité.

(b) - *Schémas d'entrées (Input Schemata)* : ensemble fini de schémas d'entrée qui reçoivent les données à partir des providers de l'activité.

(c) - *Schéma de sortie (Output Schema)* : schéma qui décrit l'emplacement réservé aux lignes vérifiant le contrôle exécuté par l'activité.

(d) - *Schéma des rejets (Rejections Schema)* : schéma décrivant l'emplacement réservé aux lignes qui ne vérifient pas le contrôle exécuté par l'activité, ou qui ont des valeurs non appropriées pour l'activité en cours.

(e) - *Liste des paramètres* (*Parameter List*): ensemble de paires qui agissent comme régulateur [12] pour la fonctionnalité de l'activité. Le premier composant dans la paire est un nom, le deuxième est un schéma, un attribut ou une constante.

(f) - *Sémantique opérationnelle des sorties* (*Output Operational Semantics*): Déclaration LDL qui décrit le contenu passé en sortie de l'opération par rapport à ses entrées. Une déclaration LDL définit : (i) l'opération appliquée sur les lignes qui passent par l'activité et (ii) le mappage implicite entre les attributs des schémas d'entrées et leurs attributs de schéma de sortie respectifs.

(g) - *Sémantique opérationnelle des rejets* (*Rejection Operational Semantics*) : Déclaration LDL qui décrit les enregistrements rejetés.

---

[11] LDL : Logical description language. Une description LDL permet de percevoir la sémantique d'une activité sans se soucier d'un langage de programmation particulier

[12] La notion de régulateur sera définie par la suite

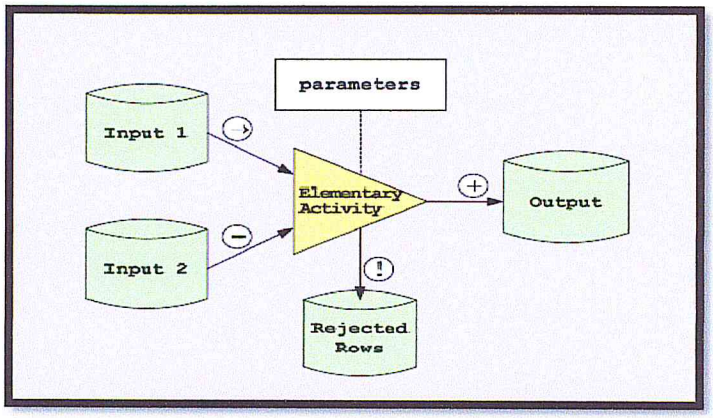


Figure 15: Le mécanisme d'activité élémentaire.

e. Représentation graphique des sommets

Composant	Notation graphique	Composant	Notation graphique
Type des données	Integer	Enregistrements	R
Type de fonction	$2€	Fonctions	my$2€
Constantes	1	Paramètres	rate
Attributs	PKEY	Activités	SK

Tableau 5: Représentation graphique des sommets dans un graphe d'architecture.

3.1.3. Les arêtes dans le graphe d'architecture

3.1.3.1. Les différents types des relations

a. PartOf

Met en relation les attributs et les paramètres avec les activités, enregistrements ou fonctions auxquels ils appartiennent. Chaque relation PartOf est annotée par le nom

du schéma (par défaut, on trouve les étiquettes IN, OUT, PAR, REJ pour indiquer respectivement si l'attribut appartient au schéma d'entrée, de sortie, de paramètre ou de rejet de l'activité).

b. InstanceOf

Permet de capturer les informations sur le typage des attributs et des fonctions. Elle met en relation un type de données ou un type de fonctions avec ses instances.

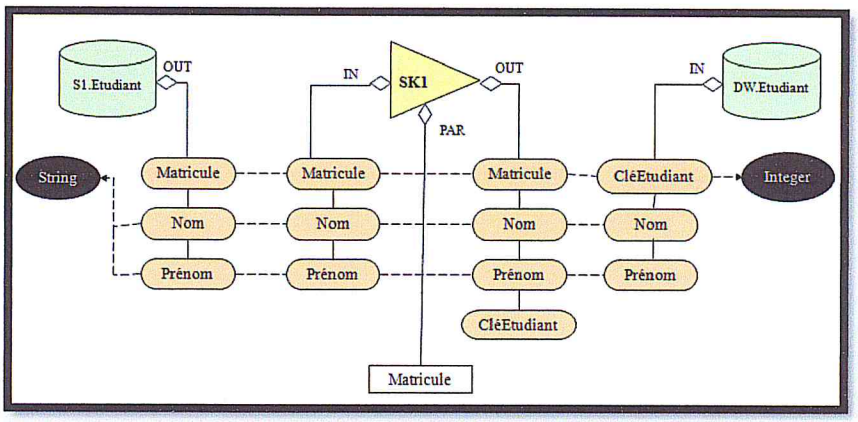


Figure 16: Les relations PartOf et instanceOf.

c. Regulator

Cette relation est définie entre les paramètres d'une activité et les termes (attributs ou constantes) qui alimentent cette activité.

d. Provider

L'entrée d'une activité peut être soit un enregistrement de données, ou une autre activité. Cette relation capture le passage des données entre fournisseurs (Providers) et consommateurs (Consumers) par la relation Provider entre les attributs des schémas concernés.

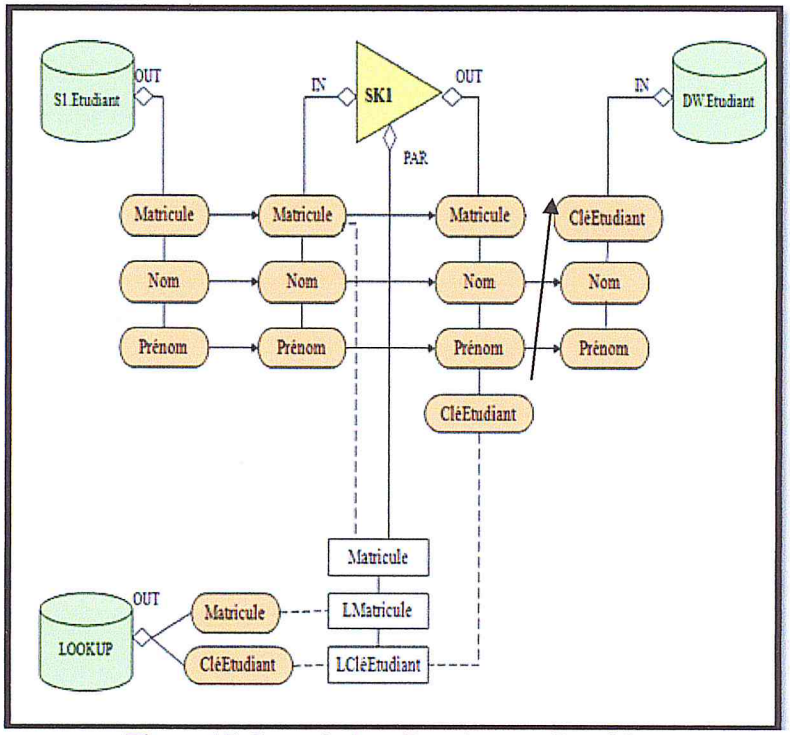


Figure 17: Les relations Regulator et Provider.

e. Derived provider

Cas particulier de la relation provider. Cette relation est utilisée lorsque les attributs de sorties sont générés par la composition des attributs d'entrées et des paramètres.

Formellement, supposons que (a) *source* est un terme (c.-à-d., attribut ou constante) dans le graphe d'architecture, (b) *cible* est un attribut dans le schéma de sortie d'une activité A et (c) x,y sont des paramètres dans la liste de paramètre de A. Une relation Derived Provider pr(*source*, *target*) existe SSI les relations Regulator suivantes existent : rr1 (*source*, x) et rr2 (y, *cible*).

3.1.3.2. Représentation graphique des relations

Relation	Notation graphique	Relation	Notation graphique
PartOf		Provider	
InstanceOf		Derived provider	
Regulator			

Tableau 6: Représentation graphique des arêtes dans un graphe d'architecture.

3.1.4 Exemple de processus ETL

La figure -18- décrit un processus ETL d'alimentation d'une table d'entrepôt de données DW.Etudiant au niveau logique.

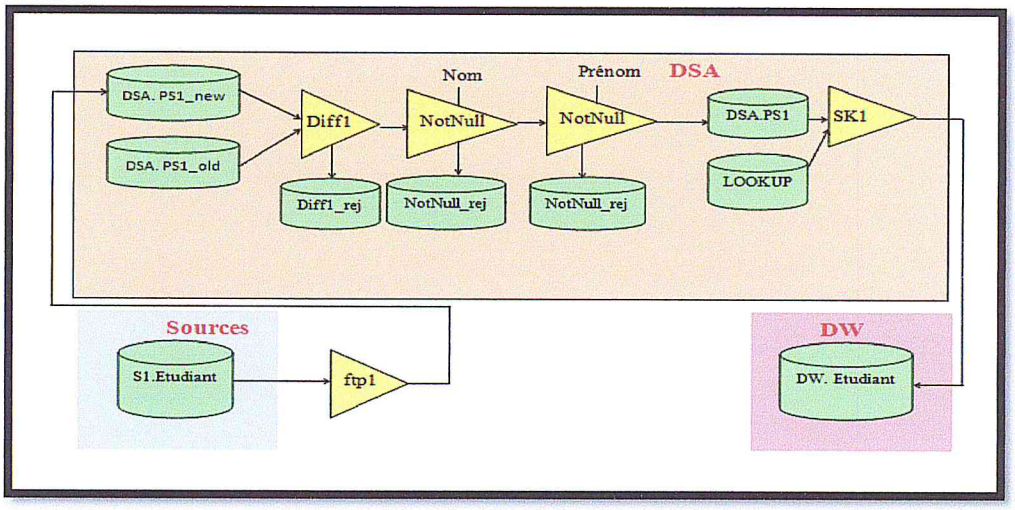


Figure 18: Exemple d'un schéma logique du processus ETL.

Le processus ETL illustré dans la figure -24- capture le flux de données entre une source S1.Etudiant vers une table DW.Etudiant de l'entrepôt de données en passant par la zone de préparation (DSA) où une séquence d'activités s'exécute. L'activité *ftp1* transfert les données à partir de la source vers le DSA, l'activité *diff1* fait la différence entre deux RecordSet et propage le résultat vers les activités NotNull. Les enregistrements rejetés par ces activités vont dans des recordSet de rejets (Diff1_rej, NotNull_rej…). Les enregistrements qui passent par toutes les activités se voient assignés une clé de substitution par l'activité SK1, et se chargent dans la table DW.Etudiant. L'utilisation du Recordset « DSA.PS1_old » suppose qu'il s'agit d'une extraction avec capture différentielle (utilisation des snapshots) pour détecter la différence entre les données sources et le contenu de l'entrepôt de données.

3.2. Plateforme proposée pour le modèle logique

Les principes de généricité et d'extensibilité constituent les idées principales dans le méta modèle conceptuel que logique. La figure suivante illustre la plateforme qui offre la généricité et l'extensibilité au méta modèle logique.

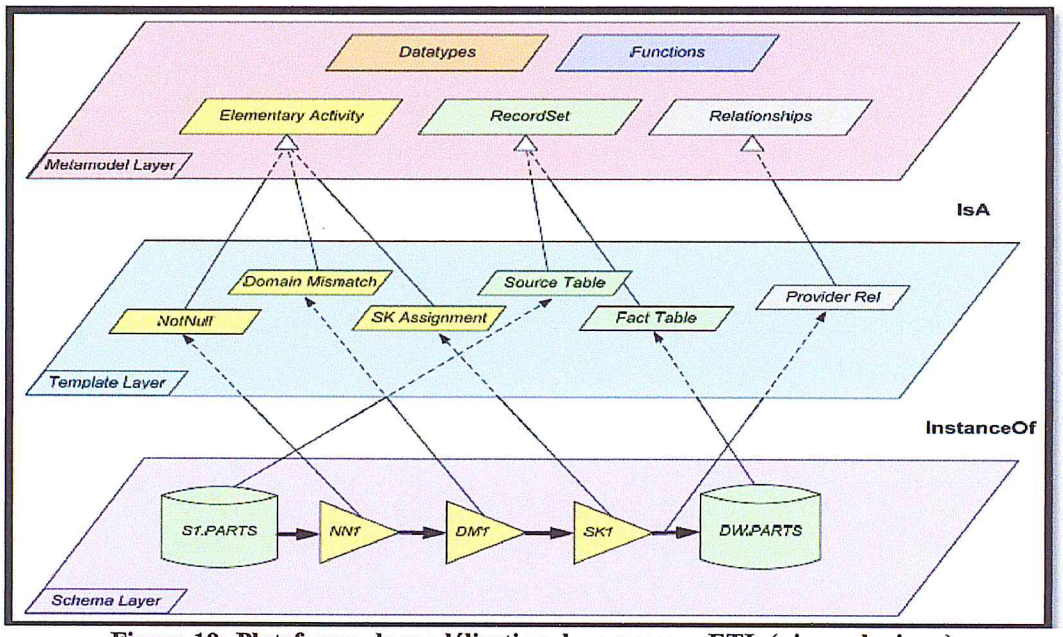


Figure 19: Plateforme de modélisation du processus ETL (niveau logique).

3.2.1. La couche méta modèle (meta model layer)

Cette couche comporte les méta-classes: *Datatypes, functions, elementary activity, Recordset et Relationships.* Cette couche implémente la généricité : les classes implémentées sont suffisamment génériques pour modéliser n'importe quel scénario ETL, avec l'instanciation appropriée.

3.2.2. La couche modèle (model layer)

Afin de rendre le méta modèle vraiment utile pour les cas pratiques d'activités ETL, il est enrichi avec un ensemble de constructeurs spécifiques, qui constituent un sous-ensemble de la couche méta modèle .En effet, les classes de la couche modèle sont des spécialisations des classes génériques de la couche méta modèle.

Grâce à ce mécanisme de personnalisation, le concepteur peut choisir les instances de la couche schéma à partir d'une palette plus riche et extensible de constructeurs.

3.2.3. La couche schéma (Schema layer)

La couche schéma, représente un scénario ETL spécifique. Toutes les entités de cette couche sont des instances des classes implémentées dans la couche méta modèle. Le lien entre les couches schéma et méta modèle est assuré par l'instanciation.

4. MAPPAGE DU MODELE CONCEPTUEL VERS LE MODELE LOGIQUE

Dans cette section, nous présentons la transition du modèle conceptuel vers le modèle logique dans le processus ETL. Durant la transition d'un modèle vers un autre, il est nécessaire d'identifier la correspondance entre les deux modèles. Puisque le modèle conceptuel est construit d'une manière plus générique, chaque entité conceptuelle correspond à une entité logique, ceci n'est pas valable dans le sens inverse.

Les sections précédentes montrent clairement que le modèle conceptuel n'est pas un workflow, il identifie simplement les transformations requises dans un processus ETL. La représentation des transformations dans le modèle conceptuel n'indique pas précisément l'ordre d'exécution. Cependant, le modèle logique représente un workflow, d'où l'importance de l'ordre d'exécution des activités. Par conséquent, une méthode est mise en œuvre pour la détermination semi-automatique d'un ordre correct d'exécution des activités dans le modèle logique, partout où c'est faisable, en regroupant les transformations du niveau conceptuel dans des étapes de transformations *ordre-équivalentes*.

Dans ce qui suit, on présente le mappage de chaque entité du modèle conceptuel vers l'entité correspondante du modèle logique. Dans le chapitre d'implémentation, on verra plus en détail le mécanisme de génération semi-automatique de chaque entité du modèle logique à partir des spécifications obtenues par le modèle conceptuel.

Composant conceptuel	Composant logique
Concept	Recordset
Concept attribute	Recordset attribute
Transformation	Activity
ETL Constraint	Activity
Note	Template activity (LDL Code)
part of	PartOf
Active candidate	Recordset

Candidate	----------
Provider	Provider
Serial composition	Sequence of activities

Tableau 7: Mappage des entités du modèle conceptuel vers celles du modèle logique.

5. CONTRIBUTION

Dans cette section, nous tentons à mettre l'accent sur certains points non explicites dans le méta modèle étudié, et nous reformulons certaines définitions afin d'affiner le méta modèle.

5.1. Interprétation des étapes du processus ETL dans un exemple

La figure -20- illustre les trois phases du processus ETL dans un exemple. En effet, en analysant un exemple modélisé avec ce méta modèle et en le comparant avec la figure -6- du chapitre II, qui porte sur l'environnement d'un processus ETL, on peut voir clairement et distinctement les phases du processus ETL (extraction, transformation et chargement) ainsi que leurs interactions avec les composants de l'architecture décisionnelle (sources de données, zone de préparation et entrepôt).

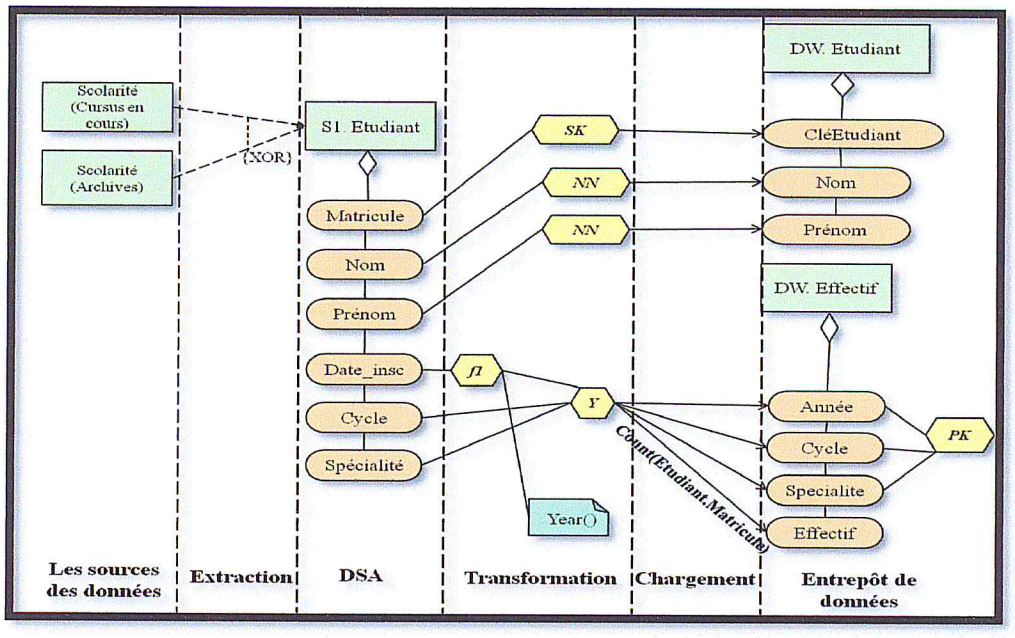


Figure 20: Séparation des phases du processus ETL.

5.1.1 Interprétation des relations 'Candidate' et 'Active Candidate'

Il devient clair que les relations 'candidate' et 'active candidate' permettent de spécifier l'alimentation de la zone de préparation (DSA) à partir des sources de données (la relation 'Candidate' illustre la phase d'extraction).

5.2. Analyse du méta modèle proposé pour le niveau conceptuel

5.2.1. Le méta modèle

La figure -21- illustre le diagramme de classe du méta modèle proposé par VASSILIADIS et *al.* pour la modélisation du niveau conceptuel du processus ETL tel qu'il a été présenté dans [VAS 02].

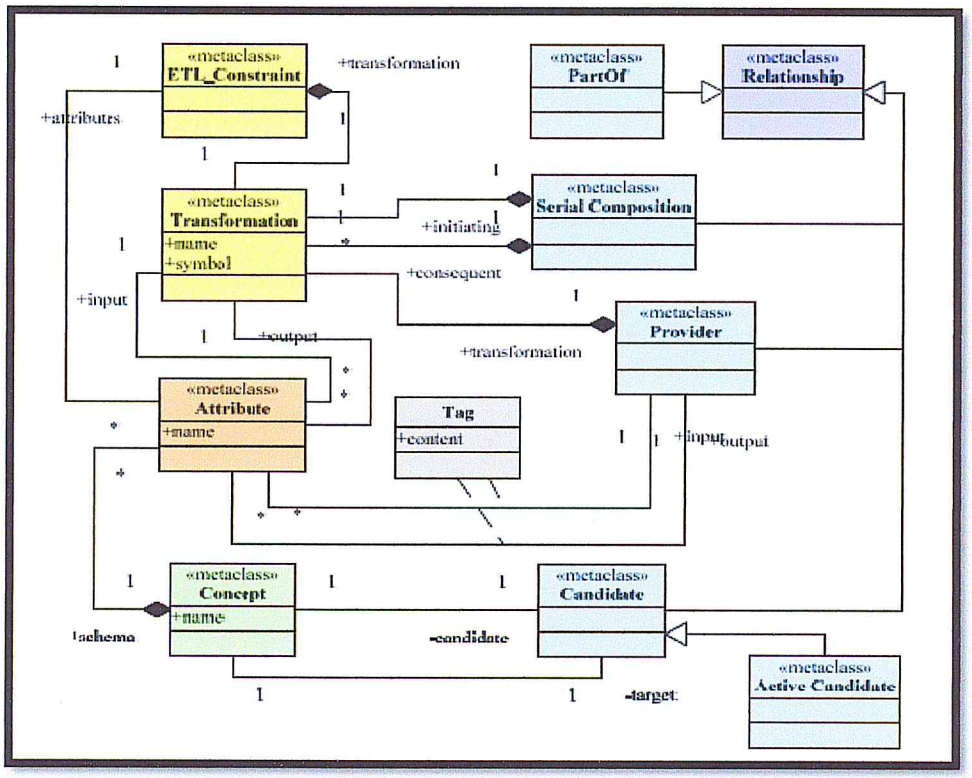


Figure 21: Diagramme UML du méta modèle conceptuel [VAS 02].

5.2.2. Analyse et valeurs ajoutées

En analysant ce diagramme, nous avons tiré les remarques suivantes :

a. La relation 'PartOf' dans le méta modèle

La relation 'PartOf' représentée en tant que méta-classe doit être associée avec les méta-classes 'Concept' et 'Attribute'. D'une part, cela permettra de représenter l'association entre un concept et ses attributs, et d'autre part elle évitera la relation de composition représentée entre les méta-classes 'Concept' et 'Attribute'.

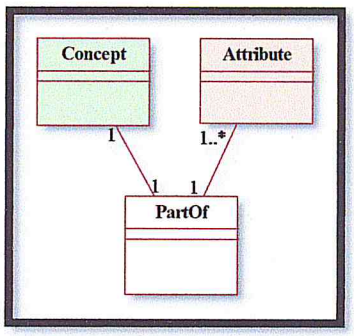


Figure 22: La relation 'PartOf' entre 'Concept' et 'Attribute'.

b. Représentation de la relation 'Candidate' (association 1 : 1 entre les méta-classes 'Concept' et 'Candidate')

La relation 'Candidate' possède plusieurs concepts candidats et un seul concept cible.

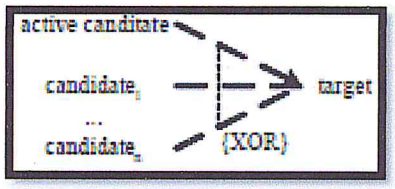


Figure 23: Représentation graphique des relations 'Candidate' et 'Active Candidate'.

Si on note *R* la relation représentée dans la figure -22- , et C1, C2, C3 les concepts candidats avec C4 comme concept cible. Selon le méta modèle, les associations sont comme suit :

Association des candidats	association de la cible
R1, C1	R1, C4
R2, C2	R2, C4
R3, C3	R3, C4

Chaque concept candidat est représenté par une relation 'Candidate'. De ce fait, les concepts candidats dans un concept cible seront dispersés dans plusieurs relations 'Candidate'.

Nous proposons de redéfinir cette relation de façon à ce qu'une relation 'Candidate' regroupe le concept cible et tous les concepts candidats.

En termes d'instances on aura :

Association des candidats	association de la cible
R, C1	R, C4
R, C2	
R, C3	

c. Les associations de la méta-classe 'Tag'

La méta-classe 'Tag' telle que représentée ne permet pas l'utilisation des notes avec tous les composants du modèle conceptuel. Il serait plus avantageux de rattacher cette méta-classe avec les méta-classes : 'Concept', 'Attribute', 'Transformation', 'RelationShip'.

d. Représentation exhaustive des sources de données et de l'entrepôt

En analysant le diagramme de la figure -21-, nous avons constaté la possibilité de représentation du contenu des sources de données, de la zone de préparation (DSA) et de l'entrepôt de données. Sachant que les concepts peuvent représenter les tables dans ces trois zones, une relation réflexive entre deux concepts peut représenter les relations entre les différentes tables (La méta-classe 'Attribute' est représentée en tant que méta-classe associative peut spécifier la clé étrangère utilisée dans le lien).

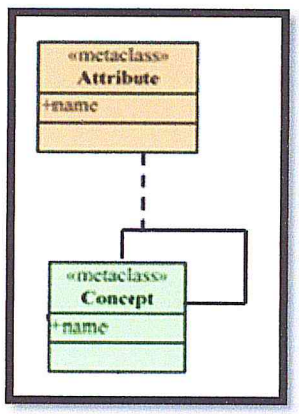


Figure 24: Représentation des associations entre concepts.

6. ARKTOS

ARKTOS est un prototype d'outil ETL développé dans le laboratoire des systèmes de bases de données et des connaissances (KDBSL : Knowledge and Database Systems Laboratory) de l'université nationale technique d'Athènes (NTUA) par l'équipe VASSILIADIS et al.

6.1. Fonctionnement général

ARKTOS implémente les modèles conceptuel et logique présentés dans ce chapitre. Le concepteur choisit les sources appropriées, construit le scénario ETL soit dans le niveau conceptuel ou dans le niveau logique, le propage vers l'optimiseur et obtient une version finale du modèle initial. Cette procédure est réalisée avec l'utilisation d'un référentiel de métadonnées qui relie les différentes parties de l'outil. La figure -25- montre les trois modules implémentés par ARKTOS, ainsi que les parties avec lesquels il interagit.

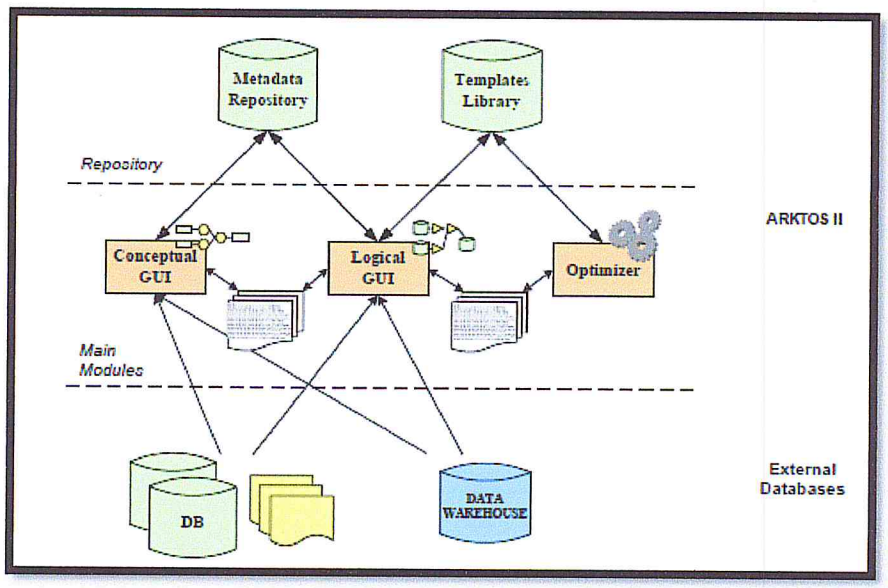


Figure 25: Architecture intérieure d'ARKTOS [SIM 04].

6.2. Les principaux modules

Les principaux modules d'ARKTOS sont (a) une GUI conceptuel, (b) une GUI logique, et (c) un optimiseur.

La **GUI conceptuelle** est utilisée pour la modélisation conceptuelle d'un processus ETL. Cette GUI est dotée des composants nécessaires pour la construction d'un scénario ETL au niveau conceptuel. Le résultat de cette étape peut être stocké dans le référentiel des métadonnées. La GUI conceptuelle est développée en MS Visual Basic .NET (Framework 1.1) [SIM 04].

La **GUI logique** est utilisée pour la modélisation logique d'un processus ETL. Le modèle logique peut être stocké dans le référentiel des métadonnées pour des utilisations ultérieures. Le résultat de cette étape peut être propagé vers l'optimiseur qui optimise le modèle et le retourne à la GUI. Dans ce processus, la GUI logique et l'optimiseur communique avec la bibliothèque des modèles afin de récupérer le code approprié pour chaque activité utilisée dans le processus. La GUI logique est développée en Ms Visual Basic.

L'optimiseur : ARKTOS implémente plusieurs algorithmes d'optimisation d'un workflow ETL : exhaustive, greedy et heuristique. Les entrées-sorties de ce module contiennent seulement des informations sur l'ordre d'exécution des activités.

6.3. Le stockage

La zone de stockage d'ARKTOS se compose de deux éléments essentiels pour le fonctionnement de cet outil. Le référentiel des métadonnées (Metadata Repository) est le lieu de stockage des différents modèles (conceptuel et logique), il sert aussi de moyen de communication entre les modules GUI conceptuelle et GUI logique. La bibliothèque des modèles (Template Library) contient le code pour tous les modèles prédéfinis dans le système ou bien définis par l'utilisateur. ARKTOS utilise les bases de données ORACLE pour le stockage des métadonnées et de la bibliothèque des modèles.

6.4. Bases de données externes

Représente les données avec lesquels interagit le système ETL lors de son exécution. En effet, l'outil ARKTOS permet d'explorer des bases de données afin de sélectionner les sources de données et les tables de l'entrepôt de données à alimenter par le processus.

6.5. Récapitulatif

Critère	Description
Intégration	-Les différents modules (Conceptual GUI, logical GUI et Optimizer) sont séparés et communique via les métadonnées.
Extensibilité	L'outil permet d'instancier des composants à partir de la bibliothèque des modèles, et il permet d'étendre cette bibliothèque avec des constructeurs définis par le concepteur.
Graphe d'architecture	-Les options de zoom-in, zoom-out d'une activité sont prises en compte dans l'outil dédié au modèle logique.
Contrôle	-Contrôle des types de données lors de l'association entre les entités élémentaires. -Les composants permis dans chaque phase sont les seules disponibles dans la palette.
Paramétrage	-L'outil permet de paramétrer les vues des différents composants graphiques (redimensionnement, couleur).
Stockage des données	- ORACLE, qui est un SGBD puissant, est utilisé pour le stockage des différents scénarios ainsi que les bibliothèques des modèles. -Gestion des comptes de concepteurs pour chaque utilisateur du système.
Convivialité	-Nécessité de connaissances préalables sur ORACLE pour la l'exploitation de l'outil. -Difficulté de manipulation des composants (déplacement, redimensionnement). -Difficulté de familiarisation avec l'environnement.

Description des recordSet.	-Un recordSet est représenté par l'outil de façon à ce qu'il permette de contenir des attributs appartenant à plusieurs sources de données dans le monde réel. Néanmoins, après sélection des attributs sources constituant le Recorset, le système ne relie pas ces derniers par une relation PartOf

Tableau 8 : Tableau Récapitulatif.

6. CONCLUSION

Le méta-modèle exposé dans ce chapitre représente un nouvel axe de recherche concernant le processus ETL et sa modélisation à différents niveaux. Ce métamodèle définit des notions et des représentations simples et consistantes, afin de modéliser les particularités du processus ETL. Il assure une grande souplesse de gestion des flux de données par la modélisation en workflow, et assure une grande qualité de données grâce à la séparation des besoins de modélisation sur des niveaux conceptuel et logique. D'autres parts, l'étude de l'outil ARKTOS nous a permis de faire le point sur certaines fonctionnalités, ce qui constitue des idées et de la matière permettant d'orienter nos choix en termes d'implémentation de notre propre outil.

Afin d'approfondir ses connaissances dans le méta-modèle, on propose aux lecteurs intéressés de lire le document [SIM 04] qui contient tous les détails sur le modèle.

PARTIE III

MISE EN ŒUVRE DU SYSTEME.

Partie III

CHAPITRE V

SPECIFICATION DES BESOINS & CONCEPTION DU SYSTEME

1. INTRODUCTION

La spécification des besoins et la conception sont des étapes indispensables dans la mise en œuvre d'une application logicielle. Au cours de ces étapes, le concepteur doit définir de façon très précise toutes les fonctionnalités que doit fournir le système ainsi que la façon d'y parvenir. Pour ce faire, nous avons opté pour la notation UML devenu un langage universel pour la modélisation objet des systèmes.

Pour la capture des besoins d'un concepteur ETL qui devra utiliser notre système, nous présenterons les diagrammes de cas d'utilisation, nous définissons l'aspect statique de notre système avec les diagrammes de classes, et par la suite les interactions entre objets seront modélisés avec les diagrammes de séquences.

2. LANGAGE DE MODELISATION

2.1. Le langage de modélisation UML

UML (Unified Modeling Language) est un langage formel, qui permet d'exprimer et d'élaborer des modèles objets, indépendamment de tout langage de programmation. Né de la fusion des méthodes objet dominantes (OMT, Booche et OOSE), puis normalisé par l'OMG en 1997. UML a plusieurs versions. La dernière version est 2.1.1 sortie en février 2007 [AUD 06].

3. SPECIFICATION DES BESOINS

3.1. Les diagrammes de cas d'utilisation

3.1.1. Définition des diagrammes des cas d'utilisation

Un diagramme de cas d'utilisation capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit. Il scinde la fonctionnalité du système en unités cohérentes, ayant un sens pour les acteurs. Les cas d'utilisation permettent d'exprimer le besoin des utilisateurs d'un

système, ils sont donc une vision orientée utilisateur de ce besoin au contraire d'une vision informatique [AUD 06].

3.1.2. Les acteurs du système

Le système interagit avec un seul type d'acteur ou d'utilisateur : 'le concepteur'. Le concepteur prend en charge tous les aspects et les étapes de modélisation du processus ETL.

3.1.3. Diagramme du cas d'utilisation global du système

a. Diagramme

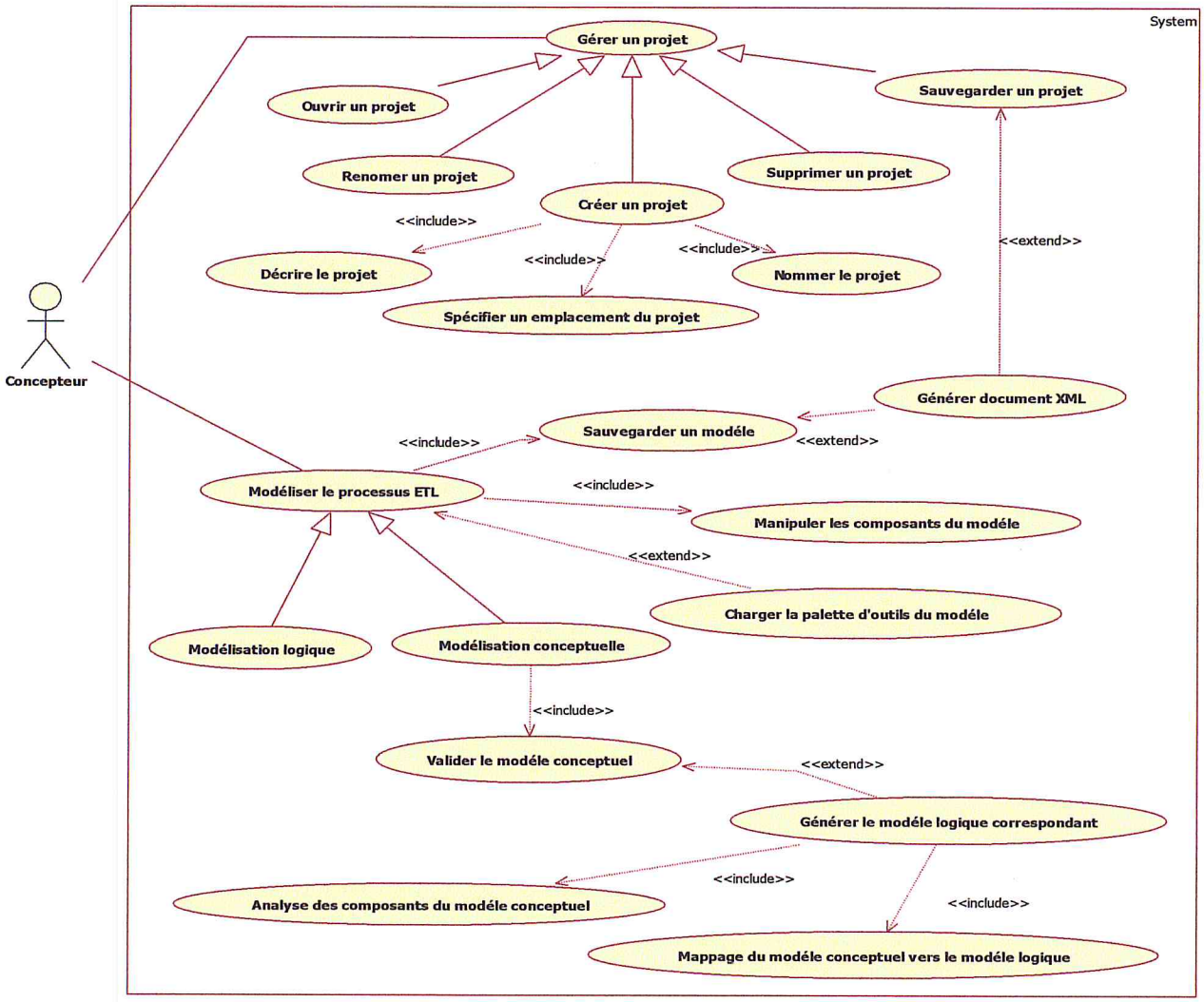


Figure 26: Diagramme du cas d'utilisation global du système.

b. Description textuelle du chaque cas d'utilisation

N	Le cas d'utilisation	Désignation
Cas d'utilisation concernant la gestion de projets		
1	Gérer un projet	Le concepteur gère le projet, en appliquant des différentes fonctionnalités (2, 6, 7, 8, 9).
2	Créer un projet	Le concepteur doit créer un projet pour qu'il puisse modéliser un processus ETL en passant par les étapes 3, 4 et 5.
3	Nommer le projet	La création d'un projet ne peut être faite qu'après spécifier le nom du projet souhaité par le concepteur du processus ETL.
4	Spécifier l'emplacement du projet	Le concepteur doit spécifier l'emplacement où le projet doit être sauvegardé.
5	Décrire le projet	Pour une raison d'organisation, il est préférable de donner une description sur le projet créé afin de résumer le contenu de celui-ci et de décrire son objectif.
6	Sauvegarder un projet	Le projet doit être sauvegardé dans un emplacement spécifique.
7	Ouvrir un projet	Les différents projets créés et sauvegardés dans des emplacements spécifiques sont toujours disponibles pour être ouverts dont le but de consultation ou bien de modification.
8	Renommer un projet	Après la création du projet, celui-ci pourra être

		renommé à n'importe quel moment.
9	Supprimer un projet	La suppression d'un projet provoque la suppression de son contenu (les modèles).
Cas d'utilisation concernant la modélisation		
10	Modéliser le processus ETL	Dans cette étape le concepteur commence à modéliser un processus ETL. Il existe deux types de modélisation (les cas d'utilisation 15, 20). Ce cas d'utilisation implique trois autres cas (11, 12 et 13).
11	Charger la palette d'outils du modèle	Le choix du modèle implique le chargement automatique de la palette d'outils du modèle correspondant.
12	Manipuler les composants du modèle	Le principe de modélisation d'un processus ETL se base sur la manipulation des différents composants du modèle. Les composants sont instanciés à partir de la palette d'outils et différentes fonctionnalités sont appliquées sur chaque composant. Le diagramme suivant détaille la manipulation d'un composant.
13	Sauvegarder un modèle	Un modèle doit être sauvegardé afin de pouvoir le consulter ou bien le modifier ultérieurement.
14	Générer document XML	-La sauvegarde d'un modèle implique la génération d'un document XML contenant tous les détails sur le modèle. -La sauvegarde d'un projet implique la génération d'un document XML contenant les spécifications sur le projet.

15	Modélisation conceptuelle	Ce cas d'utilisation permet de spécifier le modèle conceptuel. La modélisation d'un processus ETL commence par la modélisation conceptuelle.
16	Valider le modèle conceptuel	Ce cas d'utilisation permet la validation du modèle conceptuel avant le passage au modèle logique.
17	Générer le modèle logique correspondant	La validation du modèle conceptuel déclenche la génération du modèle logique correspondant. Ce cas est réalisé par les cas d'utilisation 18 et 19.
18	Analyse des composants du modèle conceptuel	Ce cas d'utilisation effectue l'analyse de tous les composants du modèle conceptuel, et les relations entre ces composants.
19	Mappage du modèle conceptuel vers le modèle logique	Chaque composant du modèle conceptuel est mappé vers le composant correspondant dans modèle logique selon les règles du modèle de VASSILIADIS et al.
20	Modélisation logique	La spécification d'un modèle logique se fait par la modification du modèle généré à partir du modèle conceptuel.

Tableau 9 : Description du diagramme du cas d'utilisation global du système.

3.1.4. Diagramme du cas d'utilisation « Manipuler un composant »

a. Diagramme

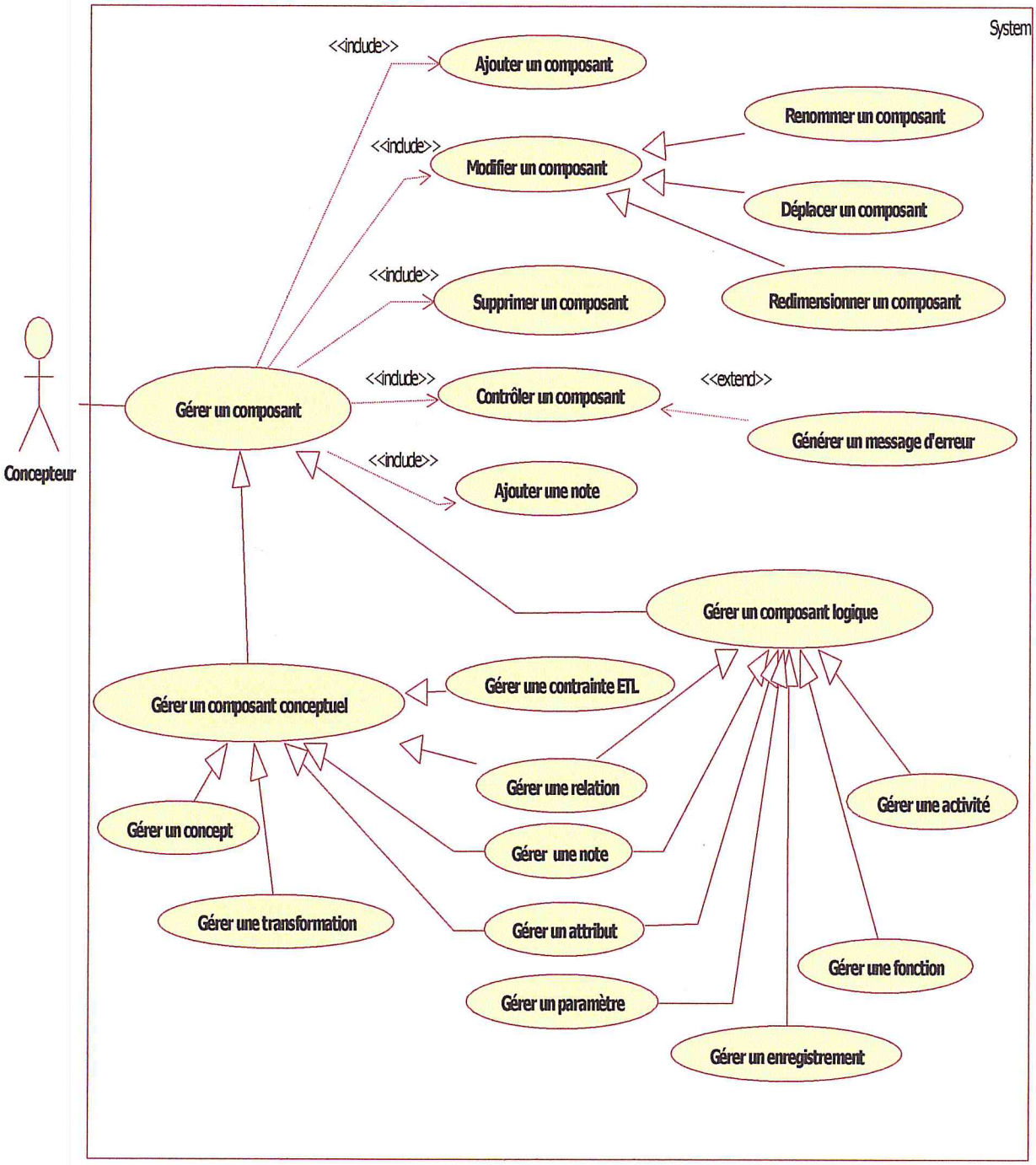


Figure 27: Diagramme du cas d'utilisation « Manipuler un composant ».

b. Description textuelle du chaque cas d'utilisation

N	Le cas d'utilisation	Désignation
1	Gérer un composant	Ce cas d'utilisation englobe toutes les manipulations d'un composant. La gestion d'un composant englobe plusieurs cas (2, 3, 7, 8, 9). On différencie entre deux types de composant : conceptuel et logique.
2	Ajouter un composant	Ce cas d'utilisation permet d'instancier un composant.
3	Modifier un composant	La modification d'un composant peut être une parmi les trois cas (4, 5, 6).
4	Renommer un composant	Ce cas permet de modifier le nom d'un composant.
5	Déplacer un composant	Ce cas permet de déplacer un composant dans la zone réservée à la modélisation.
6	Redimensionner un composant	Un composant peut être redimensionné en l'agrandissant ou bien en le réduisant.
7	Supprimer un composant	Un composant peut être supprimé du modèle.
8	Contrôler un composant	Ce cas d'utilisation permet de contrôler les actions exécutées sur un composant.
9	Générer erreur	Ce cas d'utilisation est déclenché lors de la détection d'erreurs d'une manipulation de composant.
10	Ajouter une note	Ce cas d'utilisation permet d'utiliser une note pour décrire un composant.

11	Gérer un composant conceptuel	La Gestion d'un composant conceptuel hérite du cas gestion d'un composant. Englobe tous les aspects de gestion d'un composant conceptuel.
12	Gérer un concept	Regroupe toutes les manipulations que peut subir un concept
13	Gérer une transformation	Regroupe toutes les manipulations que peut subir une transformation
14	Gérer une contrainte ETL	Regroupe toutes les manipulations que peut subir une contrainte ETL
15	Gérer une note	Regroupe toutes les manipulations que peut subir une note
16	Gérer un attribut	Regroupe toutes les manipulations que peut subir un attribut
17	Gérer un composant logique	La Gestion d'un composant logique hérite du cas gestion d'un composant. Englobe tous les aspects de gestion d'un composant logique.
18	Gérer un paramètre	Regroupe toutes les manipulations que peut subir un paramètre
19	Gérer un enregistrement	Regroupe toutes les manipulations que peut subir un enregistrement
20	Gérer une fonction	Regroupe toutes les manipulations que peut subir une fonction
21	Gérer une activité	Regroupe toutes les manipulations que peut subir une activité
22	Gérer une relation	Regroupe toutes les manipulations que peut subir une relation.

Tableau 10: Description du diagramme du cas d'utilisation « Manipuler un composant ».

4. CONCEPTION DU SYSTEME

4.1. Les diagrammes de classes

4.1.1. Définition des diagrammes de classes

Il s'agit d'une vue statique car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application. Chaque langage de Programmation Orienté Objets donne un moyen spécifique d'implémenter le paradigme objet (pointeurs ou pas, héritage multiple ou pas, etc.), mais le diagramme de classes permet de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier [AUD 06].

4.1.2. Diagramme de classes «Méta modèle conceptuel»

a. Le diagramme

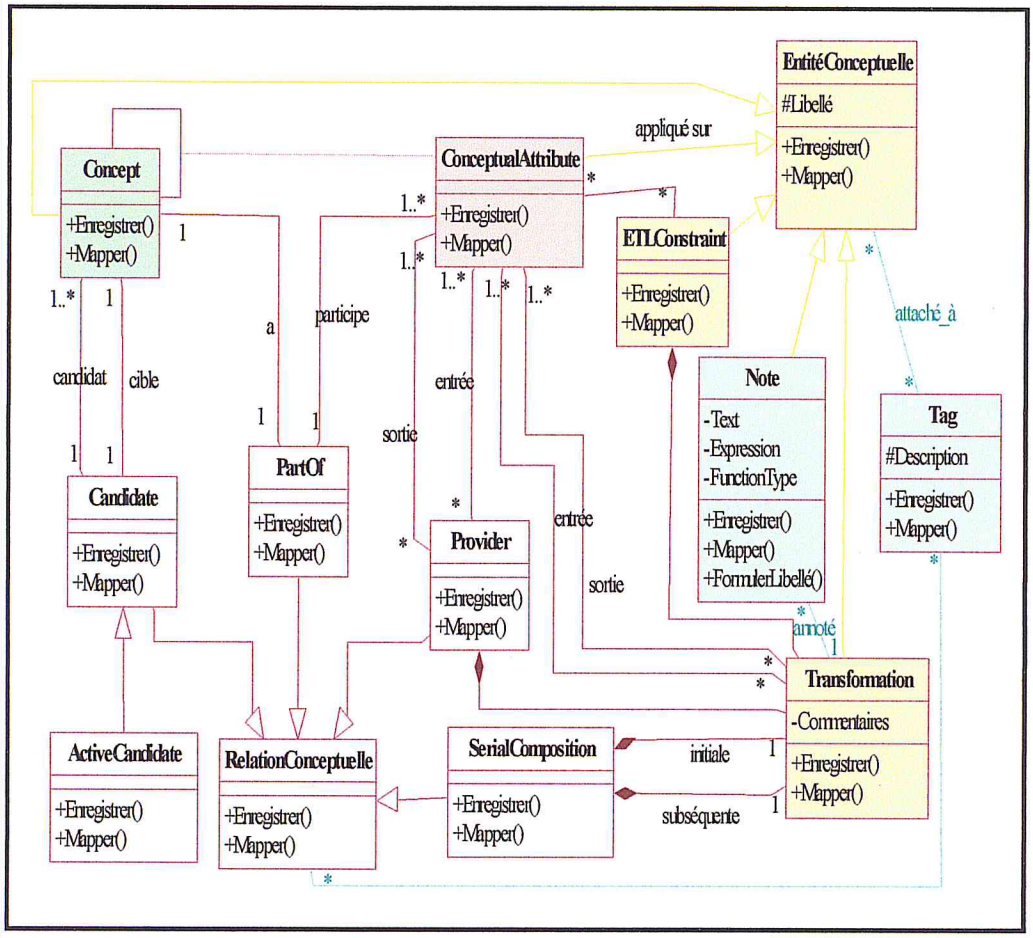


Figure 28: Diagramme de classe « Méta modèle conceptuel ».

a. Description textuelle des classes

Classe -1- EntitéConceptuelle Cette classe englobe toutes les entités du modèle conceptuel du méta modèle de VASSILIADIS (les classes 2, 3, 4, 5, 6). Toutes les classes héritant de cette classe redéfinissent les méthodes Enregistrer () et Mapper ().	
Attribut	Désignation

Libellé : Texte	Peut-être le nom ou bien un libellé d'une entité
Méthode	Désignation
Enregistrer ()	Permet de générer la représentation en XML de l'entité, et de l'enregistrer.
Mapper ()	Mapper une entité du modèle conceptuel vers l'entité du modèle logique correspondant.
Classe -2- Concept Cette classe modélise la notion de Concept introduite dans le méta modèle de Vassiliadis et al.	
Classe -3- ConceptualAttribute Cette classe modélise la notion de Attribute introduite dans le méta modèle de Vassiliadis et al.	
Classe -4- Transformation Cette classe modélise la notion de Transformation introduite dans le méta modèle de Vassiliadis et al.	
Attribut	Désignation
Commentaires : Texte	Contient les commentaires du concepteur sur la transformation.
Classe -5- ETLConstraint Cette classe modélise la notion de ETLConstraint introduite dans le méta modèle de Vassiliadis et al.	
Classe -6- Note Cette classe modélise la notion de Note introduite dans le méta modèle de Vassiliadis et al.	
Attribut	Désignation
Text : Texte	Description textuelle de la transformation.
Expression : Texte	Peut contenir une expression représentant l'abstraction d'une transformation.

FunctionType : Texte	Type de fonction de la transformation.
Méthode	Désignation
FormulerLibellé()	Regroupe une description, une expression et un type de fonction dans le libellé de la note.
Classe -7- Tag Cette classe modélise la notion de Note. Elle décrit une entité du modèle conceptuel.	
Attribut	Désignation
Description : Texte	Peut contenir une description sur une transformation, une relation, un concept…etc.
Classe -8- RelationConceptuelle Cette classe modélise les relations introduites dans le méta modèle de Vassiliadis et al.	
Classe -9-10-11-12-13 **PartOf, Provider, SerialComposition, Candidate, ActiveCandidate** Ces classes héritent de la classe RelationConceptuelle. Chaque relation relie entre des types spécifiques d'entité.	

Tableau 111: Description du diagramme de classe « Méta modèle conceptuel ».

4.1.3. Diagramme de classes «Méta modèle logique»

a. Diagramme

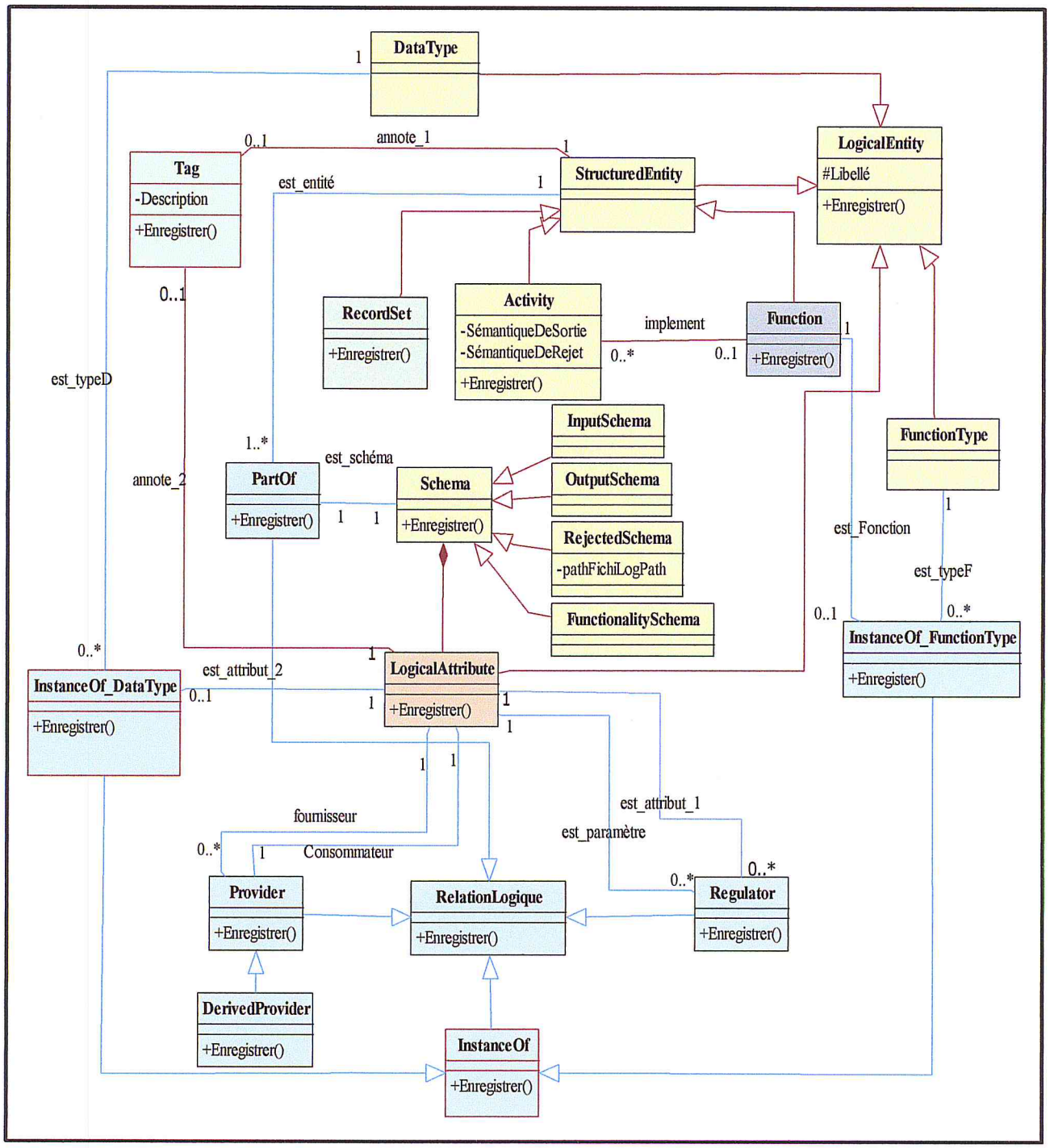


Figure 29: Diagramme de classes « Méta modèle logique ».

b. Description textuelle des classes

Ce diagramme met en évidence les différentes entités du méta modèle logique ainsi que les relations liant les différentes entités.

Classe -1- EntitéLogique Cette classe englobe toutes les entités introduites dans le méta modèle logique.	
Attribut	Désignation
Libellé : Texte	Peut-être le nom ou bien un libellé d'une entité
Méthode	Désignation
Enregistrer ()	Permet de générer la représentation en XML de l'entité, et de l'enregistrer. Toutes les classes héritant de cette super classe redéfinissent cette méthode.
Classe -2- LogicalAttribute Cette classe modélise la notion d'attribut telle qu'introduite dans le méta modèle logique.	
Classe -3- DataType Cette classe représente les différents types de données pour une entité élémentaire.	
Classe -4- FunctionType Cette classe représente les types de fonctions implémentés par une activité	
Classe -5- StructuredEntity Regroupe les entités ayant au minimum un schéma.	
Classe -6- Schema Un schéma est un ensemble d'attribut. Une entité structurée a une relation PartOf avec au minimum un schéma.	
Classe -7- RecordSet Cette classe représente la notion d'enregistrement (RecordSet) telle qu'introduite dans le méta modèle.	

Classe -8- Activity Cette classe représente la notion d'activité telle qu'introduite dans le méta modèle.	
Attribut	Désignation
SémantiqueDeSortie : Texte	Contient la sémantique des sorties de l'activité en langage LDL
SémantiqueDeRejet : Texte	Contient la sémantique des rejets de l'activité en langage LDL
Classe -9- Function Cette classe représente la notion de fonction telle qu'introduite dans le méta modèle.	
Classe -10- InputSchema Cette classe hérite de la super classe 'schéma'. Elle représente le schéma d'entrée d'une activité.	
Classe -11- OutputSchema Cette classe hérite de la super classe 'schéma'. Elle représente le schéma de sortie d'une activité.	
Classe -12- RejectedSchema Cette classe hérite de la super classe 'schéma'. Elle représente le schéma des rejets d'une activité.	
Attribut	Désignation
pathFichierLog : Texte	L'emplacement du fichier log qui contient les enregistrements rejetés.
Classe -13- RelationLogique Cette classe modélise les relations introduites dans le méta modèle de Vassiliadis et al.	
Méthode	Désignation

Enregistrer ()	Permet de générer la représentation en XML de l'entité, et de l'enregistrer. Toutes les classes héritant de cette super classe redéfinissent cette méthode.
Classe -14-15-16-17-18-19-20- PartOf, InstanceOf, InstanceOf_1, InstanceOf_2, Provider, DerivedProvider, Regulator Ces classes héritent de la classe RelationLogique. Chaque relation relie entre des types spécifiques d'entité.	
Classe -21- Tag Cette classe représente une note, elle est associée à toute entité structurée ou élémentaire.	
Attribut	Désignation
Description : Texte	Peut contenir une description sur une activité, une relation, un recordSet…etc.
Méthode	Désignation
Enregistrer ()	Permet de générer la description en XML et de l'enregistrer.

Tableau 122 : Description du diagramme de classes « Méta modèle logique ».

4.1.4. Diagramme de classes «Modèle»

a. Diagramme

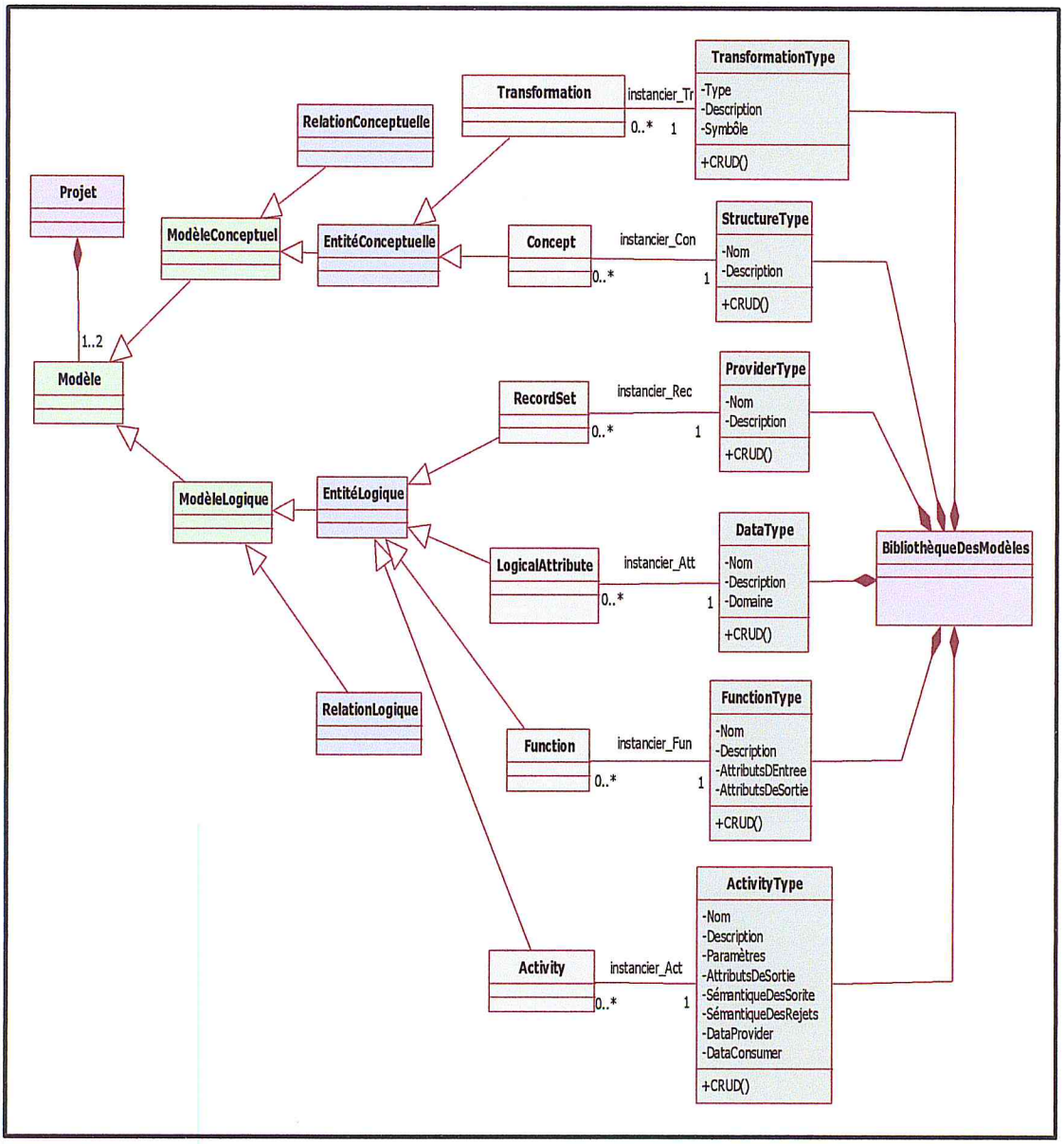


Figure 30: Diagramme de classe «Modèle».

b. Description textuelle

Le diagramme de classes 'Modèle' met en évidence les différentes entités d'un modèle de processus ETL, et leurs relations avec les types de données définis dans la bibliothèque des modèles.

Classe -1- BibliothèqueDesModèles Cette classe regroupe tous les modèles prédéfinis dans le système ou bien ajoutés par le concepteur. Elle se compose des classes (2, 3, 4, 5, 6, 7).	
Classe -2- TransformationType Cette classe modélise un type de transformation associé à une entité 'Transformation'.	
Attribut	Désignation
Nom : Texte	Nom spécifique au type.
Description : Texte	Description textuelle du type.
Symbôle : Texte	Symbôle représentant la transformation.
Méthode	Désingnation
CRUD ()	Create-Read-Update-Delete, permet la manipulation des différents types de transformation existants.
Classe -3- StructureType Cette classe modélise un type de structure associé à une entité 'Concept'.	
Attribut	Désignation
Nom : Texte	Nom spécifique au type.
Description : Texte	Description textuelle du type.
Méthode	Désingnation
CRUD ()	Create-Read-Update-Delete, permet la manipulation des différents types de structure existants.

Classe -4- ProviderType Cette classe modélise un type de fournisseur associé à une entité 'RecordSet'.	
Attribut	Désignation
Nom : Texte	Nom spécifique au type.
Description : Texte	Description textuelle du type.
Méthode	Désingnation
CRUD ()	Create-Read-Update-Delete, permet la manipulation des différents types de fournisseur existants.
Classe -5- DataType Cette classe modélise un type de donnée associé à une entité 'LogicalAttribute'.	
Attribut	Désignation
Nom : Texte	Nom spécifique au type.
Description : Texte	Description textuelle du type.
Domaine : Tableau de constantes	Ensemble des valeurs du type de données.
Méthode	Désingnation
CRUD ()	Create-Read-Update-Delete, permet la manipulation des différents types de donnée existants.
Classe -6- FunctionType Cette classe modélise un type de fonction associé à une entité 'Function'.	
Attribut	Désignation

Nom : Texte	Nom spécifique au type.
Description : Texte	Description textuelle du type
AttributsDEntrée : Liste d'attributs	Liste des attributs d'entrée
AttributsDeSortie : Liste d'attributs	Liste des attributs en sortie
Méthode	Désingnation
CRUD ()	Create-Read-Update-Delete, permet la manipulation des différents types de fonction existants.
Classe -7- ActivityType Cette classe modélise un type d'activité associé à une entité 'Activity'.	
Attribut	Désignation
Nom : Texte	Nom spécifique au type.
Description : Texte	Description textuelle du type.
Paramètres : Liste d'attributs	Liste des attributs qui constituent les paramètres de l'activité.
AttributsDeSortie : Liste d'attributs	Liste des attributs de sortie de l'activité.
SémantiqueDeSortie : Texte	Sémantique de sortie de l'activité.
SémantiqueDeRejet : Texte	Sémantique de rejet de l'activité.
DataProvider : Texte	Représente le type d'extraction à partir de l'entité 'RecordSet' en entrée de l'activité.

DataConsumer : Texte	Représente le type de chargement des sorties de l'activité dans l'entité 'RecordSet' en sortie de l'activité.
Méthode	Désingnation
CRUD ()	Create-Read-Update-Delete, permet la manipulation des différents types d'activité existants.
SémantiqueDeRejet : Texte	Sémantique de rejet de l'activité.
DataProvider : Texte	Représente le type d'extraction à partir de l'entité 'RecordSet' en entrée de l'activité.
DataConsumer : Texte	Représente le type de chargement des sorties de l'activité dans l'entité 'RecordSet' en sortie de l'activité.
Méthode	Désingnation
CRUD ()	Create-Read-Update-Delete, permet la manipulation des différents types d'activité existants.

Tableau 133:Description du diagramme de classes « Modèle».

4.1.5. Diagramme de classes « Couche applicative JAVA»

a. Diagramme

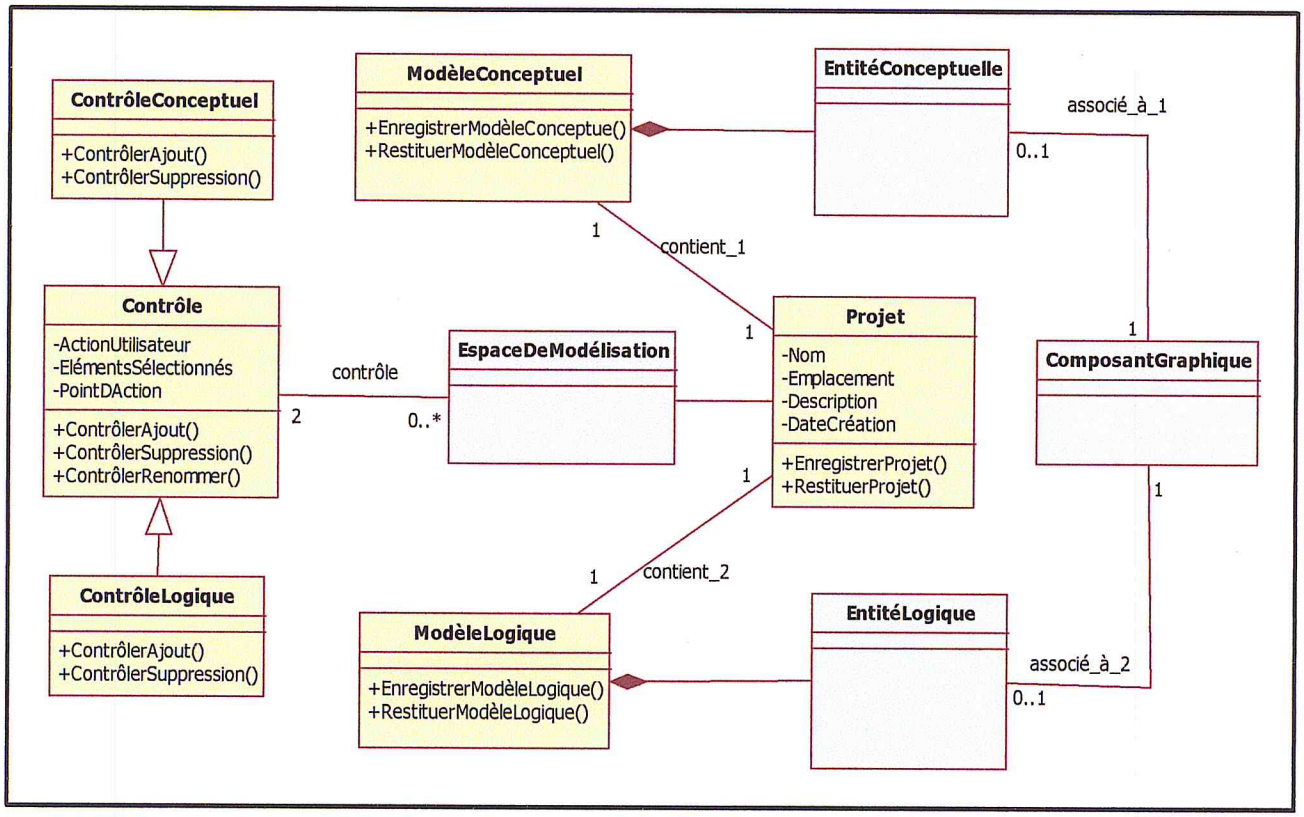


Figure 31: Diagramme de classes «Couche applicative JAVA».

b. Description du diagramme

Ce diagramme de classes modélise le fonctionnement global du système ainsi que la partie contrôle d'un projet.

Classe -1- Projet Cette classe regroupe toutes les propriétés d'un projet.	
Attribut	Désignation
Nom : Texte	Spécifie le nom du projet.
Emplacement : Texte	L'emplacement ou le projet doit être enregistré.

Description : Texte	Contient une description textuelle du projet.
DateDeCréation : Date	Date de création du projet.
Méthode	Désignation
EnregistrerPropjet()	Permet d'enregistrer un projet (génération de la représentation en XML.
RestituerProjet()	Restitue les propriétés du projet à partir d'un document XML.
Classe -2- ModèleConceptuel Représente un modèle conceptuel d'un processus ETL.	
Méthode	Désignation
EnregistrerModèleConceptuel()	Permet d'enregistrer les propriétés du modèle conceptuel (génération de la représentation en XML.
RestituerModèleConceptuel()	Restitue les propriétés du modèle conceptuel à partir d'un document XML.
Classe -3- ModèleLogique Représente un modèle logique d'un processus ETL.	
Méthode	Désignation
EnregistrerModèleLogique()	Permet d'enregistrer les propriétés du modèle conceptuel (génération de la représentation en XML).

RestituerModèleLogique()	Restitue les propriétés du modèle conceptuel à partir d'un document XML.
Classe -4- Contrôle S'occupe des contrôles appliqués sur les modèles	
Attribut	Désignation
ActionUtilisateur : Texte	Spécifie l'action choisie par l'utilisateur (ajout d'un enregistrement, suppression d'une transformation…).
ComposantsSélectionnés : Tableau d'Entité	Contient l'ensemble des composants sélectionnés par l'utilisateur.
PointDAction : Point	Spécifie le point dans la fenêtre de modélisation ou l'utilisateur désire ajouter un composant.
Méthode	Désignation
ContrôlerAjout()	Contrôle l'ajout d'un composant, et génère des erreurs lors d'une opération d'ajout non autorisée
ContrôlerSuppression()	Contrôle la suppression du composant sélectionné par l'utilisateur, et génère la suppression des composants dépendant en suppression.
ContrôlerRenommer()	Gère l'espace de noms, et contrôle la duplication.
Classe -5- ContrôleConceptuel C'est une spécialisation de la classe Contrôleur. S'occupe des contrôles appliqués sur les composants conceptuels.	
Méthode	Désignation

ContrôlerAjout()	Contrôle l'ajout d'un composant conceptuel, et génère des erreurs lors d'une opération d'ajout non autorisée.
ContrôlerSuppression()	Contrôle la suppression du composant conceptuel sélectionné par l'utilisateur, et génère la suppression des composants dépendant en suppression.
Classe -6- ContrôleLogique C'est une spécialisation de la classe Contrôleur. S'occupe des contrôles appliqués sur les composants logiques	
Méthode	Désignation
ContrôlerAjout()	Contrôle l'ajout d'un composant logique, et génère des erreurs lors d'une opération d'ajout non autorisée.
ContrôlerSuppression()	Contrôle la suppression du composant conceptuel sélectionné par l'utilisateur, et génère la suppression des composants dépendant en suppression.
Classes -7-8-9-10- **EspaceDeModélisation, EntitéConceptuelle, EntitéLogique, ComposantGraphique** Ces classes figurent respectivement dans les diagrammes 'Vues graphiques', 'Méta modèle Conceptuel', 'Méta modèle logique', 'Vues graphiques'.	

Tableau 144: Description du diagramme de classes «Couche applicative JAVA»

4.1.6. Diagramme de classes «Vues Graphiques»

a. Diagramme

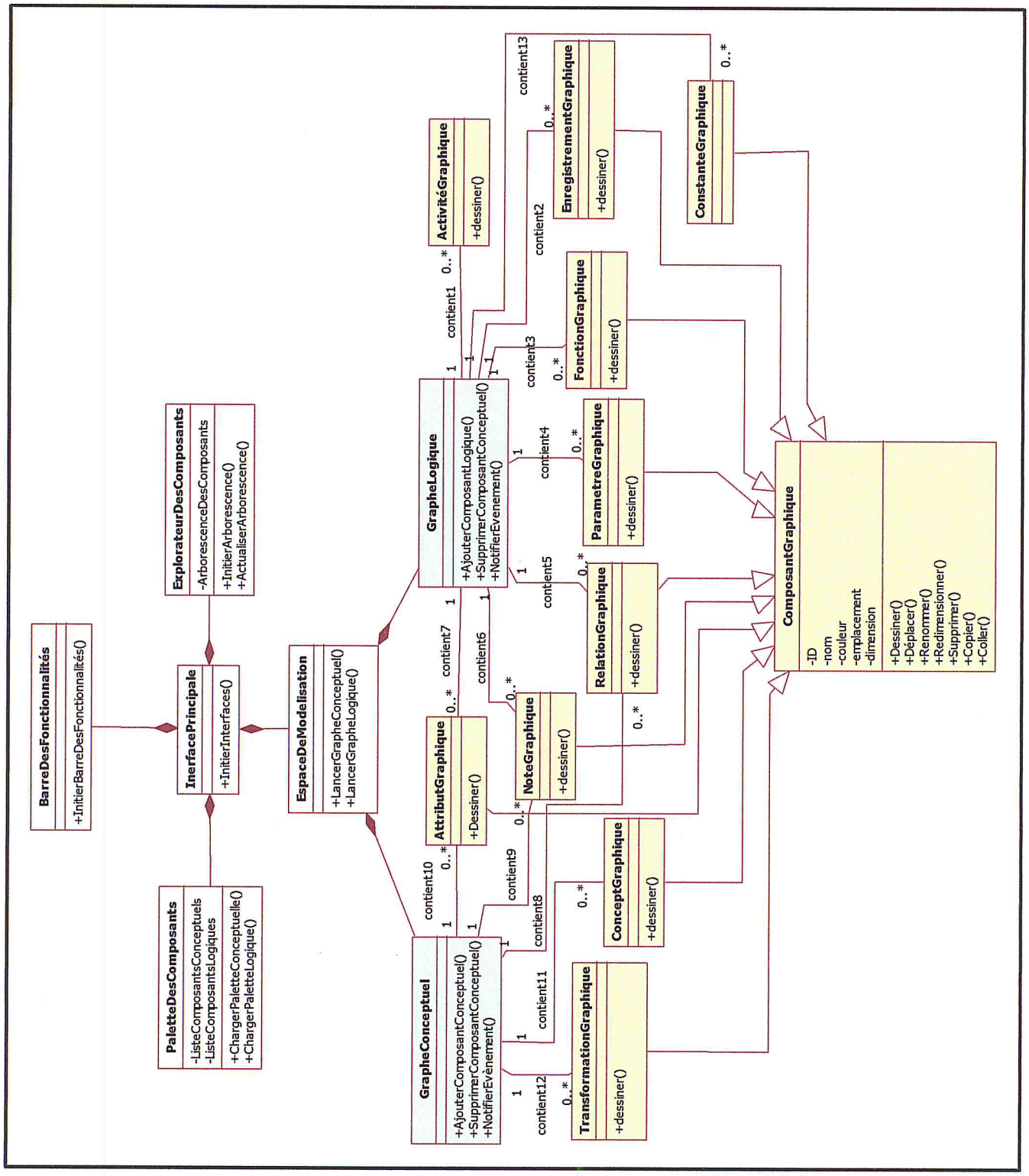


Figure 32: Diagramme de classes « Vues Graphiques ».

b. Description textuelle des classes

Ce diagramme de classes modélise les aspects graphiques de notre système ainsi que les relations entre les différentes classes.

Classe -1- InerfacePrincipale Cette classe rassemble toutes les vues graphiques du système. C'est le conteneur de tous les composants graphiques.	
Méthode	Désignation
InitierInterfaces()	Appelle les méthodes qui se chargent du lancement de tous les initiateurs des composants de l'interface graphique.
Classe -2- BarreDesFonctionnalités Cette classe contient une grande partie des fonctionnalités permises par le système.	
Méthode	Désignation
InitierBarreDesFonctionalités()	S'occupe de l'initialisation des fonctionnalités du système.
Classe -3- PaletteDesComposants C'est la classe qui s'occupe des composants conceptuels et logiques du modèle.	
Attribut	Désignation
ListeComposantsConceptuels : liste de composants	Contient une liste de composants conceptuels.
ListeComposantsLogiques : Liste de composants	Contient une liste de composants logiques.
Méthode	Désignation
ChargerPaletteConceptuelle()	Charge la palette des composants conceptuels vers l'interface à partir de l'attribut ListeComposantsConceptuels.

ChargerPaletteLogique()	Charge la palette des composants logiques vers l'interface à partir de l'attribut ListeComposantsLogiques.
Classe -4- ExplorateurDesComposants S'occupe des différents composants instanciés dans le modèle.	
Attribut	Désignation
ArborescenceDesComposants : Arbre	C'est une arborescence contenant tous les composants instanciés dans le modèle
Méthode	Désignation
InitierArborescence()	Permet le chargement de l'interface de l'arborescence.
ActualiserArborescence()	Modifie le contenu de l'arborescence lors de chaque modification dans le modèle.
Classe -5- EspaceDeModélisation Désigne l'environnement de gestion d'un modèle.	
Méthode	Désignation
LancerGrapheConceptuel()	Permet d'instancier un graphe conceptuel.
LancerGrapheLogique()	Permet d'instancier un graphe logique.
Classe -6- GrapheConceptuel Regroupe tous les composants du modèle conceptuel sous forme de graphe	
Méthode	Désignation
AjouterComposantConceptuel()	Permet l'ajout d'un composant au graphe conceptuel.

SupprimerComposantConceptuel()	Permet la suppression d'un composant du graphe conceptuel.
NotifierEvenement()	Ecoute et informe le système de toutes les interactions de l'utilisateur sur le graphe et sur les composants conceptuels.
Classe -7- GrapheLogique Regroupe tous les composants du modèle logique sous forme de graphe.	
Méthode	Désignation
AjouterComposantConceptuel()	Permet l'ajout d'un composant au graphe logique.
SupprimerComposantConceptuel()	Permet la suppression d'un composant du graphe logique.
NotifierEvenement()	Ecoute et informe le système de toutes les interactions de l'utilisateur sur le graphe et sur les composants logiques.
Classe -8- ComposantGraphique Modélise la graphique d'un composant	
Attribut	Désignation
ID : Entier	Identificateur unique pour chaque composant.
Nom : Text	Nom du composant ou bien l'étiquette.
Couleur : Color	Couleur spécifique au type de composant.
Emplacement : Point	Le point de référence dans l'espace de modélisation.
Dimension : Dimension	Les dimensions du composant (largeur et longueur).
Méthode	Désignation

Dessiner()	Permet de représenter graphiquement un composant.
Déplacer()	Déplace le composant dans la zone réservée à la modélisation.
Renommer()	Permet de modifier le nom du composant.
Redimensionner()	Modifie les dimensions d'un composant.
Supprimer()	Supprime un composant du graphe.
Copier()	Cree une copie du composant sélectionné.
Coller()	Déploie une copie 'un composant.
Classes -9-10-11-12-13-14-15-16-17- ConceptGraphique-AttributGraphique-TransformationGraphique-EnregistrementGraphique-ActivitéGraphique-FonctionGraphique-ParamètreGraphique-ConstanteGraphique-NoteGraphique Ces classes héritent toutes de la classe ComposantGraphique, néanmoins elles redéfinissent la méthode Dessiner().	
Méthode	Désignation
Dessiner()	Permet de spécifier la représentation graphique d'un composant particulier (attribut, activités…).
Classes -18- RelationGraphique Cette classe hérite de la classe ComposantGraphique. Tous les types de relations définis dans le métamodèle de VASSILIADIS sont des classes héritant de cette classe (PartOf, Provider, Regulator…). Pour des raisons de lisibilité, ces classes n'ont pas été intégrer dans le diagramme.	

Tableau 155 : Description du diagramme de classes « Vues Graphiques ».

4.2. Les diagrammes de séquences

4.2.1. Définition des diagrammes de séquence

Le diagramme de séquence permet le dialogue entre les objets dans le temps. Il détaille la communication entre objets (l'enchaînement des opérations; quels messages sont envoyés; quand et à qui sont envoyés les messages.)

Le diagramme de séquence permet de représenter à l'utilisateur la dynamique du futur système qui sera mis en place. Ce diagramme fait référence au diagramme de cas d'utilisation et à un diagramme de classe.

4.2.2. Diagramme de séquence « Créer un projet »

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

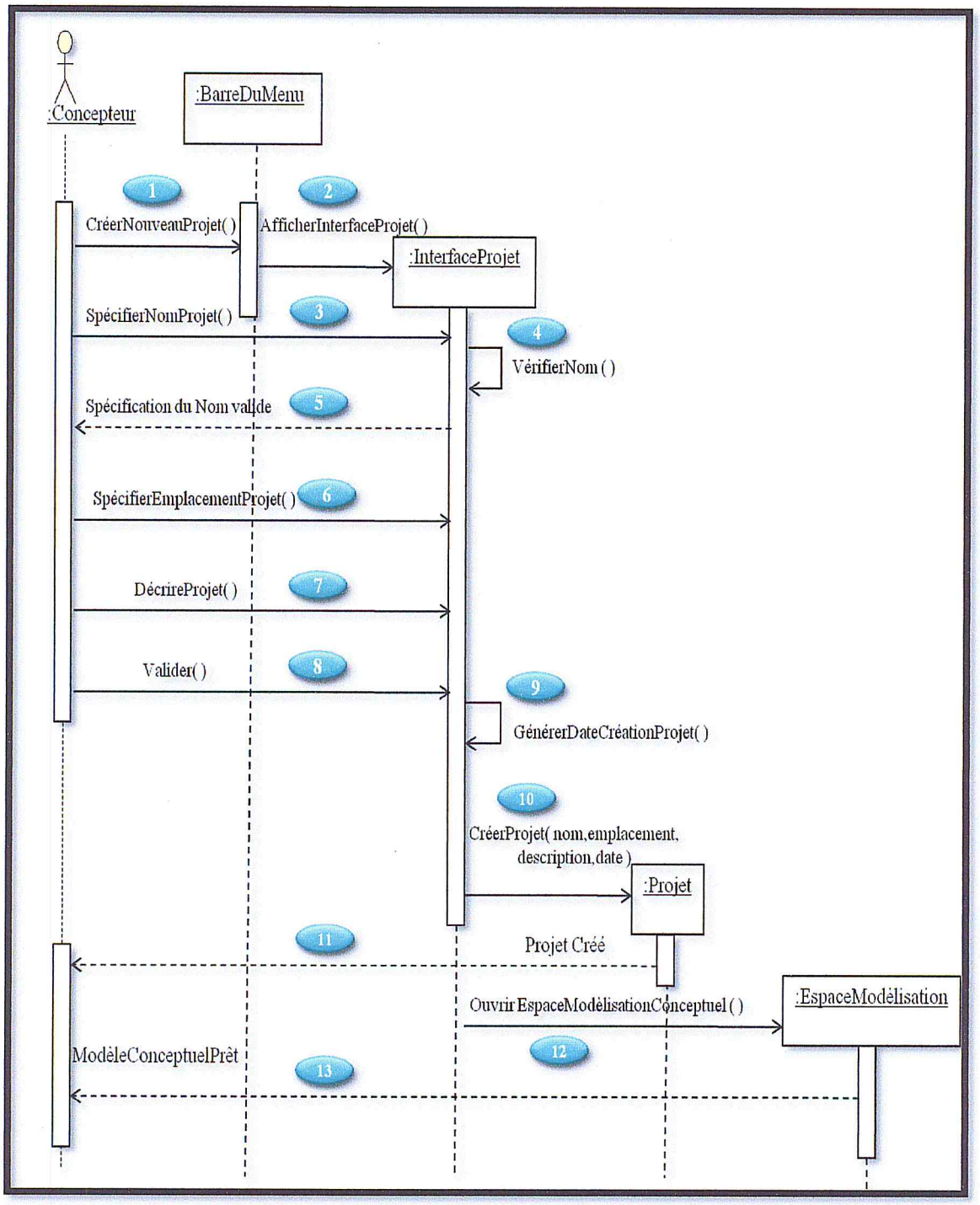


Figure 1:Diagramme de séquence « Créer un projet ».

4.2.3. Diagramme de séquence « Ouvrir un projet »

Ce scénario décrit la séquence d'étapes suivie pour l'ouverture d'un projet existant dans l'espace de travail.

4.2.4. Diagramme de séquence « Ajouter une transformation »

Ce diagramme représente les interactions entre les objets intervenant dans l'ajout d'une nouvelle transformation. L'ajout d'un concept, une activité, un recordSet, une activité ou une fonction se produisent de la même manière avec quelques différences dans les objets intervenant.

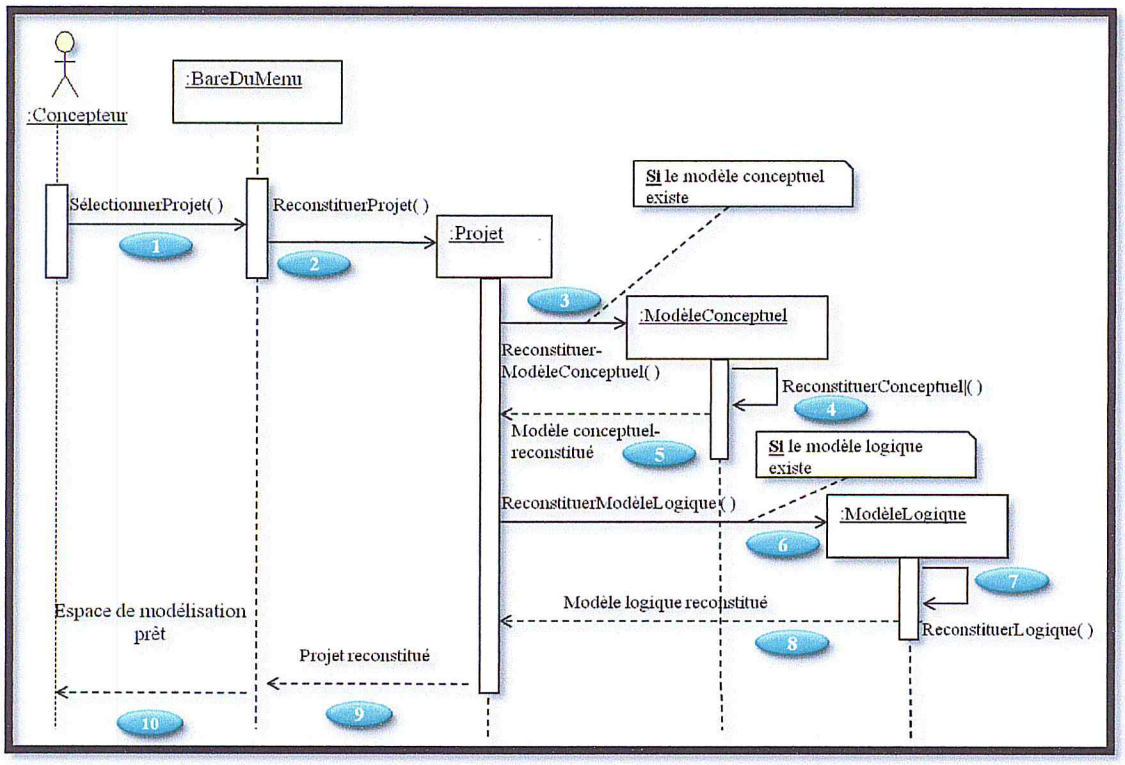


Figure 2: Diagramme de séquence « Ouvrir un projet ».

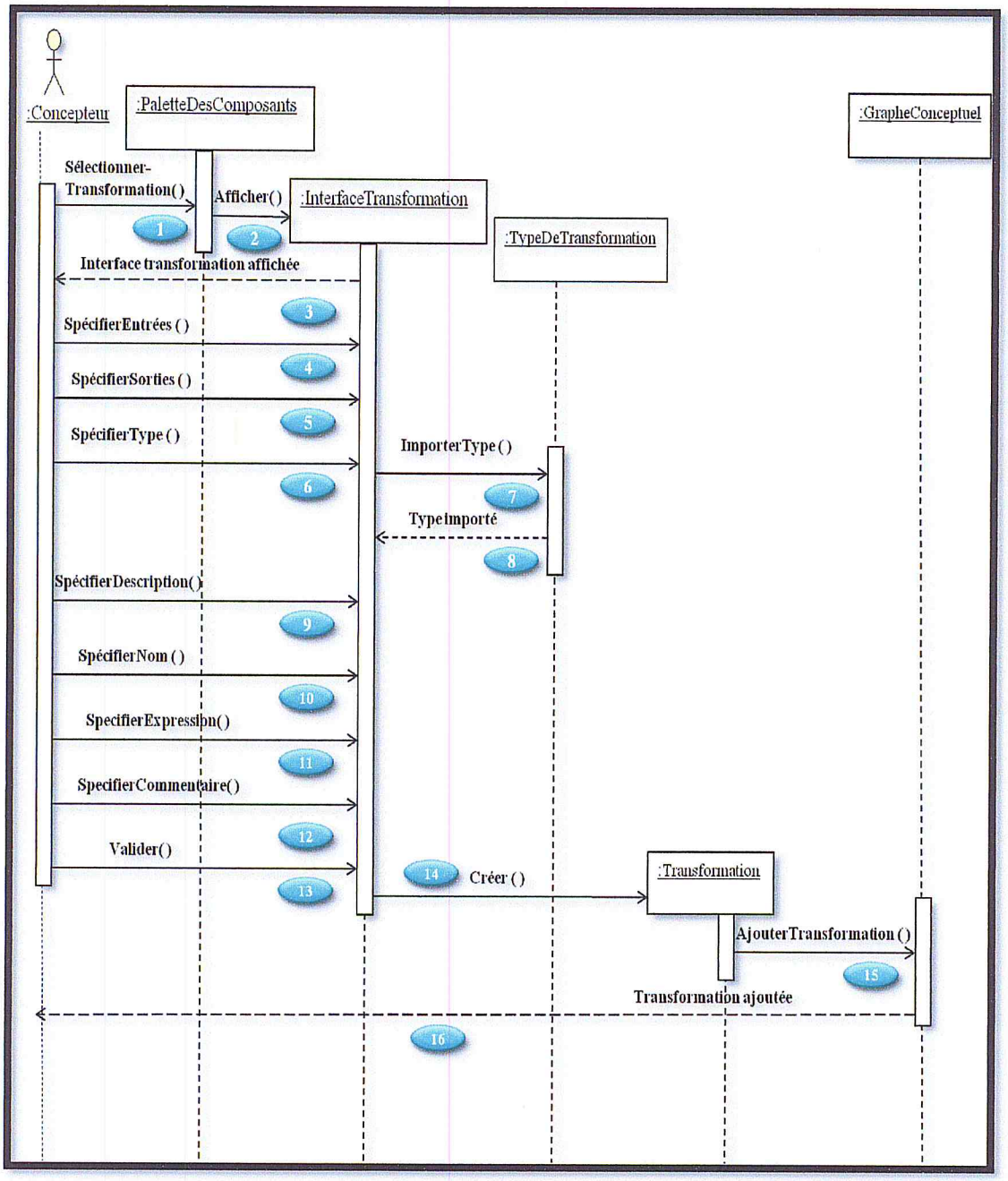


Figure 3: Diagramme de séquence « Créer une transformation ».

4.2.5. Diagramme de séquence « Ajouter une relation »

Ce scénario décrit les interactions entre objets lors de l'ajout d'une nouvelle relation entre deux coomposants.

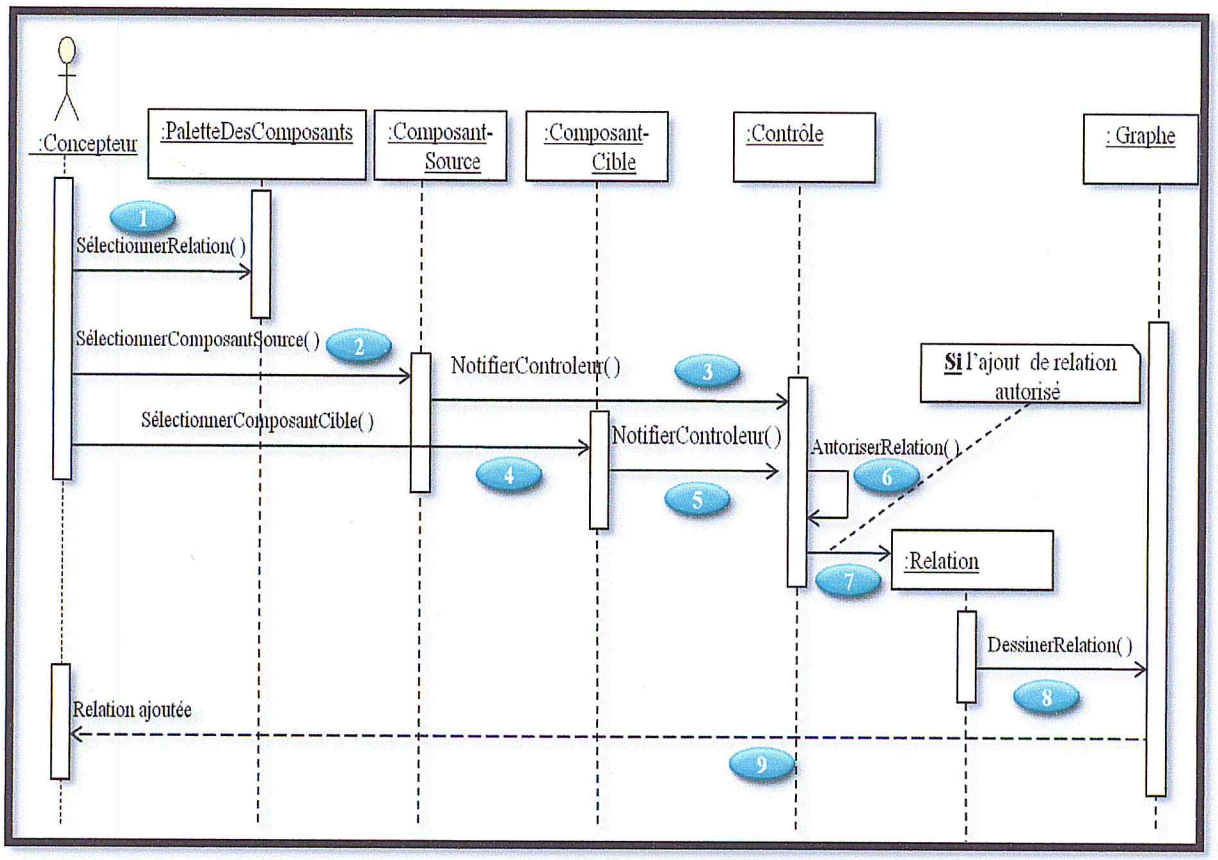


Figure 4: Diagramme de séquence « Ajouter une relation ».

4.2.6. Diagramme de séquence «Supprimer un composant »

Ce scénario montre la séquence d'opérations et d'interaction entre objets dans le cas de suppression d'un composant.

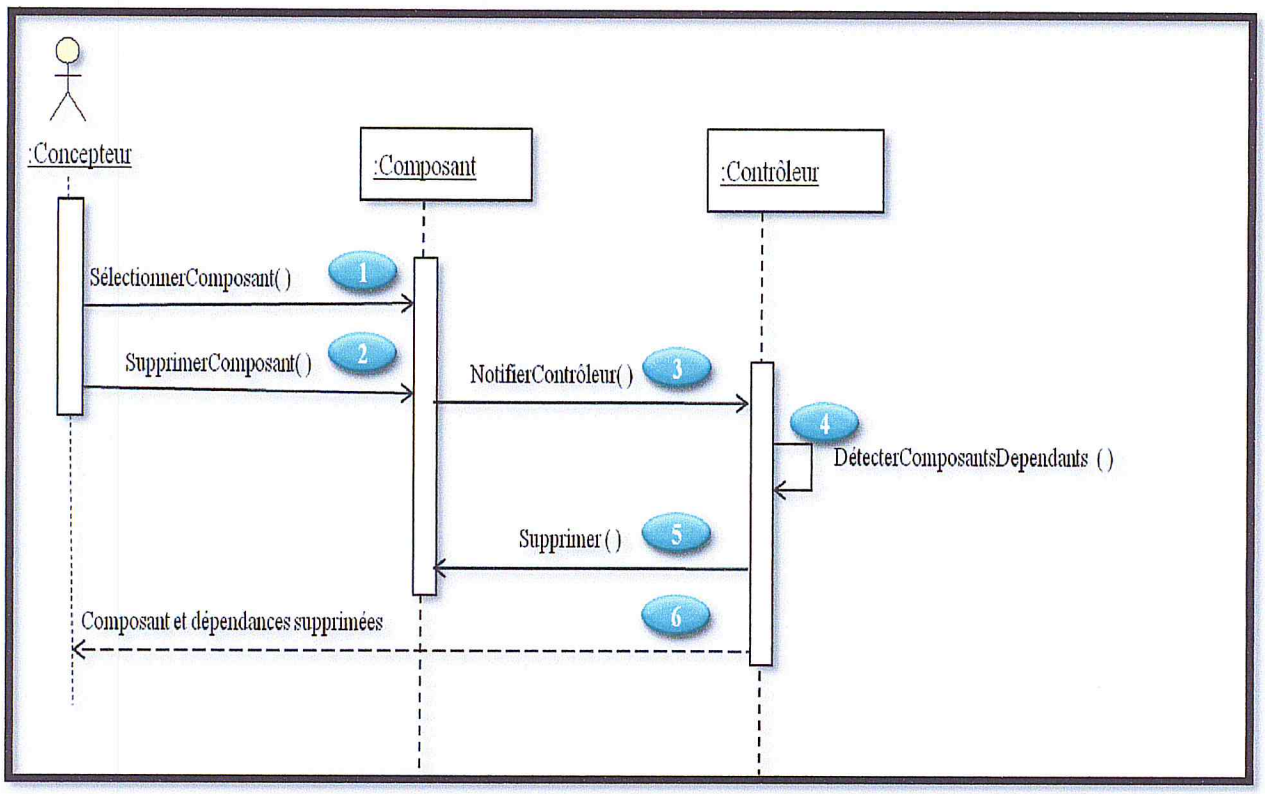


Figure 5: Diagramme de séquence « Supprimer un composant ».

5. CONCLUSION

L'objectif de notre travail est la réalisation d'un outil de modélisation du processus ETL selon le formalisme et le méta modèle de VASSILIADIS et al. Dans ce chapitre nous avons tenté de modéliser tous les aspects (fonctionnel, statique et dynamique) du futur système, afin de pouvoir entamer la partie implémentation et réalisation avec une bonne base, ce qui nous évitera les problèmes pouvant surgir lors de l'implémentation.

Partie III

CHAPITRE VI

IMPLEMENTATION ET VALIDATION

1. INTRODUCTION

L'outil développé dans le cadre de ce projet est nommé « Master-ETL-Designer ».

Dans ce chapitre nous allons définir d'abord les choix en termes d'environnement et outils de développement ainsi que l'architecture du système. En second lieu, nous procédons à la validation de notre outil avec des tests d'un certain nombre de fonctionnalités.

2. PRESENTATION DE CHOIX DE DEVELOPPEMENT

2.1. L'environnement de développement

2.1.1. Choix du langage de programmation (JAVA)

Java est un langage de programmation informatique orienté objet créé par SUN Microsystems. L'environnement Java peut être scindé en deux parties à savoir l'éditeur du programme Java et la machine virtuelle (JVM) qui va se charger de l'exécution du programme. C'est cette plateforme qui garantit la portabilité de Java. [BAP 07].

2.1.2. Eclipse

Eclipse est un environnement de développement intégré, il intègre les trois outils indispensables au développement : un éditeur de texte, un compilateur et un débogueur. Eclipse, développé à l'origine par IBM, a la particularité d'être un logiciel libre écrit en JAVA, on le trouve donc gratuitement sur toutes les plateformes (Windows, Linux,...). C'est aujourd'hui un outil largement utilisé dans le monde industriel [ECL 06]. Au cours du développement de notre application, on a choisi d'utiliser la version 3.6 (HELIOS).

2.2. Pattern MVC [OLI 11]

L'architecture Modèle/Vue/Contrôleur est une méthode de conception qui organise une interface IHM (Interface Homme-Machine) d'une application. Elle consiste à distinguer trois parties distinctes qui sont :

- modèle : données (accès et mise à jour)
- vue : interface utilisateur (entrées et sorties)
- contrôleur : gestion des événements et synchronisation

2.2.1. Rôle du modèle

Il décrit ou contient les données manipulées par l'application. Dans le cas typique d'une base de données le modèle offre des méthodes pour mettre à jour ces données (insertion, suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ces données. Les résultats renvoyés par le modèle sont dénués de toute présentation. C'est le modèle qui contient toute la logique métier de l'application.

2.2.2. Rôle de la vue

La vue correspond à l'interface avec laquelle l'utilisateur interagit. Sa première tâche est de présenter les résultats renvoyés par le modèle qui lui sont passés par le contrôleur. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur. Ces différents événements sont envoyés au contrôleur. La vue n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle.

2.2.3. Rôle du contrôleur

Le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle et ensuite avertit la vue que les données ont changé pour qu'elle se mette à jour.

2.2.4. Application du pattern MVC dans MASTER-ETL-DESIGNER

Nous avons remédié au pattern MVC pour la séparation entre les parties traitement, présentation et données.

- La partie vue s'occupe, d'une part, de la présentation globale de l'application (La palette des composants, visualisation du graphe…) et d'autre part, de la représentation graphique des différents composants (Concept, transformation, activité…).
- La partie modèle contient toutes les données sur les entités définies dans le méta modèle conceptuel et logique, ainsi que les méthodes de manipulation de ces dernières.
- La partie contrôle synchronise les vues et les modèles. Elle est notifiée à chaque interaction de l'utilisateur, contrôle cette action et renvoie le modèle correspondant à la vue.

  Par exemple : l'ajout d'une relation entre deux composants. Cet évènement est transmis au contrôleur correspondant. Le contrôleur vérifie l'autorisation de l'association entre les deux composants, appelle le modèle responsable de cette relation et l'envoie vers la vue correspondante en cas de permission d'ajout de cette relation.

2.3. Manipulation des graphes

2.3.1. Présentation de JGraph

JGraph est une bibliothèque open source de manipulation des graphes écrite en Java, et compatible avec la bibliothèque graphique Swing. Elle offre les principales fonctionnalités de création, visualisation et d'interaction avec les graphes. La bibliothèque JGraph est fondée sur la théorie mathématique des réseaux et la théorie des graphes.

Une application JGraph implémente le pattern MVC (Modèle-Vue-Contrôleur).

a. Le modèle du graphe (la classe GraphModel)

Le modèle contient les données sur le graphe ainsi que sur les différents composants, et fournit diverses méthodes pour accéder à ces données.

b. La vue du graphe (la classe GraphLayoutCache)

La ou les vues du graphe sont une ou plusieurs couches logiquement au-dessus du modèle qui effectuent la tâche de présenter visuellement les composants du graphe. Les vues sont automatiquement mises à jour lorsque les données du modèle changent.

2.3.2. Les composants d'un graphe

Tout composant dans un JGraph est considéré comme un objet de type GraphCell. Nous distinguons trois types d'objets

- Cellule (Vertex) : la donnée à afficher. Une cellule a une référence à un ensemble de ports qui lui sont attachés.
- Lien (Edge) : objet graphique qui relie deux cellules. Un lien est associé à deux ports (source, target)
- Port (Port) : objet graphique qui sert de point d'attache entre une cellule à un lien. Un port garde référence de la cellule 'parent' et des liens 'edges'.

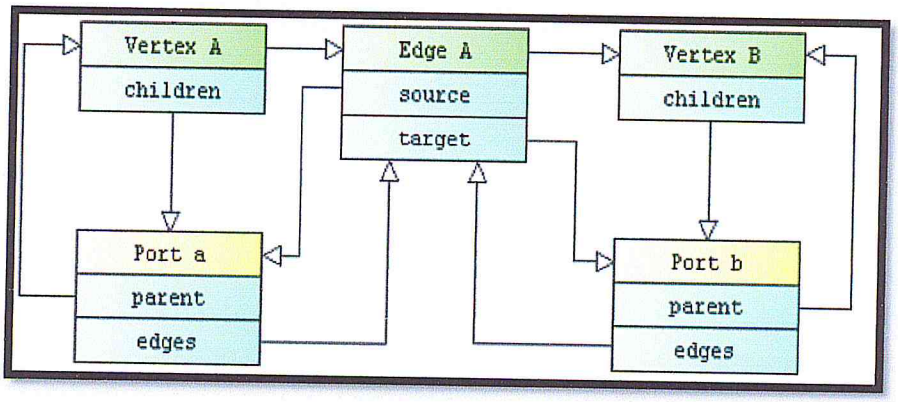


Figure 38: Représentation d'une association entre deux cellules [BEN 06].

2.3.3. Application de JGraph dans Master-ETL-Designer

La considération d'un processus ETL en tant que workflow implique la modélisation sous forme de graphe de la dynamique du processus.

La redéfinition de la classe GraphLayoutCache nous a permis de gérer les vues correspondant au méta modèle de VASSILIADIS (représentation graphique des concepts, activités, attributs, paramètres… etc.)

Le modèle de JGraph (GraphModel) contient les données sur chaque composant (les dimensions, la couleur, le libellé… etc.).

La classe JGraph regroupe tous les composants et permet de faire le lien entre un composant la vue qui lui est associée et les données sur le modèle.

Cependant, JGraph ne gère que l'aspect graphique d'une entité dans notre framework. L'aspect métier d'une entité est géré par la sous classe de la classe Entité correspondante (Diagrammes de classes « méta modèle conceptuel » et « méta modèle logique »). La figure -39- illustre la communication entre un composant JGraph et une entité de notre méta modèle.

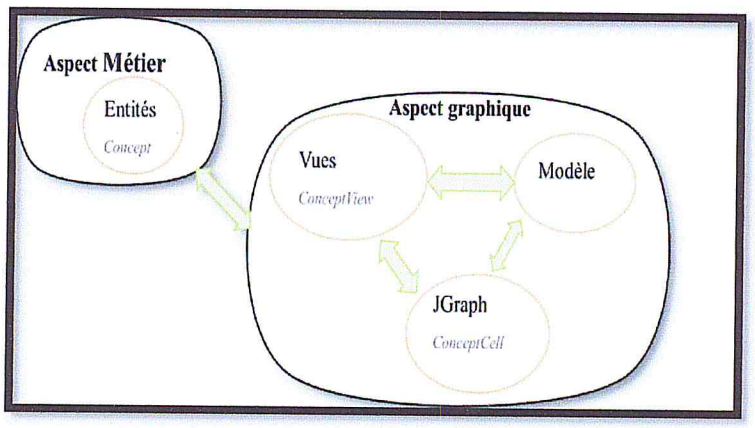


Figure 39: Gestion d'une entité avec JGraph.

2.4. Manipulation des fichiers XML

2.4.1. Présentation de XML

XML, eXtensible Markup Language, est le standard soutenu par le W3C pour le balisage de documents. Il définit une syntaxe générique utilisée pour formater des données avec des balises simples et compréhensibles par l'homme [RUS 01].

L'une des fonctionnalités primordiales de notre système est le stockage des projets réalisés sous format XML afin de pouvoir les consulter, modifier ou communiquer avec d'autres concepteurs. Les modèles réalisés par le concepteur, ainsi que tous les détails concernant le projet sont stockés dans des fichiers XML. Nous avons opté pour l'utilisation de ce modèle de données pour les grands avantages qu'il présente.

Le choix de XML comme format de stockage revient à la grande simplicité qu'il présente tant dans le développement des applications communicant avec XML, mais notamment dans la facilité de déploiement d'une telle application. D'autre part, la simplicité de XML ne limite pas son extensibilité du fait qu'il permet de représenter n'importe quel domaine d'application. En effet, XML est un langage structuré, extensible et simple d'utilisation.

2.4.2. Notions de bases de XML

Un document : Les documents XML sont des arbres. Un document XML se compose d'éléments complexes (en contenant d'autres) et d'éléments simples contenant du texte. Un arbre XML a un élément racine. On parle d'élément père et d'élément fils contenus dans l'élément père. Chaque élément à part la racine est associé à un élément père.

Un élément : Une unité particulière de données et son marqueur est appelée 'élément'. Les spécifications XML précisent la syntaxe exacte de ce marqueur : comment les éléments sont délimités par des balises, à quoi ressemble une balise, quel nom est acceptable pour un élément, où les attributs sont positionnés et ainsi de suite. [RUS 01].

Une DTD : Les marqueurs autorisés dans une application XML spécifique peuvent être documentés dans une définition de type de document (Document type Definition-DTD). La DTD liste tous les marqueurs autorisés et détermine où et comment un élément doit être inclus dans un document. Des instances particulières de document peuvent être comparées à la DTD. Les documents qui correspondent à la DTD sont dits 'valide'. [RUS 01]

2.4.3. Les outils de manipulation de documents XML

a. SAX

L'API SAX (Simple API for XML), une interface de programmation d'applications Java basée sur un modèle évènementiel et mise en œuvre par la majorité des parseurs XML. [RUS 01]. L'API SAX permet d'analyser un document XML basé sur le traitement séquentiel des données. Lorsque le parseur lit un document XML, il envoie en temps réel de l'information au programme. Chaque fois que le parseur rencontre une balise de début, une balise de fin, des données textuelles ou une instruction de traitement, il en informe le programme.

b. DOM

Document Object Model, une API basée sur une arborescence qui considère qu'un document XML est un ensemble d'objets imbriqués munis de diverses propriétés. [RUS 01]

C'est une spécification du W3C pour proposer une API qui permet de modéliser, de parcourir et de manipuler un document XML. Le principal rôle de DOM est de fournir une représentation mémoire d'un document XML sous la forme d'un arbre d'objets et d'en permettre la manipulation (parcours, recherche et mise à jour). [CYN 05]

c. JDOM

JDOM est une API spécifique pour Java qui utilise DOM pour manipuler les éléments d'un Document Object Model spécifique (créé grâce à un parseur basé sur SAX).

2.4.4. Application de XML dans Master-ETL-Designer

La restitution des détails d'un projet réalisé avec Master-ETL-Designer nécessite de sauvegarder les données concernant les modèles conçus (stocké dans 'Exemple_Modèle.xml') ainsi que les détails sur le projet (stocké dans 'Exemple_Projet.xml'). Les DTD constituent la grammaire permettant de valider un fichier XML. Nous avons proposé deux DTD spécifiques à nos besoins. Nous avons proposé aussi d'autres DTDs pour les bibliothèques des modèles manipulés par l'application (Types de transformation, types d'activités…), qui correspondent aussi à la spécification de la bibliothèque qu'on a défini dans le diagramme de classes 'Modèle de Master-ETL-Designer'. Le corps de ces DTDs est exposé en annexe.

3. ARCHITECTURE DU SYSTEME

Master-ETL-Designer est un framework destiné à la modélisation d'un workflow de type ETL. Un framework est un kit de composants logiciels structurels, qui servent à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel. En effet, notre framework permet à un concepteur la modélisation d'un processus ETL sur différents niveaux (conceptuel et logique), ainsi que le mappage d'un modèle conceptuel vers le modèle logique correspondant, tout en restant dans le même environnement intégré.

3.1. L'architecture de Master-ETL-Designer

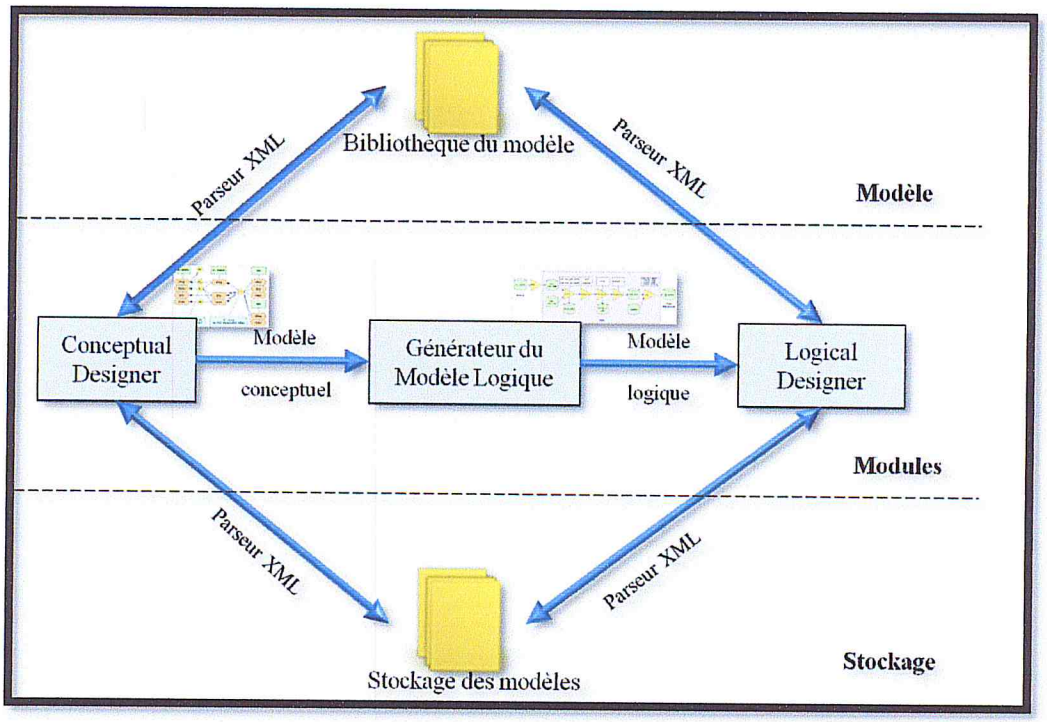


Figure 40: Architecture de Master-ETL-Designer.

3.1.1. Les principaux modules

Le rôle principal de notre outil est la modélisation d'un processus ETL sur le niveau conceptuel et le niveau logique. Ces deux fonctionnalités sont implémentées par les modules « Conceptual Designer » et « Logical Designer ». Ces deux modules disposent des différents éléments nécessaires pour la modélisation du processus (palette de composants, zone de modélisation, explorateur de composants). Elles permettent ainsi de concevoir un modèle conceptuel ou logique du processus ETL. Le module « Générateur du Modèle Logique» s'occupe du mappage du modèle conceptuel vers un modèle logique (certaines spécificités de ce module seront détaillées un peu plus loin dans ce chapitre.

3.1.2. Bibliothèque des modèles

Les trois principaux modules prennent en charge la modélisation en communiquant avec une bibliothèque des modèles. Cette bibliothèque contient des instances des

constructeurs les plus utilisés (des instances de transformation, des activités…). Ces instances sont stockées dans des fichiers XML, et sont utilisées par les modules « Conceptual Designer » et « Logical Designer » afin de spécifier les entités instanciées. D'autre part, le module « Générateur du Modèle Logique » utilise cette bibliothèque pour définir l'entité logique correspondante (par exemple : la spécification de la fonction implémentée par une activité à partir du type de fonction de la transformation mappée).

La communication des différents modules avec une bibliothèque indépendante du système permet d'implémenter le mécanisme d'extensibilité proposé par les auteurs du méta modèle utilisé. Ainsi, nous apportons à l'utilisateur la possibilité de personnalisation de l'environnement en fonction de ses besoins spécifiques.

3.1.3. Stockage

Le stockage est une fonctionnalité primordiale dans notre outil, du fait qu'elle permet la réutilisation des modèles. L'outil implémente des parseurs XML qui permettent de générer le modèle XML correspondant au modèle conceptuel et logique, et d'effectuer la fonction inverse (génération du modèle conceptuel et logique correspondant au modèle XML).

3.2. Le mappage du modèle conceptuel vers le modèle logique

Le module de mappage a pour charge de générer pour chaque composant du modèle conceptuel le composant correspondant dans le modèle logique. On a dressé un tableau de mappage dans le chapitre IV. Cependant, il existe certains détails sensibles à examiner séparément.

3.2.1. Les transformations

Une transformation du modèle conceptuel est mappée vers une activité dans le modèle logique. Le mappage semi-automatique des transformations implique trois considérations :

- La spécification des propriétés d'une activité (le nom, les schémas -d'entrée, de sorite, de rejets-, la sémantique…).
- La composition en série de transformations (Serial composition).
- La définition de l'ordre d'exécution des activités dans le modèle logique.

Lorsqu'on s'intéresse au mappage d'une transformation, on distingue entre deux catégories.

Les filtres : conservent le même schéma des données en entrée et en sortie ; (ex : filtre, nettoyage)

Les transformateurs : Changent le schéma des données en ajoutant ou supprimant des attributs ; (ex : agrégation).

Une activité peut avoir un schéma d'entrée (les attributs d'entrée), un schéma de sortie (les attributs de sortie), un schéma de fonctionnalité (les paramètres de l'activité), un schéma de rejets (les attributs qui figurent en entrée mais pas en sortie de l'activité) et un schéma généré (les nouveaux attributs générés par l'activité).

a. Mappage d'un filtre

Considérant la transformation (filtre) 'T', illustrée dans la figure -41-. Le mappage de 'T' vers une entité du modèle logique résultera en une activité A.

- Le schéma du concept source 'S1' et du concept de destination 'S2' formeront respectivement le schéma d'entrée et le schéma de sortie de l'activité 'A'.
- Une relation 'Provider' est générée entre chaque attribut dans le concept source 'S1'-respectivement le concept de destination 'S2'-, et l'attribut correspondant dans le schéma d'entrée –respectivement le schéma de sortie- de 'A'.
- Les attributs du concept 'S1' impliqués dans la transformation formeront le schéma !fonctionnel de l'activité 'A'.
- Une relation 'Provider' est générée entre chaque attribut dans le schéma d'entrée et un attribut dans le schéma de sortie de l'activité, tant que ces attributs ne font pas partie du schéma fonctionnel.

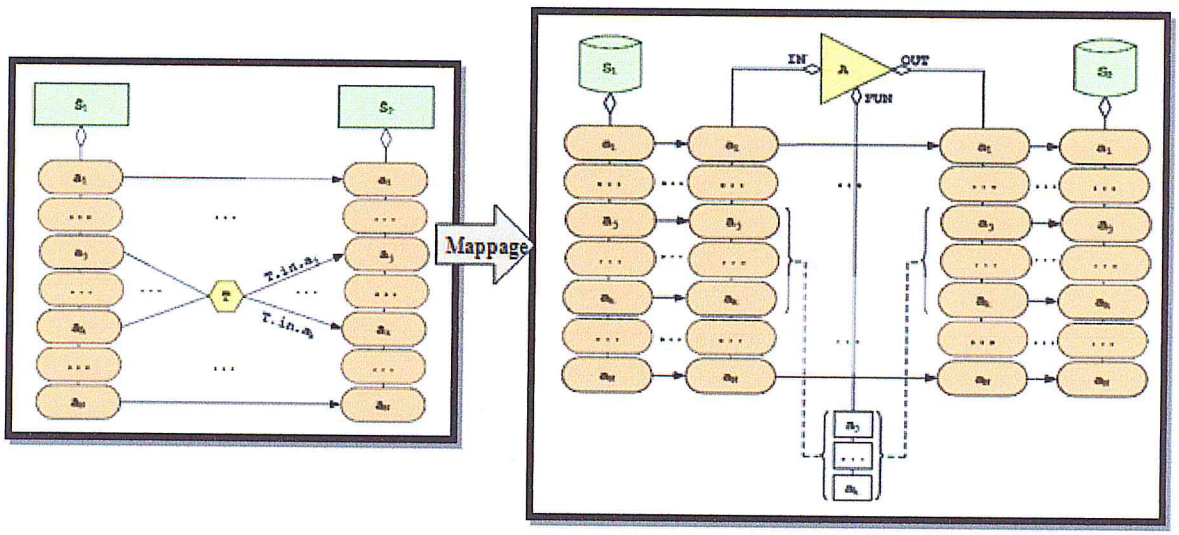


Figure 41: Mappage d'une transformation T vers une activité A (cas des filtres).

b. Mappage d'un transformateur

Ce type de transformations diffère des filtres dans la mesure où les transformateurs modifient les schémas de l'activité. Les schémas générés et projetés ne peuvent être vide à la fois. En effet, un transformateur modifie le schéma en sortie, soit en ajoutant des nouveaux attributs (schéma généré), soit en supprimant des attributs (schéma des rejets). Considérant le transformateur 'T' illustré dans la figure -42- .

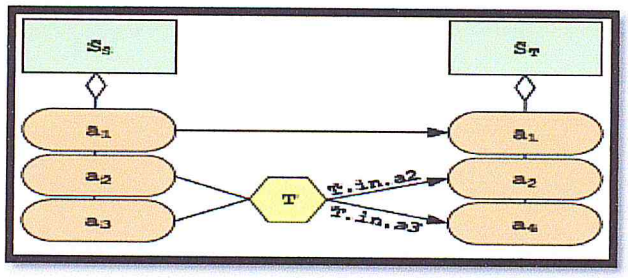


Figure 42: Exemple de population d'un concept ST via un transformateur.

L'activité générée après le mappage de la transformation 'T' définit les schémas suivants :

Schéma d'entrée	A.in = S_S.out = {a1, a2, a3}
Schéma de sortie	A.out = S_T.in = {a1, a2, a4}
Schéma de fonctionnalité	A.fun = T.in = {a2, a3}
Schéma généré	A.gen = T.out-T.in = {a2, a4}-{a2, a3} = {a4}
Schéma rejeté	A.pro = T.in-T.out = {a2, a3}-{a2, a4} = {a3}

3.2.2. Les compositions en série

Considérant deux transformations 'T_1', 'T_2' qui sont liées par une relation de composition en série. La représentation logique correspondante comprend une séquence de deux activités 'A_1', 'A_2', et le schéma d'entrée de 'A_2' est le schéma de sortie de 'A_1'.

$A_2.in = A_1.out$

3.2.3. Contrainte ETL

La particularité d'une contrainte ETL est qu'elle ne contient pas un schéma de sortie. Considérant La contrainte ETL 'PK' illustrée dans la figure -43-. Les schémas de l'activité correspondante sont comme suit :

Schéma d'entrée	PK.in = S_S.out = {a1, a2, a3}
Schéma de sortie	A.out = S_T.in = {a1, a2, a4}
Schéma de fonctionnalité	A.fun = T.in = {a2, a3}
Schéma généré	A.gen = T.out-T.in = {a2, a4}-{a2, a3} = {a4}
Schéma rejeté	A.pro = T.in-T.out = {a2, a3}-{a2, a4} = {a3}

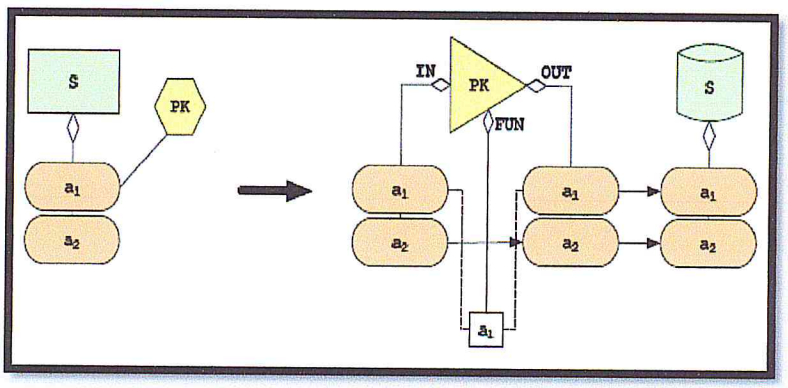


Figure 43: Mappage d'une contrainte ETL vers une activité.

3.2.4. Mappage d'une note

Une note contient la sémantique de la transformation annotée. En effet, une note dans le modèle conceptuel a une structure bien particulière. Une note peut contenir comme information :

- Une description textuelle : qui commence par 't::'.
- Le nom du type de fonction : qui commence par 'f ::'.
- L'expression de la transformation : qui commence par 'e::'.

A partir des informations contenues dans la note, on définit les propriétés de l'activité correspondante à la transformation.

3.2.5. Ordre d'exécution des transformations

Dans le niveau logique, on s'intéresse à la séquence d'exécution des activités. La détermination de l'ordre d'exécution des activités prend en considération :

- L'ordre des transformations dans une relation de composition en série (Serial composition). Les activités générées gardent le même ordre que les transformations.
- Les transformations qui ont le même schéma d'entrée et le même schéma de sortie, sont d'ordres équivalents.
- Les contraintes ETL ont une plus grande priorité.

4. PRESENTATION DEL'APPLICATION

Dans cette partie nous présenteront les plus importantes fonctionnalités de notre outil afin de tester si les objectifs fixés lors la description du méta modèle de VASSILIADIS et lors de la définition des besoins sont atteints.

4.1. Présentation de la fenêtre principale

La fenêtre principale de Master-ETL-Designer comporte les cinq composants essentiels :

(1) La palette des composants ; contient tous les composants qui vont être utilisées dans la modélisation d'un processus ETL.
(2) L'espace de modélisation ; c'est la zone de dessin où le concepteur positionne les composants du processus.
(3) L'explorateur des composants ; ce composant affiche tous les composants crées par le concepteur dans son projet de modélisation.

(4) La barre de menu ; comporte les quatre fonctionnalités nécessaires dans la modélisation d'un processus ETL:

- Fichier : Englobe tous les fonctionnalités relies a la manipulation d'un fichier (ouvrir, enregistrer, fermer…etc.).
- Edition : Présente les options d'édition offertes à l'utilisateur (Annuler, copier, coller…etc.).
- Project : Présente les options reliées à la gestion du projet (valider le modèle, propriétés du projet, suppression du projet).
- Types : Présente tous les types des données utilisables dans le processus, on trouve les types d'activités, de données et de fonction.

(5) La barre d'outils ; comporte les boutons suivants :

- (6) Nouveau projet ; permet de créer un nouveau projet de modélisation
- (7) Ouvrir projet ; permet d'ouvrir un projet qui existe déjà.
- (8) Enregistrer : permet d'enregistrer un projet.
- (9) Les deux boutons Annuler et Refaire ; permettent respectivement d'annuler ou de refaire une action lors la modélisation d'un processus ETL.
- (10) Copier ; permet de copier les composants dans l'espace de modélisation.
- (11) Coller ; permet de coller les composants dans l'espace de modélisation.
- (12) Supprimer ; supprimer un composant de l'espace de modélisation.
- (13) Valider le modèle ; permet de générer un modèle logique à partir du modèle conceptuel.
- (14) Les trois fonctions de zoom ; comporte le zoom en avant, zoom par défaut, zoom en arrière.
- (15) Imprimer ; pour imprimer le processus modélisé.
- (16) Aide ; donner à l'utilisateur des renseignements sur l'outil.
- (17) Pointeur ; pour sélectionner le pointeur.

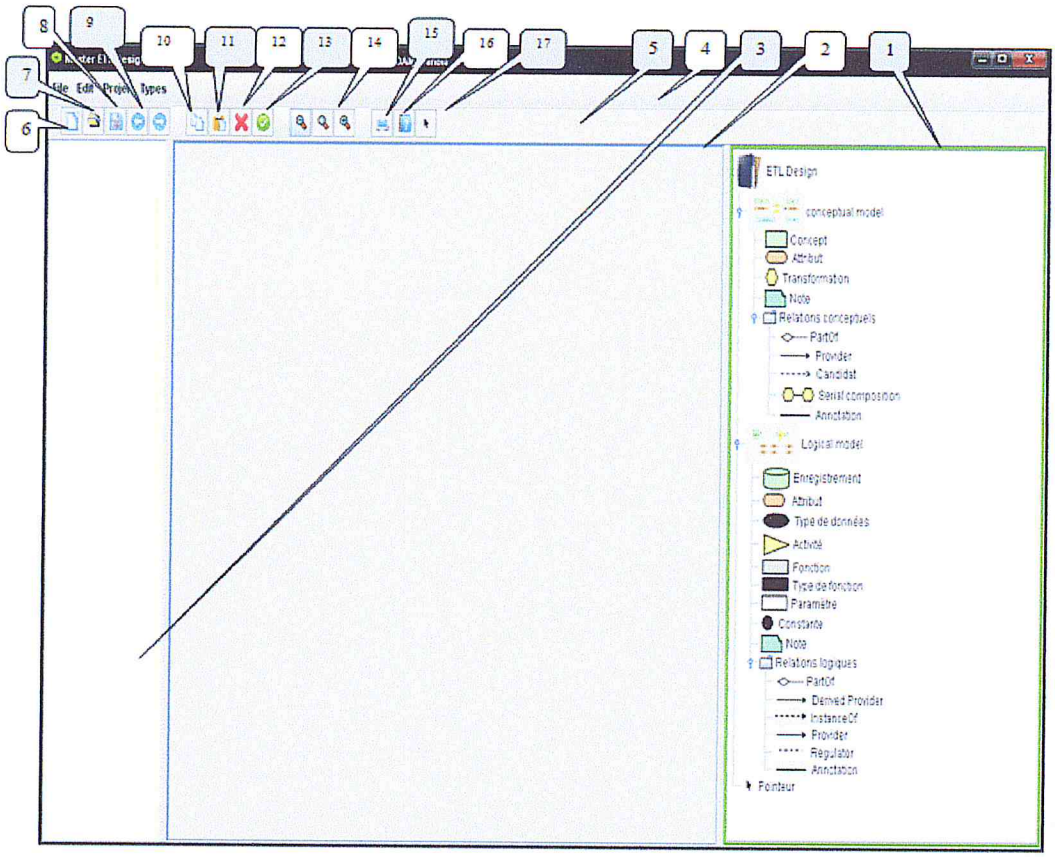


Figure 44: La fenêtre principale.

4.2. Présentation de l'interface « Nouveau projet »

Le clique sur le bouton 'Nouveau projet' (*numéroté (6) dans la figure -44-*) génère l'interface illustrée dans la figure -45- .

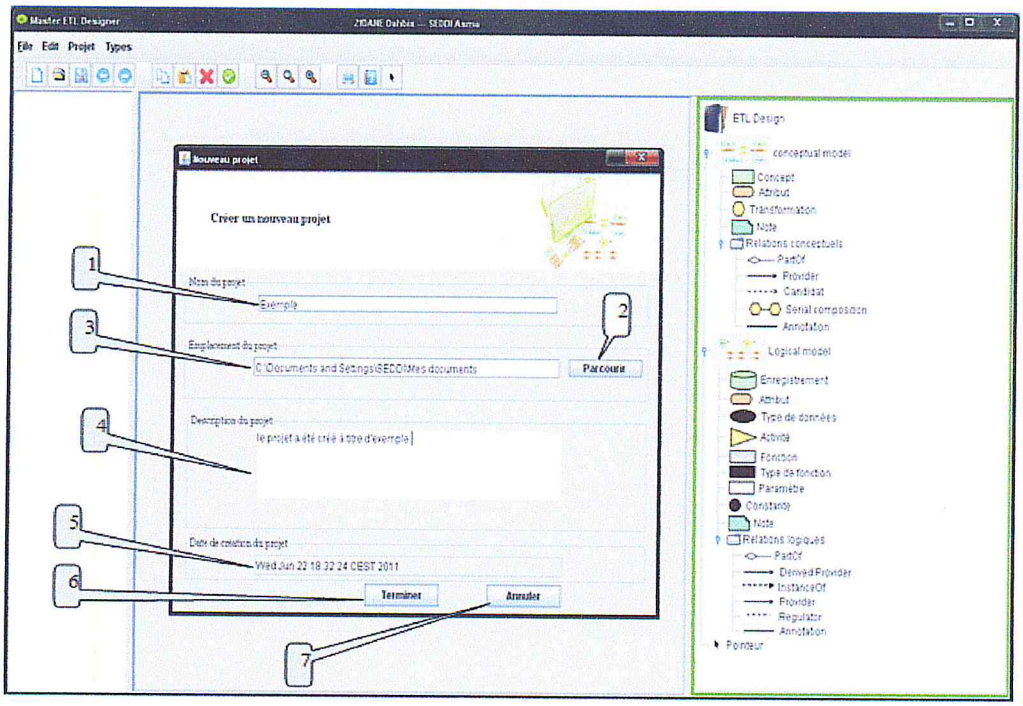


Figure 45: L'interface « Nouveau projet ».

(1) Nom du projet ; c'est une zone de texte qui permet de saisir le nom choisi par le concepteur pour son nouveau projet.

(2) Parcourir ; ce bouton permet de choisir l'emplacement de création du projet.

(3) Emplacement du projet ; zone de texte qui affiche le chemin de l'emplacement du projet.

(4) Description du projet ; c'est une zone de texte où le concepteur peut affecter une description au projet.

(5) Date de création du projet ; contient la date de création du projet générée automatiquement par le système.

(6) Terminer ; ce bouton achève la création d'un nouveau projet.

(7) Annuler ; le concepteur peut annuler la création du projet en cliquant sur ce bouton.

4.3. Présentation de l'interface « Ajouter un composant»

Après la création d'un nouveau projet en cliquant sur le bouton Terminer (*numéroté (6) dans la figure -44-*), l'espace de modélisation est prêt à être utilisé par le concepteur.

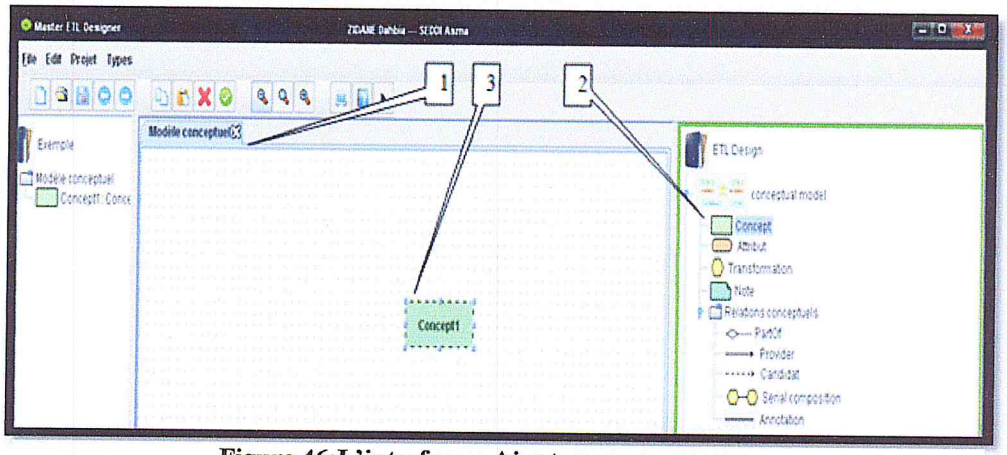


Figure 46:L'interface « Ajouter un composant ».

Le premier objet produit dès la création d'un projet est l'onglet de modélisation conceptuelle.

(1) L'onglet de modélisation conceptuelle ; correspond à l'espace de modélisation au niveau conceptuel.

L'opération « Ajouter un composant » dans l'espace de modélisation se passe comme suit :

-La sélection d'un composant à partir la palette des composants *(comme illustré dans la figure -44- numéro(1))*,

-Positionner le composant en cliquant sur l'emplacement choisi dans l'espace de modélisation *(figure -44- ; numéro (2))*.

Dans l'exemple de la figure -46-, on a ajouté un composant «Concept » nommé : Concept1.

4.4. Présentation de l'interface « Ajouter un attribut »

L'ajout d'un attribut se produit de la même manière que l'ajout d'un composant, sauf que l'ajout d'un attribut exige de spécifier le concept auquel il est attaché.

Comme illustré dans la figure -47-, on a :

(1) Choisir un concept ; c'est l'interface qui s'affiche dès qu'on positionne l'attribut dans l'espace de modélisation.
(2) Liste des concepts ; cette liste propose tous les concepts disponibles dans l'espace de modélisation.

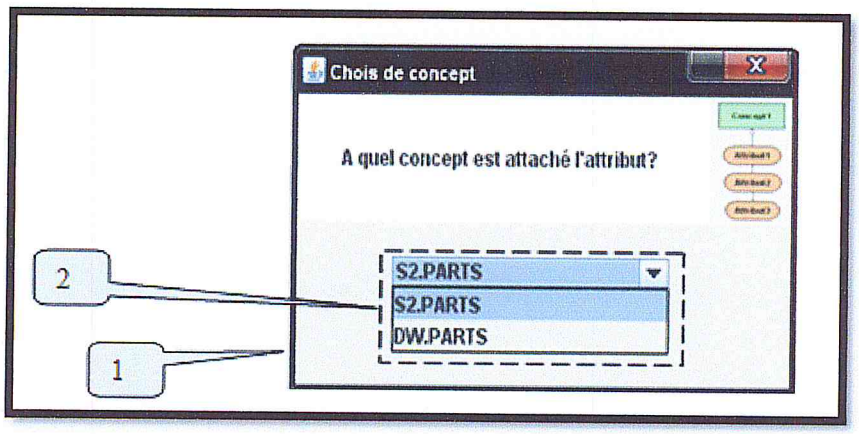


Figure 47:L'interface « Ajouter un attribut».

4.5. Présentation de l'interface « Ajouter une transformation »

L'ajout d'une transformation affiche la fenêtre présentée dans la figure -48-. La fenêtre d'une transformation *(numéro (1))* comporte :

(4) Les composants disponibles ; représente la liste des composants disponibles dans l'espace de modélisation et qui peuvent participer dans l'entrée ou bien la sortie d'une transformation.

(2) Les entrées ; représente la liste des composants qui sont choisis afin de participer dans les entrées d'une transformation.

(6) Les sorties ; représente la liste des composants qui sont choisis afin de participer dans la sortie d'une transformation.

(3)/(5) Les deux boutons '<' et '>' représentent respectivement les boutons de sélection et de rejection des composants à la liste des entres ou des sorties.

(7) Nom ; c'est une zone de texte pour spécifier le nom de la transformation.

(8) Commentaires ; c'est une zone consacrée pour les commentaires associés à la transformation.

(9) Description ; contient une brève description sur le type de la transformation.

(10) Type ; une liste de choix permettant de sélectionner le type de transformation.

(11) Expression ; le concepteur décrit l'expression utilisée dans la transformation.

(12) OK ; un bouton pour finaliser une transformation.

(13) Annuler ; le concepteur peut annuler l'ajout d'une transformation par un clique sur le bouton Annuler.

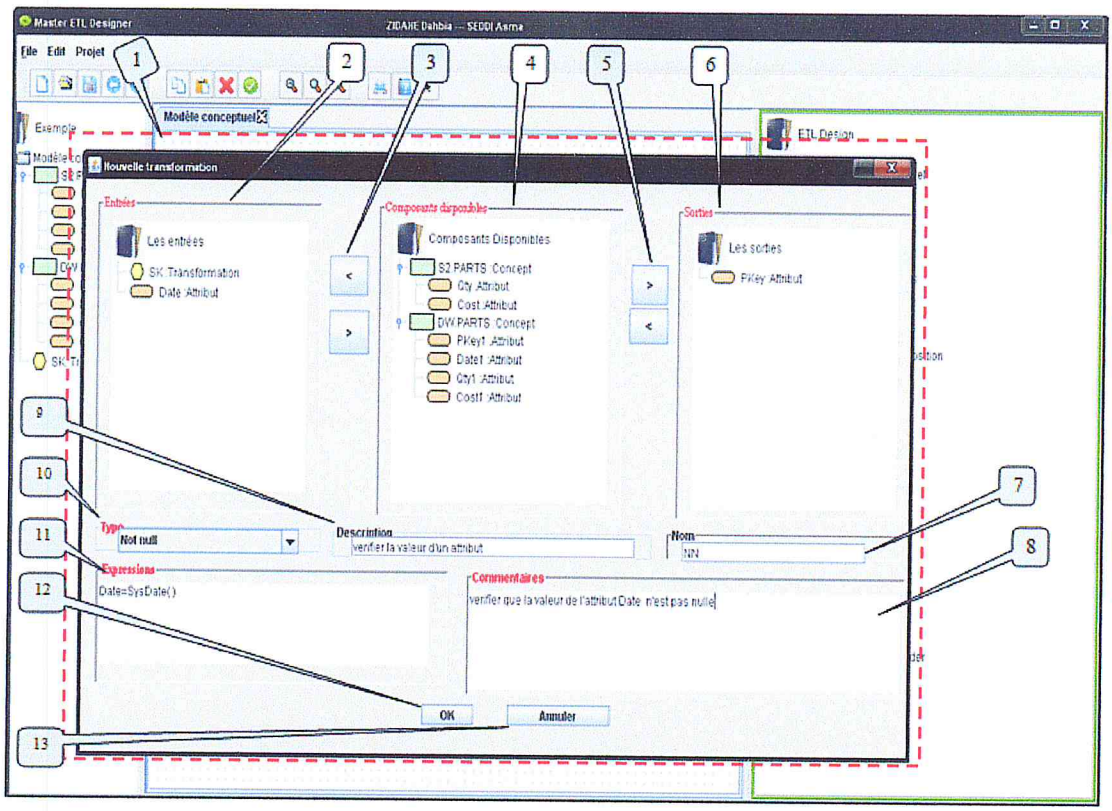


Figure 48: L'interface « Ajouter une transformation».

4.6. Présentation de l'interface « Exemple d'un processus ETL»

L'interface présentée dans la figure -49- représente un exemple d'un processus ETL.

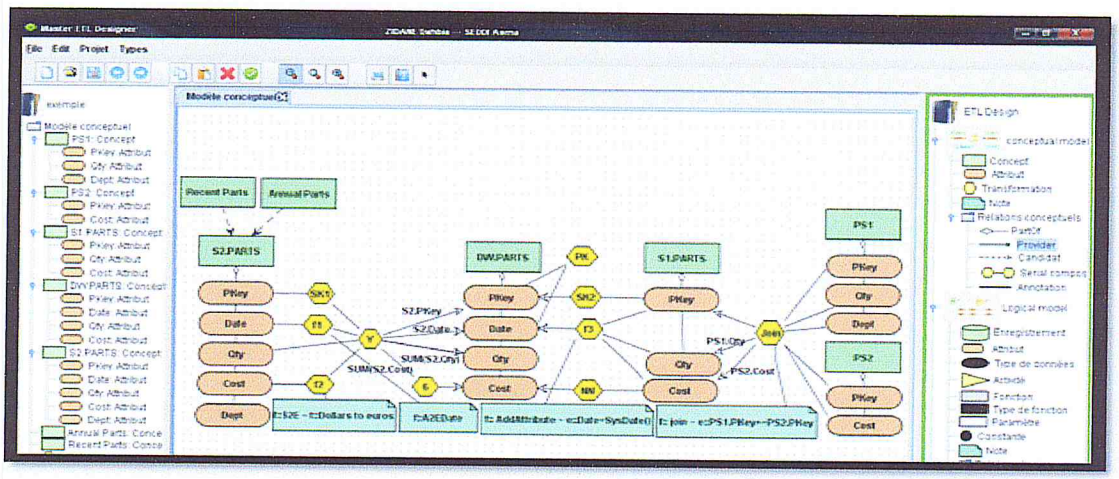


Figure 49: Exemple d'un processus ETL.

4.7.Le mappage du modèle conceptuel vers le modèle logique

Chaque modèle conceptuel devra être validé pour qu'il soit mappé en modèle logique en appuyant sur le bouton 'Valider' (*numéroté (13) dans la figure -44-*).

4.7.1. Présentation de l'interface «Assignation du type de données pour les attributs »

Cette interface permet de spécifier le type de données pour tous les attributs.

(1) Liste des DataStores ; représente la liste des enregistrements qui seront disponibles dans l'espace de modélisation logique.

(2) Type des données ; cette liste est faite pour attribuer à chaque attribut d'un enregistrement un type de données en cliquant sur le bouton (3).

(4) Valider ; le concepteur termine l'opération de mappage en cliquant sur le bouton 'Valider'.

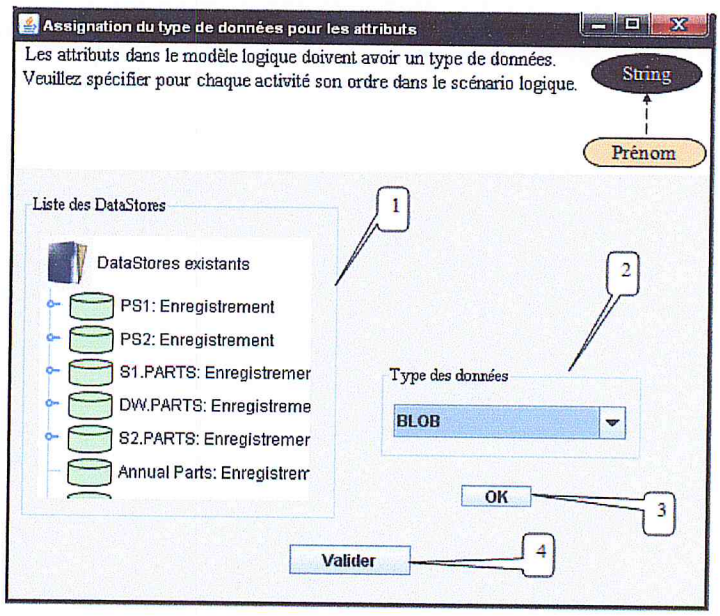


Figure 50: Assignation du type de données pour les attributs.

IV.7.2. Exemple de mappage

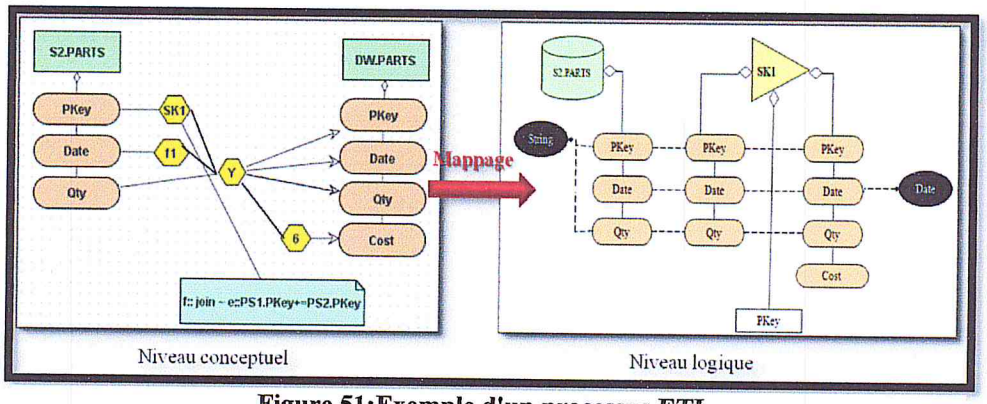


Figure 51:Exemple d'un processus ETL.

La figure -51- contient des captures de deux interfaces (à gauche : une partie d'un processus conceptuel, à droite : une partie du modèle logique généré à partir de ce processus). Une fois le modèle logique généré, le concepteur pourra l'enrichir de la même manière que pour le modèle conceptuel. En effet, la palette des composants contient aussi tous les composants nécessaires pour la modélisation d'un processus ETL au niveau logique.

Deux objectifs principaux nous ont été de guide dans l'implémentation des interfaces de notre outil. D'une part, la proposition d'un outil respectant le méta modèle de VASSILIADIS et al., et d'une autre, la présentation d'interface conviviale, simple d'utilisation, et guidant le concepteur dans son projet de modélisation.

5. CONCLUSION

L'implémentation et la validation font partie fondamentale du cycle de vie d'un produit logiciel, du fait qu'elles permettent la mise en œuvre et la validation des besoins définis dans les premières phases du projet. Au cours de ce chapitre, nous avons présenté les fonctionnalités de l'outil Master-ETL-Designer, et on a montré sa validation aux besoins de modélisation d'un processus ETL et notamment avec le formalisme et le méta modèle de VASSILIADIS et al. Nous avons aussi bien donné importance à la conformité au méta modèle proposé, qu'aux aspects ergonomie, convivialité et assistance dans la conception afin de fournir un environnement de conception familial et facile d'utilisation.

CONCLUSION GENERALE

CONCLUSION

Les systèmes ETL (Extraction-Transformation-Loading) sont une catégorie d'outils ayant pour tâche l'homogénéisation des données sources, leurs nettoyage, filtrage, … et enfin leur chargement dans l'entrepôt de données.

Nous avons découvert au cours de ce travail l'importance et l'impact du processus ETL dans la réussite d'un projet d'entreposage, sa complexité en termes de flux de données et synchronisation d'un nombre important de tâches. Pour pouvoir maîtriser toute cette complexité, les plans de gestion du processus sont d'une importance capitale.

La problématique de la modélisation des processus ETL reste d'actualité et le nombre de contributions devant l'importance et la complexité du processus reste insuffisant.

L'état de l'art nous a permis d'aboutir au choix d'une approche de modélisation du processus ETL. Le modèle de Vassiliadis a été retenu pour la mise en œuvre du Framework Master-ETL-Designer. Ce modèle avec toute sa richesse et sa puissance occupe une place très importante et particulière dans ce mémoire.

Ainsi, nous l'avons étudié et découvert tous les concepts sur les niveaux conceptuel et logique. Nous avons fait quelques recommandations sur le métamodèle conceptuel pour le rendre plus fin et plus clair. Au niveau logique, nous avons proposé notre propre métamodèle en se basant sur les concepts et principes caractérisant cette couche.

Enfin, nous avons mis en œuvre un prototype qui implémente toute la logique présentée par les métamodèles afin de permettre à un concepteur de faire le design d'un processus ETL quelconque avec toute la souplesse et la convivialité que mérite cette tâche.

Au terme de ce travail, nous considérons avoir maîtrisé les grands aspects et principes des systèmes décisionnels mais de manière plus profonde les processus

ETL. Néanmoins, comme tout projet, celui-ci mérite des améliorations et une continuité que nous pourrons résumer comme suit :

PERSPECTIVES

- En termes de méta modélisation, ce travail mérite l'intégration d'un méta modèle pour décrire les ressources, l'environnement et les détails d'exécution du processus au niveau physique,

- Quant au prototype développé, il s'agit de trouver des solutions pour minimiser au maximum l'effort du concepteur dans le processus de mappage semi-automatique en implémentant des algorithmes d'ordonnancement des transformations (toujours est-il que certains aspects dans le mappage nécessitent toujours l'intervention du concepteur).

- Enfin, une des perspectives les plus importantes est celle qui consiste à générer le modèle du processus ETL au niveau physique en spécifiant les connexions vers les sources et l'entrepôt cible, ce qui permettra au système de générer un processus exécutable qui assure réellement l'entreposage.

BIBLIOGRAPHIE

Ouvrages

[RAH 00] RAHM. E., HAI DO. H., "Data Cleaning: Problems and Current Approaches". Bulletin of the Technical Committee on Data Engeneering. P:3-13. December 2000.

[GOM 05] Giraldo Gomez GL, "Construction automatisée de l'ontologie de systèmes médiateurs". Thèse de doctorat. 2005.

[INM 92] INMON.WH., "Building the data warehouse.". Wiley. 1992.

[KIM04] KIMBALL.R., CASERTA.J., "The data warehouse ETL toolkit". Wiley. 2004.

[KHO 09] KHOURI. S., "Modélisation conceptuelle à base ontologique d'un entrepôt de données.", Thèse de Magistère. 2009.

[MOR 05] L.MORA S., "Data Warehouse design with UML". Thèse de doctorat. 2005.

[PON 01] PONNIAH P., "Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals". Wiley, 2001.

[ROM 07] ROMERO, O. et A. "Abelló. Automating Multidimensional Design from Ontologies". DOLAP 07 P:1-8. 2007.

[RUS 01] RUSTY HAROLD.E., MEANS.S.W. , "XML in a nutshell.". O'REILLY. 2001.

[SIM 04] SIMITSIS A., "Modeling and Optimization of Extraction-Transformation-Loading (ETL) Processes in Data Warehouse Environments.". Thèse de doctorat. 2004.

[SIM 08] SIMITSIS A., VASSILIADIS P., "A method for the mapping of conceptual designs to logical blueprints for ETL processes.". Decision Support Systems 45 (2008). P: 22-40.

[STR 02] STRANGE.K., "ETL Was the Key to this Data Warehouse's Success.". Technical Report CS-15-3143, Gartner, 2002.

[VAS 02] VASSILIADIS P., SIMITSIS A, SKIADOPOULOS S., "Conceptual modeling for ETL Processes.". DOLAP 02. 2002.

[VAS 05] VASSILIADIS P., SIMITSIS A., GEORGANTAS P., TERROVITIS M, SKIADOPOULOS S., "A generic and customizable framework for the design of ETL scenarios.". Information Systems 30 (2005), P:492-525. 2005.

[VAS 09] VASSILIADIS P., "A Survey of Extract–Transform–Load Technology.". International Journal of Data Warehousing and Mining. 2009.

[TES 00] TESTE.O., "Modélisation et manipulation d'entrepôts de données complexes et historisées" , Thèse de doctorat, 2000.

Webographie

[AUD 06] AUDIBERT.L., http://laurent-audibert.developpez.com/Cours UML/ html/Cours-UML001.html. 31/10/2006

[BAP 07] BAPTISTE W., "Bien débuter en Java". http://www.developpez.net. 3/08/2007.

[BEN 06] BENSON D., "JGraph and JGraph Layout Pro User Manual.". Http: //www.jgraph.com. 21/12/2006.

[CYN 05] CYNOBER N., "Manipuler des données XML avec Java et JDOM". www.developpez.com. 2005.

[DEM] DEMAREST M., "The politics of data warehousing.". http:// www.hevanet.com/demarest/marc/dwpol.html.

[ECL 06] http://www.eclipsetotale.com. 10/10/2006.

[INM 97] INMON B., "The Data Warehouse Budget. DM Review Magazine.". www.dmreview.com/master.cfm?NavID=55&EdID=1315.

[OLI 11] OLIVIER CARTON., "Architecture Modèle/Vue/Contrôleur", http:// www.liafa.jussieu.fr/~carton/Enseignement/InterfacesGraphiques/ MasterInfo/Cours/Swing/mvc.html. 15/05/2011.

[SHI] SHILAKESC, TYLMAN J., "Enterprise Information Portals.". http://www.sagemaker.com/company/downloads/eip/indepth.pdf.

LISTE DES ACRONYMES

API	Application Programming Interface
CRUD	Create Read Update Delete
DOM	Document Object Model
DSA	Data Staging Area
DTD	Document Type Definition
DW	Data Warehouse
DWEP	DataWarehouse Engineering Process
E/R	Entité /Relation
ED	Entrepôt de données
ETL	Extract Transform Load
FASMI	Fast Analysis of Shared Multidimensional Information
GUI	Graphical User Interface
HOLAP	Hybrid On-Line Analytical Processing
IDE	Integrated Developpement Environnement

IHM	Interface Homme Machine
JDK	Java Development Kit
JVM	JAVA Virtual Machine
MOLAP	Multidimensional On-Line Analytical Processing
MVC	Modèle -Vue-Contrôleur
OLAP	On-Line Analytical Processing
OLTP	On-Line Transaction Processing
OMG	Object Management Group
OMT	Object Modeling Technique
OOSE	Object Oriented Software Engineering
PK	Primary Key
ROLAP	Relational On-Line Analytical Processing
SAX	Simple API for XML
SCD	Slowly Changing Dimension
SGBD	Système de Gestion de Base de Données
SK	Surrogate Key
UML	Unified Modeling Language
W3C	World wide web Consortium
XML	eXtensible Markup Lungage

ANNEXE A

LES DTDs PROPOSEES

Introduction

Dans cette partie, nous présentons les différentes DTDs qu'on a définis pour instaurer une grammaire pour les fichiers XML manipulés par l'outil Master-ETL-Designer.

a. DTD proposées

- **DTD du projet**

Cette DTD définit la structure du fichier XML contenant les détails sur un projet et nous permet de restituer les propriétés d'un projet sauvegaardé. Cette DTD respecte le diagramme de classes définit dans le chapitre V.

```
<!ELEMENT root (Projet)>
<! ATTLIST ProjetNom   CDATA  #REQUIRED
DateDeCréationCDATA #REQUIRED
Description          CDATA  #IMPLIED>
```

- **DTD du modèle**

Cette DTD définit la structure du fichier XML contenant le modèle conceptuel et le modèle logique et nous permet de restituer le modèle en question. Cette DTD respecte le diagramme de classes définit dans le chapitre V.

```
<!ELEMENT Modèle (ModèleConceptuel, ModèleLogique)
<!ELEMENT ModèleConceptuel
(Concept*,Attribute*,Transformation*,ETLConstraint*,Note*,PartOf*,Provider*,
SerialComposition*,Candidate*,ActiveCandidate*,Annotation*)>

<!ELEMENT  Concept (Composant)>
<! ATTLIST  Concept  Identifier  ID  #REQUIRED>

<!ELEMENT  Attribute (Composant)>
<! ATTLIST Attribute  Identifier  ID  #REQUIRED>

<!ELEMENT  Transformation (Composant)>
<!ATTLIST Transformation Identifier                ID          #REQUIRED
TransformationType    CDATA    #REQUIRED>

<!ELEMENT  ETLConstraint (Composant)>
<! ATTLIST ETLConstraint  Identifier                ID          #REQUIRED
ID_TransformationIDREF    #REQUIRED>

<! ELEMENT Note (Composant)>
<! ATTLIST  Note        Identifier        ID            #REQUIRED
Text          CDATA      #REQUIRED
Expression    CDATA #REQUIRED
FunctionType   CDATA #REQUIERD >

<!ELEMENT  Tag (Composant)>
<!ATTLIST Tag Identifier ID #REQUIRED>
```

```
<! ELEMENT PartOf(ID_Attribut+)>
        <! ATTLISTID_AttributID_Attribut IDREF #REQUIRED>
        <! ATTLIST PartOfID_Concept IDREF #REQUIRED>

<! ELEMENT Provider (ID_Attribut_Entree+,ID_AttributSortie+)>
        <! ATTLISTID_Attribut_EntreeAttributIDREF #REQUIRED >
<! ATTLISTID_Attribut_SortieAttribut IDREF  #REQUIRED>
        <! ATTLIST Provider ID_Transformation IDREF #REQUIRED>

<! ATTLISTSerialCompositionID_Tr_InitialeIDREF #REQUIRED
ID_Tr_SubSequente  IDREF#REQUIRED>

<!ELEMENT  Candidate(ID_Candidat+)>
<! ATTLIST ID_CandidatID_CandidatIDREF #REQUIRED >
<! ATTLIST Candidate ID_ConceptIDREF  #REQUIRED>

<! ATTLIST  ActiveCandidateID_Concept IDREF #REQUIRED
ID_CandidatActif IDREF #REQUIRED>


<!ELEMENT
ModèleLogique(RecordSet*,Attribute*,Activity*,Function*,Tag*,PartOf*,PartOf_In*,PartOf
_Out*,PartOf_Rej*,PartOf_Par*,PartOf_Fun*,InstanceOf_FunctionType*,InstanceOf_DataTy
pe*,Provider*,Regulator*,Annotation*)>

<!ELEMENT  RecordSet (Composant)>
<ATTLIST RecordSetIdentifier ID #REQUIRED>

<!ELEMENT  Attribute (Composant)>
<! ATTLIST Attribute Identifier ID #REQUIRED>

<!ELEMENT  Activity (Composant)>
<! ATTLIST Activity   Identifier ID    #REQUIRED
SémantiqueDeSortie      PCDATA      #IMPLIED
SémantiqueDeRejet       PCDATA      #IMPLIED>
```

```
<! ELEMENT  Function (Composant)>
        <! ATTLIST Function   Identifier ID #REQUIRED>

<!ELEMENT  Tag (Composant)>
        <! ATTLIST Tag IdentificateurID#REQUIRED>

<!ELEMENT  PartOf(ID_Attribute+)>
      <! ATTLISTPartOfID_RecordSetIDREF #REQUIRED>

<!ELEMENTPartOf_In(ID_Attribute+)>
<!ATTLIST PartOf_In   ID_ActivityIDREF #REQUIRED>

<!ELEMENTPartOf_Out(ID_Attribute+)>
<!ATTLISTPartOf_OutID_Activity IDREF#REQUIRED>

<! ELEMENT PartOf_Rej(ID_Attribute+)>
<!ATTLIST PartOf_RejID_Activity IDREF #REQUIRED>

<!ATTLISTID_AttributeID_AttributeIDREF #REQUIRED>

<!ELEMENT  PartOf_Par(ID_Parameter+)>
<! ATTLIST  PartOf_ParID_Activity IDREF #REQUIRED>

<!ELEMENT  PartOf_Fun(ID_Parameter+)>
<! ATTLIST PartOf_FunID_Function IDREF #REQUIRED>

<! ATTLIST ID_ParameterID_Parameter IDREF #REQUIRED>

<!ATTLIST  InstanceOf_DataTypeID_Attribute     IDREF #REQUIRED
DataType CDATA  #REQUIRED>

<! ATTLIST InstanceOf_FunctionTypeID_FunctionIDREF  #REQUIRED
FunctionTypeCDATA  #REQUIRED>

<! ATTLIST  Provider     ID_AttributeL_Provider IDREF  #REQUIRED
ID_Attribute_Consumer  IDREF  #REQUIRED>

<! ATTLIST  RegulatorID_Attribute IDREF  #REQUIRED
ID_Parameter  IDREF #REQUIRED>
```

```
<! ATTLISTID_AttributeID_Attribute IDREF #REQUIRED>

<! ATTLIST  AnnotationID_NoteIDREF  #REQUIRED
ID_EntityIDREF  #REQUIRED>

<! ATTLIST Composant
libellé         CDATA       #REQUIRED
x               CDATA       #REQUIRED
y               CDATA       #REQUIRED
longueur        CDATA       #REQUIRED
largeur         CDATA       #REQUIRED>
```

- **DTD de la bibliothèque des modèles**

La bibliothèque des modèles contient tous les types d'entités prédéfinis dans le système et ceux définis par le concepteur. Les différents types d'entités sont stockés dans des fichiers XML que voici les DTDs spécifiant leurs syntaxes.

1- DTD des types de structure

```
<!ELEMENT Structures (StructureType*)>
<! ATTLIST  StructureType     Identifier ID #REQUIRED
 NomCDATA  #REQUIRED
Description         CDATA     #IMPLIED>
```

2- DTD des types de fournisseur

```
<!ELEMENT Providers (ProviderType*)>
<! ATTLIST ProviderType   Identifier ID #REQUIRED
Nom CDATA  #REQUIRED
Description CDATA  #IMPLIED>
```

3- DTD des types de transformation

```
<!ELEMENT Transformations (TransformationType*)>
< ! ATTLISTTransformationType  Identifier ID #REQUIRED
   Nom CDATA #REQUIRED
 Description CDATA #IMPLIED
Symbôle CDATA #IMPLIED>
```

4- DTD des types de données

```
<!ELEMENT DataTypes (DataType*)>
<!ELEMENTDataType(Constante*)>
<! ATTLISTConstanteConstante  CDATA #REQUIRED>
<! ATTLISTDataTypeIdentificateurID #REQUIRED
   Nom CDATA #REQUIRED
Description CDATA  #IMPLIED>
```

5- DTD des types de fonctions

```
<! ELEMENT Functions (FunctionType*)>
<! ELEMENT FunctionType (AttributDEntree*,AttributDeSortie*)>
<! ATTLIST AttributDEntreeAttributCDATA  #REQUIRED>
<! ATTLIST AttributDeSortieAttributCDATA  #REQUIRED>
<! ATTLIST FunctionType IdentifierID #REQUIRED
                        Nom          CDATA #REQUIRED
                        Description  CDATA #IMPLIED>
```

6- DTD des types d'activité

```
<!ELEMENT Activities (ActivityType*)
<!ELEMENT ActivityType(Paramètre*,AttributDeSortie*)>
<! ATTLIST AttributDeSortieAttributCDATA   #REQUIRED>
<! ATTLIST ParamètreParamètre    CDATA   #REQUIRED>
```

```
<! ATTLIST ActivityType  Identifier  ID  #REQUIRED
 Nom CDATA  #REQUIRED
 Description CDATA #IMPLIED
SémantiqueDeSortie    CDATA   #IMPLIED
SémantiqueDeRejet     CDATA    #IMPLIED
DataProvider    (SELECT  | DELETE)  #IMPLIED
DataConsumer  (APPEND | OVERWRITE) #IMPLIED>
```