République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saâd Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme Master en informatique.
Option : Ingénierie des logiciels

Sujet :

Intégration des bases de données hétérogènes dans le processus de construction d'un DataWhareHouse

Présenté par : - ALIMAZIGHI Hadjer
- ATTAF Sarah

Proposé par - M.BALA
Promoteur - M.BALA

Soutenus le: , devant le jury composé de :

Nom président du jury, grade, organisme	Président
Nom examinateur 1, grade, organisme	Examinateur
Nom examinateur 2, grade, organisme	Examinateur

- 2008/2009-

Dédicace

Au prunelles de mes yeux

A ces êtres respectueux, Qui puissent exister sur terre,

A ceux qui ont su m'élevé, m'éduqué,

A ceux qui m'ont apporté leur grand amour, leur patience et qui m'ont nourrit de leur sagesse et bons conseils.

A ceux qui ont forgé ma personnalité et qui ont beaucoup contribué à devenir ce que je suis,

A ceux qui ont été à mes cotés dans mes joies et mes peines

« A toi cher père, A toi ma chère mère »

A ma chère sœur Nadia, à mes frères Mohamed, Ouahid et Walid , A ma belle soeur Amina, mes chère neveux Larbi et abdelmalek.

A ma chère binôme Hadjer et sa famille

A chère amie Imane

SARAH

Dédicace

C'est à ma famille que j'ai dédié ce mémoire, en particulier à mes très chers parents, qui sont ce que j'ai de plus cher au monde, Que dieu me les protègent «in chae Allah» et sans eux ma réussite n'aurait jamais eu lieu ...

a toi père que j'aime énormément et qu'il trouve ici mon éternelle reconnaissance ,pour m'avoir toujours fait confiance, a toi mère qui m'a toujours encouragé ,et a cru en moi cette année et tant d'autres, et pour ton soutien morale ,j'espère avoir un jour l'occasion de prouver ma gratitude la plus infime.

A ma grande sœur et mon amie «Khadidja » avec qui, j'ai passé de moment inoubliable et agréable pendant ces années études, et sans oublie ces adorables filles «Maroua », «Yassmine» , son petit fils « Mohamed »,et son mari «Samir ».

A ma très chère sœur «Souhila » et son mari «Abd-el-krim» qui m'ont toujours conseillé et encouragé durant ces années études, et à leurs fils «Aymen », «Monsif » et leurs nouveau née «Meriem».

A ma chère sœur «Sihem»,et mon frère adorable «Abdenour», je vous souhaite tous le bonheur du monde.

A ma chère binôme « Sarah » et sa famille.

Pour votre soutien, votre humour et surtout votre amour, je vous dis «Conférer à ce merci un indice d'absolue et vous serez encore loin du compte ».

HADJER

REMERCIMENT

Tout d'abord, on tient à remercier dieu de nous avoir donné la force pour accomplir ce travail.

Nos remerciements sont destinés vers toute la famille des enseignants du département Informatique qui nous ont formé le long de notre cycle.

Nous tenons à remercier vivement notre promoteur, Monsieur BALA Mahfoud de nous avoir proposé le thème de notre mémoire, pour avoir accepté de diriger notre travail. Nous le remercions pour sa lecture attentive, ses critiques et suggestions constructives, son aide, sa disponibilité, ses conseils, et son soutien durant toute cette période, *nous ont toujours redonné confiance et volonté. Qu'il trouve ici l'expression de notre sincère reconnaissance et de notre profonde gratitude.*

Enfin nous remercions tous ceux qui nous ont aidés à accomplir notre travail, de prés ou de loin.

Sommaire

Introduction générale

Partie I : État de l'art

Chapitre 1
Datawarehouse

Chapitre 2

Approches d'intégration de données

Partie II Conception du système

Chapitre 1
Sources de données traitées et modèle pivot

Chapitre 2
Modélisation d'ETL Data

Partie III : Mise en œuvre

Chapitre 1 Processus d'alimentation

Chapitre 2 Réalisation

Conclusion générale

LISTE DES TABLEAUX

Partie I :État de l'art
Chapitre I DataWareHouse

Partie II : Conception du Système

Chapitre 1 Sources de données traitées et modèle pivot

Chapitre 2 Modélisation d'ETL Data

LISTE DES FIGURES

Partie I :État de l'art

Chapitre 1 DataWareHouse

Chapitre 2 Approches d'intégration de données

Partie II : Conception du Système

Chapitre 1 Sources de données traitées et modèle pivot

Chapitre 2 Modélisation d'ETL Data

Partie III : Mise œuvre du système

Chapitre1 Alimentation du datamart

Chapitre2 Réalisation

1. Contexte général

Les technologies de l'information étaient à l'origine réservées à quelques utilisateurs spécialisés. Aujourd'hui, elles font partie intégrante de nos activités quotidiennes.

Ainsi le monde informatique regorge de données aux formats très hétérogènes, autrement dit, utilisent des modèles différents pour la représentation de l'information qu'il est nécessaire d'intégrer pour construire des applications. En effet, l'hétérogénéité des sources de données se situe à plusieurs niveaux.

Dans un tel contexte, le besoin d'intégration se fait de plus en plus sentir. Cependant, pour répondre à ce besoin, le développement des applications d'intégration se voit contraint de composer avec la répartition des sources et l'hétérogénéité de leurs structures et de gérer l'intégration entre les données en différents formats qu'ils manipulent.

Cette évolution a également conduit aux développements des systèmes d'informations très complexes, gérant des volumes d'informations très importants. Ces informations sont regroupées dans des sources de données hétérogènes, réparties et autonomes. Afin de les exploiter, deux activités de recherche principales ont vu le jour :

(1) La recherche d'information : dans laquelle plusieurs moteurs de recherche ont été développés, comme Google, Yahoo...etc.

(2) L'intégration de sources de données : afin de faciliter leur exploitation.

Notre travail se focalise sur ce dernier domaine de recherche et traite le cas de l'intégration des bases de données hétérogènes, afin de pouvoir les intégrer dans un Entrepôt de données, ou des systèmes d'aide à la décision.

2. Problématique

La constitution d'entrepôts de données est une réponse au problème de l'intégration d'une grande quantité de données variées, relatives à un certain domaine d'application, et stockées physiquement dans différentes sources de données. L'entrepôt de données regroupe certaines informations sous une forme exploitable par des traitements utiles pour l'aide à la décision. Ces informations, qui

sont potentiellement pertinentes pour telle ou telle catégorie de décideurs du domaine, doivent être :

- Extraites : Accéder à la majorité des systèmes de stockage de données (SGBD, documents XML, ...) et récupérer les données pertinentes.
- Transformées : Transformer les données avant de les charger dans le datawarehouse
- fusionnées : Insérer les données dans le Data Warehouse.

  L'intégration nécessite alors l'accès aux données des sources et la capacité de les fusionner.

  De manière générale, la problématique peut se résumer comme suit :

  - *Comment permettre à l'administrateur d'un entrepôt de données de récupérerer des données pertinentes qui se situent dans des sources différentes de données distribuées, hétérogènes ; et les transformer afin de les charger dans le datawarehouse*

Des solutions logicielles sont alors nécessaires à leur intégration et à leur homogénéisation, Ces outils ont pour objet de s'assurer de la cohérence des données du Data Warehouse et d'homogénéiser les différents formats trouvés dans les bases de données opérationnelles.

3. Objectif

Ce mémoire vise principalement à étudier les différents systèmes d'intégration des données ; plus particulièrement l'intégration de données dans un entrepôt de données.

Le but de notre travail consiste à présenter une approche d'intégration permettant d'intégrer le plus grands nombre de sources de données tout en s'appuyant sur le modèle relationnel comme modèle pivot. Notre système devra considérer spécialement les données provenant des SGBD SQL Server et PostgreSQL avec des schémas différents, en plus des documents XML

L'approche suivie est l'approche ***fédérée fortement couplée.*** Cette dernière est l'une des solutions rentables pour la problématique de l'intégration des sources de données hétérogènes qui offre à l'utilisateur une vue uniforme et centralise des

données distribuées, cette vue pouvant aussi correspondre à une vision plus abstraite, via le schéma fédéré fourni par cette approche.

4. Organisation de mémoire

Le présent mémoire est structuré comme suit :

- La première partie est composée de deux chapitres

  - ✓ Nous présentons dans le premier chapitre une étude détaillée sur les datawarehouse
  - ✓ Dans le deuxième chapitre nous présentons les différents approches d'intégration de données existantes on se focalisant sur l'approche fédérée fortement couplée suivie pour l'intégration de données dans un datawarehouse.

- La deuxième partie est composé de deux chapitres .

  - ✓ Nous présentons dans le premier chapitre les différents formats de données choisis pour être traité comme des données sources dans notre système afin de les intégrer dans de datawarehouse.
  - ✓ Nous allons présenter dans le deuxième chapitre la démarche de conception de notre application et les fonctionnalités offertes par notre interface.

- La troisième partie est composée de deux chapitres.

  - ✓ Nous décrivant dans le premier chapitre le module de traitement de la requête d'alimentation du datawarehouse .
  - ✓ Nous allons montrer dans le deuxième chapitre les détails de la réalisation (l'environnement de développement matériel et logiciel) et la présentation des différentes interface de notre application.

Partie I

État de l'art

Introduction

Un *Datawarehouse* répond aux problèmes de données surabondantes et localisées sur différents systèmes hétérogènes, c'est une architecture capable de servir de fondation aux applications décisionnelles. Pour être exploitables, toutes les données provenant des systèmes *distribués* doivent être organisées, coordonnées, intégrées et enfin stockées pour donner à l'utilisateur une vue globale des informations.

La phase d'intégration de schémas dans un entrepôt de données est très particulière pour la raison suivante: le schéma résultant n'est pas un schéma conceptuel intégré définitif mais un schéma *très évolutif*, et d'autre part le schéma global permet de simplifier la vu à l'administrateur, (le schéma fédéré donne l'impression à l'utilisateur qu'il utilise un seul schéma)

Différentes *approches d'intégration* ont été développées, ainsi que des systèmes basés ces approches sont développés mais qui demeurent toujours restreints à cause de la complexité du problème d'intégration.

Chapitre I DataWareHouse

Introduction

Les systèmes de production sont caractérisés par une activité constante composée de modifications et d'interrogations fréquentes des bases de données par de nombreux utilisateurs .À l'inverse, nul besoin de modification de données dans les systèmes d'informations de décision. En effet, le but principal consiste en l'interrogation du système d'information afin de permettre l'analyse des indicateurs pertinents pour faciliter les prises de décisions. Les différences entre le monde opérationnel et décisionnel peuvent être résumées ainsi [W1]:

DECISIONNEL	OPERATIONNEL
Gros volumes de données à gérer.	Petits volumes de données à gérer.
Processus ouverts pour permettre la génération de connaissance.	Processus fermés, transactionnels, le but est de donner le moins de marge possible.
Données en lecture seule.	Données en lecture - Écriture.
Rapidité moyenne comparée aux systèmes opérationnels	Réponses très rapides.
Niveau de granularité très grand (on peut avoir des résumés sur ce qui s'est passé durant les 10 dernières années par exemple).	Niveau de granularité fin.
Centralisés (on veut avoir toutes les données de l'entreprise dans une seule structure).	Décentralisés

Tableau1.1.1 : Comparaison entre le monde opérationnel et décisionnel

Dans un système décisionnel, le Data Warehouse est utilisé pour accumuler sur une longue période les données opérationnelles qui seront exploitées par les outils de reporting, d'analyse et de prise de décision, pour suivre l'activité ou déterminer des tendances. Ce chapitre permettra de présenter une étude bien détaillée sur les datawarehouse.

1. Entrepôt de donnée « DataWareHouse »

1.1. Définition

« Un datawarehouse est une collection de données thématiques, intégrées, non volatiles et historisées, organisées pour la prise de décision. » (Inmon, 1996).

Les entrepôts de données sont un type de systèmes d'intégrations de données centralisées. Il s'agit d'un stockage de données consolidées et rassemblées en un même endroit, à partir de plusieurs sources hétérogènes. D'où dans un entrepôt de données les informations doivent être restructurées afin de les rendre plus facilement compréhensibles et utilisables. Il faut par conséquent extraire, transformer et agréger les données à traiter [SAN 01].

1.2. Caractéristiques d'un entrepôt de données

- ***Orientation sujet (thématique)***

Le Datawarehouse est organisé autour des sujets qui ont un intérêt majeur pour l'entreprise. On assemblera à cet effet, les informations par thèmes contrairement aux modélisations traditionnelles (transactionnelles) qui regroupent les informations par fonctions. L'intérêt de cette organisation est de passer d'une vision verticale de l'entreprise à une vision transversale, beaucoup plus riche.[GRI 07]

- ***Données intégrées***

Les données sont mises en forme selon un standard afin d'obtenir la transversalité recherchée. Cela nécessite une forte normalisation, une bonne gestion des référentiels et de la cohérence, une parfaite maîtrise de la sémantique et des règles de gestion des données manipulées. Lors de l'alimentation des données, ces dernières sont hétérogènes et proviennent de systèmes opérationnels différents. Il faut doter ces données d'une codification unique et pertinente afin qu'elles puissent aisément s'intégrer dans le Datawarehouse. Il faudra donc faire appel à des

conventions de nommage, des structures de codage, qualifier les mesures et réaliser l'intégration de la sémantique. [GRI 07]

- ***Données non volatiles***

Les données intégrées dans l'entrepôt le sont utilisées en mode consultation et ne peuvent subir aucune altération. Cela se justifie pour assurer la fiabilité des résultats des requêtes. Ainsi, une même requête lancée à plusieurs mois d'intervalle donnera toujours les mêmes résultats. Cela permet au Data Warehouse d'acquérir au cours du temps un historique détaillé de l'activité de l'entreprise. Ceci s'oppose fondamentalement à la logique des systèmes de production qui remettent à jour les données qui sont de nature volatiles. [GRI 07]

- ***Données historisées***

L'ensemble des données qui sont intégrées dans l'entrepôt contient un ensemble de caractéristiques qui sont datées. L'historisation est nécessaire pour suivre dans le temps l'évolution des différentes valeurs des indicateurs à analyser. Ainsi, un référentiel temps doit être associé aux données afin de permettre l'identification dans la durée de valeurs précises. Si tel n'était pas le cas, l'analyse ne serait pas possible, le suivi de l'évolution non plus. Cela rejoint la notion de non volatilité expliquée précédemment. [GRI 07]

1.3. Les composants du Datawarehouse

La Figure suivante nous illustre les différents composants du Datawarehouse présenté par Ralph Kimball :

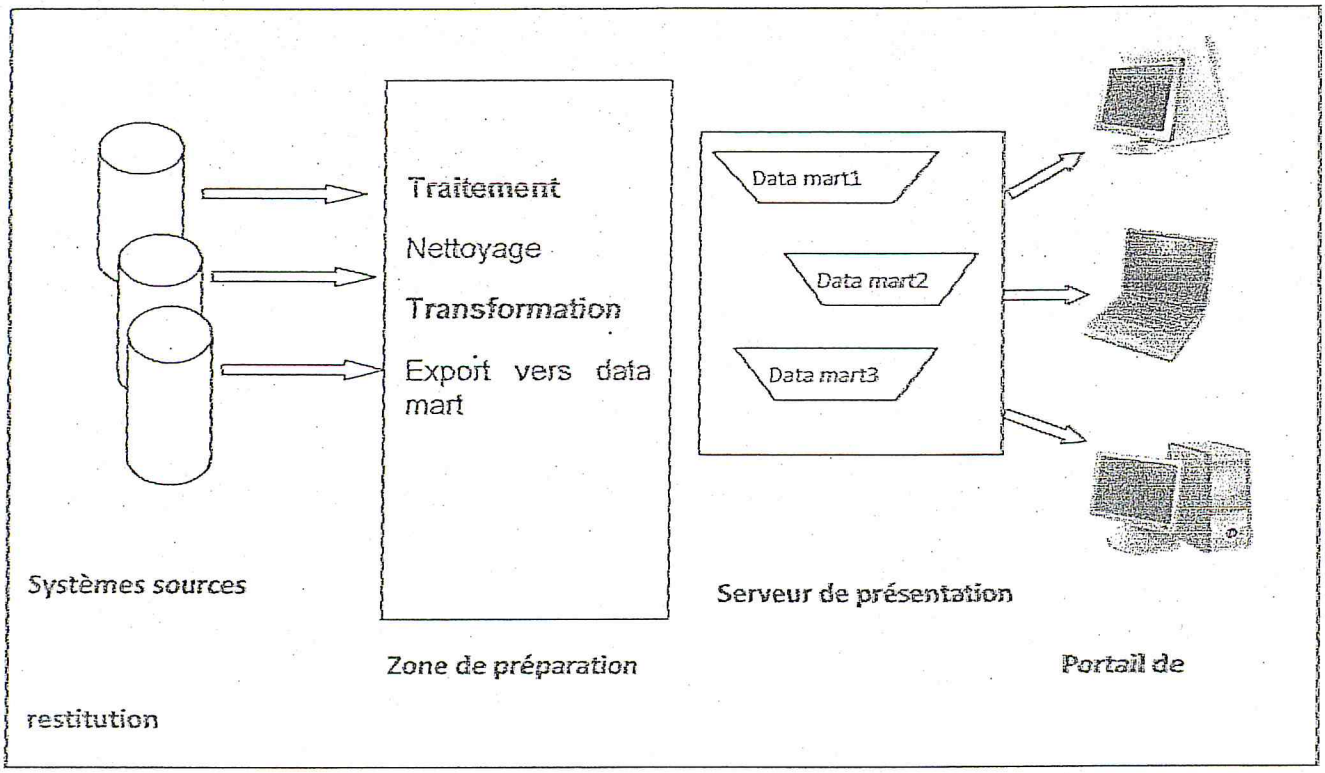


Figure1. 1.1 les composants du Datawarehouse[KIM 01]

1.3.1 Les sources de données

Système opérationnel d'enregistrement. Il permet de capturer les transactions liées à l'activité de l'entreprise. Il s'agit souvent de ce que l'on appelle les applications de gestion de l'entreprise. La principale priorité du système source est le temps de disponibilité. Cette partie sera bien détaillée dans la partie « Systèmes sources d'un datawarehouse »

1.3.2 La zone de préparation des données

Cette zone comprend tout ce qui se trouve entre les systèmes sources et le serveur de présentation. Elle regroupe l'ensemble des processus qui nettoient, transforment, combinent, archivent, suppriment les doublons. Elle prépare les données sources en vue de leur intégration et de leur exploitation dans le datawarehouse. La frontière qui détermine la zone de préparation des données est que la zone de préparation des données ne doit en aucun cas être accessible à

l'utilisateur final par requêtes ou par un quelconque autre service de présentation.[KIM 01]

1.3.3 Le serveur de présentation

Ce composant correspond à la machine cible sur laquelle l'entrepôt de données est stocké et organisé pour répondre en accès direct aux requêtes provenant des utilisateurs, des générateurs d'états ou d'autres applications. Sur le serveur de présentation, les données sont stockées et présentées sous une forme dimensionnelle, de façon à faciliter l'accès pour l'utilisateur final. Dans la majorité des cas, le serveur est basé sur une base de données relationnelle, de sorte que les tables y sont organisées sous forme de datamart. [KIM 01]

1.3.3.1 Le Datamart ou magasin de données

Un Datamart est un sous Sous ensemble d'un entrepôt de données, contenant des informations se rapportant à un secteur d'activité particulier de l'entreprise ou à un métier qui y est exercé (un Datamart pour les ventes, pour les commandes, pour les ressources humaines) [W1]

1.3.3.2 L'entrepôt de données ou Datawarehouse

Le Datawarehouse comme l'union de tous les Datamarts qui le composent. L'entrepôt de données est alimenté par la zone de préparation des données.

1.3.3.3 Le référentiel de Meta données

Le dictionnaire des données ou référentiel est au cœur du Data Warehouse décrit Les métadonnées nécessaires à l'administration et à la gestion de l'entrepôt de données.

Les métadonnées sont des « données sur les données » et définissent les informations relatives à l'entrepôt et aux processus associés.
Les métadonnées gèrent le contrôle de l'information en assurant sa fiabilité, sa cohérence sa réplication, sa distribution, leur historisation et la détermination du périmètre de calcul des données. [GRI 07]

1.3.4 Portail de restitution

Permet de représenter ce que voient les utilisateurs, les outils avec lesquels ils travaillent. Sous ce terme sont regroupées toutes les applications qui s'appuient sur les données du data warehouse pour les restituer soit à l'utilisateur, soit à une autre application. Les services offerts par le portail de restitution sont les services d'accès aux données, les applications de modélisation et de data mining. Les services d'accès aux données comprennent : la navigation dans l'entrepôt, la surveillance de l'activité, la gestion des requêtes et la génération d'états standards. Les applications de modélisation offrent différents types d'analyses basées sur des modèles tels que modèles financiers, systèmes d'évaluation des clients, optimisation des processus et prévisions, ainsi que les activités centrales du datamining telles que la catégorisation, la classification, l'estimation et prédiction.

1.4 Modélisation dimensionnelle

La modélisation dimensionnelle est une méthode de conception qui vise à présenter les données sous une forme qui permet des accès hautement performants. Elle adhère totalement à la dimensionnalité ainsi qu'à une discipline qui exploite le modèle relationnel. Chaque modèle dimensionnel, se compose d'une table contenant des clés multiples, appelée « table de fait », et un ensemble de tables plus petites, nommées « tables de dimension».

Chacune de ces dernières possède une clé primaire unique, qui correspond exactement à l'un des composantes des clés multiples de la table de fait.[WI1]

1.4..1 La modélisation conceptuelle

1.4.1.1 Définition du Fait

Le fait est le sujet à analyser, il est formé de mesures correspondantes aux informations de l'activité analysée

Une table de fait est une table qui contient les données observables (les faits) que l'on possède sur un sujet et que l'on veut étudier, selon divers axes d'analyse (les dimensions). [TES 00]

Prenons comme exemple le fait « chiffre d'affaire » qui peut être analysé par rapport à trois axes d'analyse (trois dimensions) : la dimension produit, la

dimension région, et la dimension temps. ***Un fait est tout ce qu'on voudra analyser.***

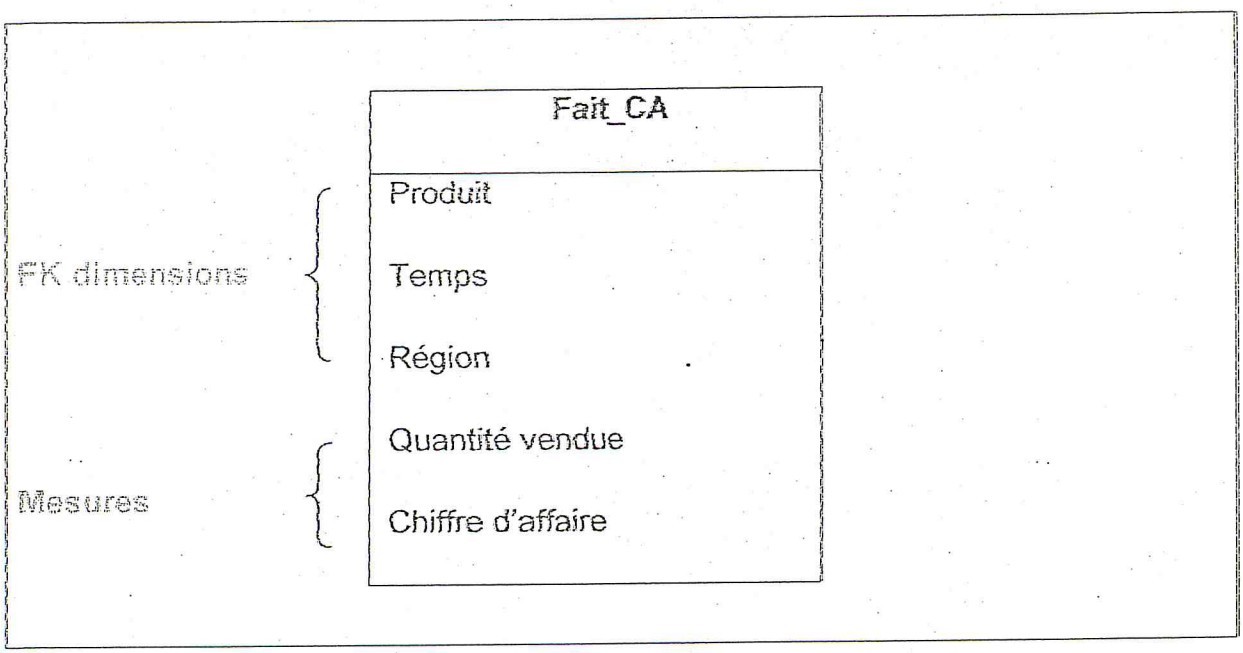


Figure1.1.2. Table de fait

1.4.1.2 Définition de la dimension

On entend par dimensions les axes avec lesquels on veut faire l'analyse. Il peut y avoir une dimension Temps, une dimension produit, une dimension région (pour faire des analyses par secteur géographique), etc. ***Une dimension est tout ce qu'on utilisera pour faire nos analyses.*** [W1]

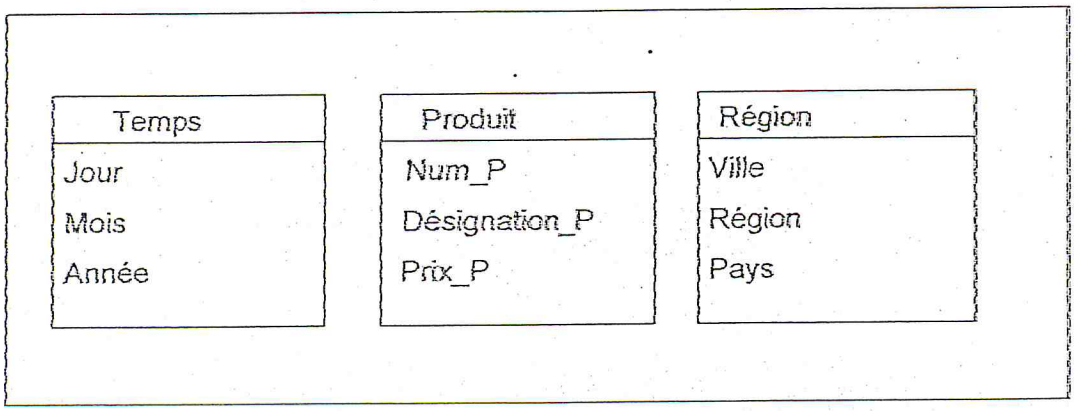


Figure1.1.3 Table de dimension

1.4.1.3 Modèle en étoile

Un schéma en étoile est une façon de mettre en relation les dimensions et les faits dans un entrepôt de données. Le principe du ce modèle est que les dimensions sont directement reliées à un fait (schématiquement, ça fait comme une étoile).[W1]

Avantage

- Facilité de navigation pour l'utilisateur
- Performances (Limitation du nombre des jointures)
- Gestion aisée des agrégats
- Fiabilité des résultats

Inconvénient

- Redondance dans les tables de dimension
- Procédures d'alimentation complexes

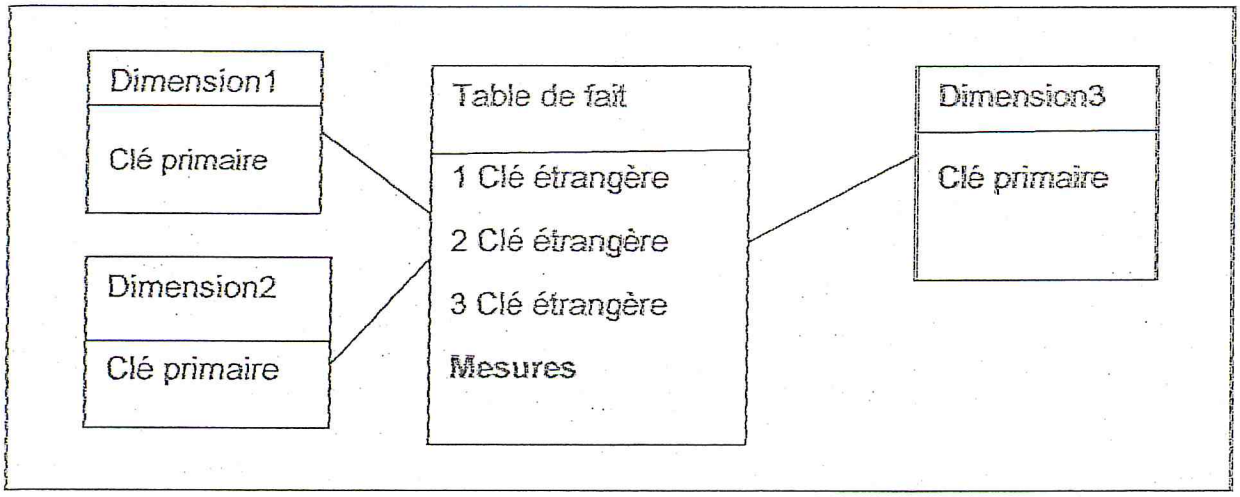


Figure1. 1.4. Modèle en étoile

1.4.1.4 Modèle en flocon

Un autre modèle de mise en relation des dimensions et des faits dans un entrepôt de données. Le principe étant qu'il peut exister des hiérarchies de dimensions et qu'elles sont reliées aux faits, ça fait comme un flocon.

Le schéma en flocon de neige est un schéma en étoile, dont les dimensions sont normalisées la normalisation en 3ième forme normale, pour éviter les redondances des lignes. La modélisation en flocon existe pour des raisons de performances. En

effet, des dimensions de plusieurs millions de lignes (modèle en étoile) peuvent poser des problèmes de lenteur lors l'exploitation des données.[W2]

Avantages

- Réduction du volume de la base
- Permettre le drill-down sur la dimension

Inconvénients

- Navigation plus difficile
- Jointure plus nombreuses

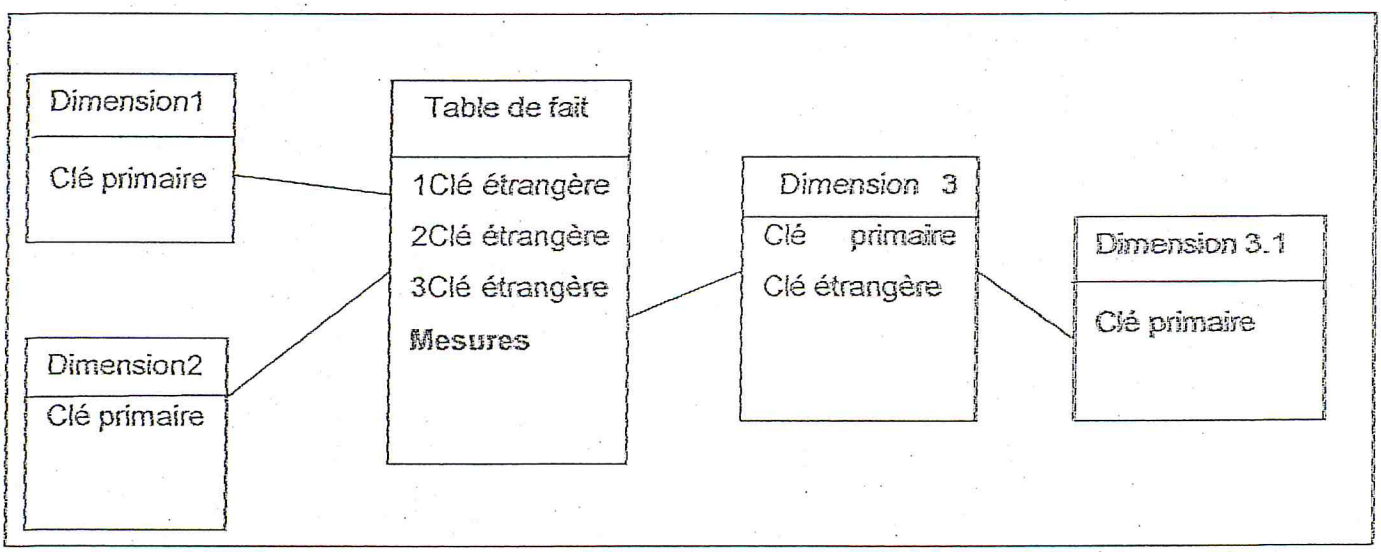


Figure1. 1.5 Modèle en flocon

1.4.2 La modélisation logique

Au niveau logique plusieurs possibilités sont envisageables pour la modélisation multidimensionnelle .Il est possible d'utiliser [TES 00]

- Un système de gestion de base de données (SGBD) existant tel que le SGBD relationnels (ROLAP). Ou bien SGBD orientés objet (OOLAP).
- Un système de gestion de base de données multidimensionnelle (MOLAP)

1.4.2.1 ROLAP et OOLAP

L'approche la plus utilisée consiste à utiliser un système de gestion de base de données relationnelles. On parle de l'approche ROLAP(Relational On-Line Analytical Processing). Le modèle multidimensionnel est traduit de la manière suivante :

- Chaque fait correspond à une table , appelée table de fait .
- Chaque dimension correspond à une table , appelée table de dimension .

Ainsi, la table de fait est constitué d'attributs représentant les mesures d'activité et les attributs clés étrangères de chaqu'une des tables de dimension, les tables de dimension contiennent les paramètres et une clé primaire permettant de réaliser des jointures avec la table de fait.

Plus récemment, une autre approche s'appuie sur le paradigme objet ; On parle de l'approche OOLAP (Object On-Line Analytical Processing) . Le modèle se traduit ainsi :

- Chaque fait correspond à une classe, appelé classe de fait
- Chaque dimension correspond à une classe, appelé classe de dimension .

1.4.2.2 MOLAP

MOLAP (Multidimensionnelle On Analytical Processing) consiste à utiliser un système multidimensionnel pur qui gère des structures multidimensionnelles natives. Les structures multidimensionnelles natives utilisées sont des tableaux a N dimensions (les cubes), cette approche stocke les données de manière multidimensionnelle, l'intérêt est que les temps d'accès sont optimisés mais cette approche nécessite de redéfinir des opérations pour manipuler ces structures multidimensionnelles. Les bases MOLAP sont rapides et performante mais limités au gigaoctet. [SAN 01]

1.5 L'alimentation du Datawarehouse

Alimenter un entrepôt de données , est la difficulté technique majeur et la plus couteuse .Cela due à l'hétérogénéité des données au niveaux des sources .Il existe des logiciels permet d'assurer la cohérence des données de l'entrepôt de données , et homogéniser les différents formats trouvés dans les base de données opérationnelles ,telles que DTS (Data Transformation Services), et Kettle (outil ETL Open Source). Les étapes d'alimentation d'un entrepôt de données sont :
Découverte des données et *l'acquisition des données* .

1.5.1 Découverte des données

Il s'agit de localiser dans le système opérationnel les données qu'il est nécessaire de prélever. Cette étape est importante dans la mesure où elle va déterminer le niveau de finesse des analyses du Data Warehouse : il s'agit de la granularité.
Prendre un trop grand nombre de données complexifiera les étapes d'intégration des données, mobilisera d'autant plus de capacités système et d'espace de stockage pour l'entrepôt de données.A l'inverse, diminuer le nombre d'informations peut limiter, voire fausser les analyses de l'entrepôt.[GRI 07]

1.5.2 L'acquisition des données

Acquisition des données se compose de trois phase : l'extraction , la préparation et le chargement

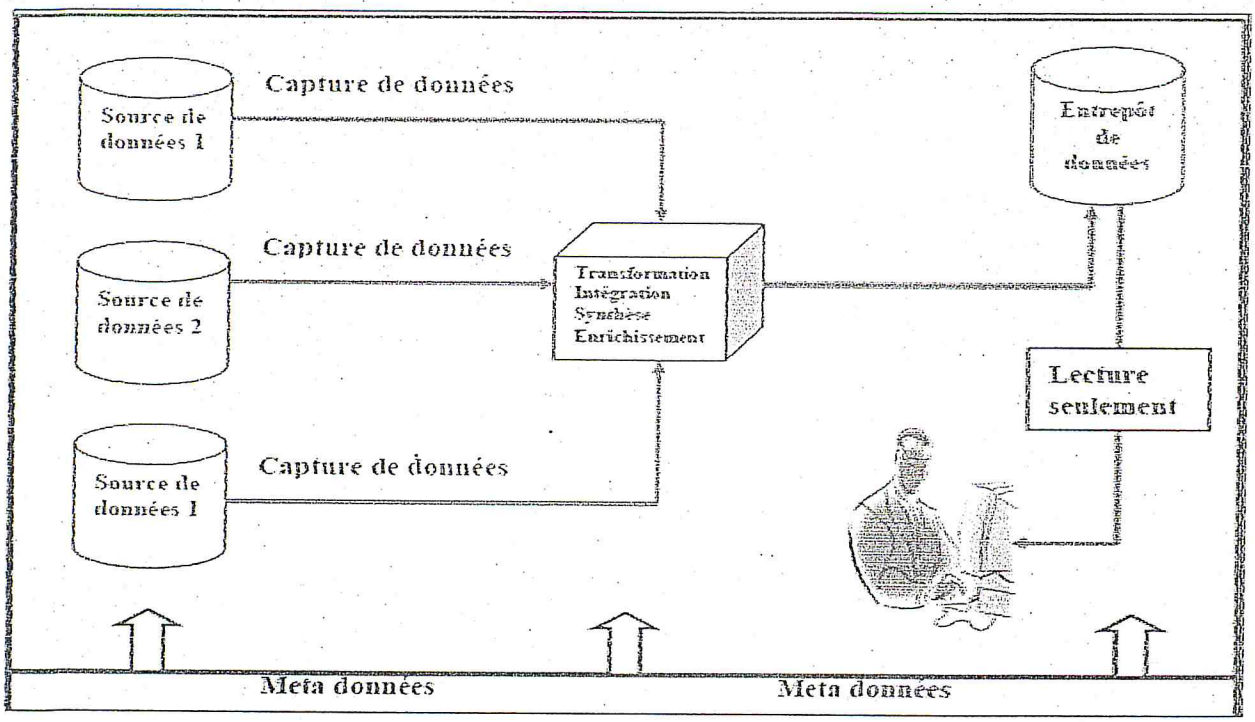


Figure1. 1.6 Les phases de l'acquisition de données

1.5.2.1 Extraction des données

Les données sont en général hétérogènes du fait de la diversité des systèmes de données sources : il peut s'agir de fichiers plats, de bases de données relationnelles. Les données sont donc complexes (organisation transactionnelle) et diffuses (les systèmes sources sont multiples, et géographiquement éloignés). Toutes ces données doivent être prélevées sans pour autant perturber les systèmes sources. [GRI 07]

1.5.2.2Transformation

Le système décisionnel se doit de fournir des informations fiables car celles-ci serviront de base pour prendre les décisions stratégiques de l'entreprise. Cela repose sur la qualité des données au sein de l'entrepôt et c'est l'étape de transformation qui se charge de la garantir.

Le transformateur se charge de nettoyer les données afin de les rendre compatibles avec la granularité et le schéma du DataWarehouse . Ce processus a été étudié dans un contexte multi base de données dont voici les principales caractéristiques : Il s'agit tout d'abord de recenser les entités dont la sémantique est similaire. Ainsi, une même entité peut être représentée de manière

différente dans les diverses sources et réciproquement. Cette tâche est assez délicate à gérer, c'est pourquoi on fait appel à l'intervention humaine .

Ensuite, il faut identifier et résoudre les éventuels conflits entre les entités. La cardinalité des champs sources et cibles doit être respectée : une adresse complète peut être décomposée en rue, ville et code postal par exemple. Les données synonymes doivent être rassemblées : Par exemple, la désignation femme peut dans les systèmes sources être désignée par les données suivantes : « F », « Femme », « 1 » en cas de codification numérique. L'utilisateur définit que ces appellations correspondent à la donnée « femme ».

Enfin, il faut s'assurer de la cohérence des données en supprimant les doublons grâce à des filtres prédéfinis : les données manquantes ou incohérentes sont ainsi corrigées. [GRI 07]

1.5.2.3 Chargement ou rafraichissement

Une fois que les données sont fusionné, ils intégrées peuvent être chargées dans le Datawarehouse. Cette étape s'appelle le chargement ou le rafraîchissement. [GRI 07]

Il existe des logiciels, qui contrôle les phases d'extraction ,de données , de transporte et de chargement, ces logiciels s'appelle ETL(Extract, Transform & Load) Cet outil doit pouvoir se connecter aux sources qu'il s'agisse des applications (métiers ,ERP...), des fichiers ou des bases en production. En ce sens ,il joue un role d'intégration .Il fait partie donc parti d'un sous ensemble des EAI (Entreprise Application Integration), domaine plus général regroupant toutes les formes d'intégration entre des applications ,des processus ou/et des interfaces ,L'ETL se positionne sur l'intégration de données .

Âpres avoir été paramétré suivant les besoins décisionnel, avec les données en entrée, les données en sortie et les processus de transformation à effectuer, l'ETL effectue l'alimentation, les mêmes processus de transformations sont appliqués de manière récurrente lors de chaque alimentation. Le traitement étant très gourmands, l'alimentation s'exécute souvent la nuit, pour ne pas impacter les ressources machines et réseaux pendants les heures de bureau. [SAN 01]

1.6 Conception du Datawarehouse

Un vrai entrepôt de données, selon la définition officielle „est une vue complète et centralisée des données de l'entreprise. La modélisation en étoile ou en flocon, elle, ne s'intéresse qu'à la conception d'un sous ensemble d'entrepôt, ***une seule table de fait.*** La fonction commerciale d'une entreprise peut comporter une étoile pour les ventes, un flocon pour les commandes, une autre étoile pour les retours, etc.

Ce qui est juste, c'est qu'un entrepôt de données est l'ensemble de ces étoile et/ou flocons.[W2] par fonction ou par utilité dans l'entreprise et un entrepôt est l'ensemble de tous les datamarts de l'entreprise. [W1]

Il existe plusieurs approches pour mettre en place un Datawarehouse. Par contre seulement trois approches sont communes. Il s'agit de :

- l'approche "Top-Down" ou Datamarts dépendant prônée par B.Inmon.
- l'approche "Bottom-up", ou Datamarts indépendant de Kimball.
- l'approche "Hybride" qui dérive des deux premières approches.

1.6.1 Top-Down

C'est la méthode la plus lourde, la plus contraignante et la plus complète en même temps. Elle consiste en la conception de tout l'entrepôt (toutes les étoiles), puis en la réalisation de ce dernier. Imaginez le travail qu'une telle méthode implique: savoir à l'avance toutes les dimensions et tous les faits de l'entreprise, puis réaliser tout ça Le seul avantage que cette méthode comporte est qu'elle offre une vision très claire et très conceptuelle des données de l'entreprise ainsi que du travail à faire.[W1]

1.6.2 Bottom-Up

C'est l'approche inverse, elle consiste à créer les étoiles une par une, puis les regrouper par des niveaux intermédiaires jusqu'à obtention d'un véritable entrepôt pyramidal avec une vision d'entreprise. Les datamarts sont dit indépendants ce qui veut dire qu'il n'existe aucune intégration ou communication entre ces derniers.
L'avantage de cette méthode est qu'elle est simple à réaliser (une étoile à la fois), l'inconvénient est le volume de travail d'intégration pour obtenir un entrepôt de données ainsi que la possibilité de redondances entre les étoiles (car elles sont faites indépendamment les unes des autres).[W1]

1.6.3 Hybride

Cette approche, comme son nom l'indique, est un mix des deux premières approches. On commence par concevoir un modèle de données de l'entreprise en même temps que les modèles spécifiques. Puis on crée un modèle normalisé d'entreprise de haut niveau.[W2]

1.7 Systèmes sources d'un datawarehouse

1.7.1 Type des systèmes sources

Les sources du datawarehouse.peuvent être de plusieurs types :

- **Les sources structurées**

Les sources structurées ont un schéma prédéfini, tout élément de données est défini sous ce schéma , donc le schéma impose un format pour tous les éléments de données de la source (exp : les données relationnelles , et les données objets) .

- **Les sources semi structuré**

Ont une structure qui n'est pas prédéfinie sous forme d'un schéma strict. Cependant chaque élément de données doit porter sa propre définition sémantique sous forme de label (HTML, XML)

- **Les sources non structuré**

N'ont aucune structure (texte, images, son)

1.7.2 Caractéristiques des systèmes sources d'un datawarehouse

Un Datawarehouse est alimenté à partir de systèmes préexistants, ce qui explique l'existence de plusieurs types de sources de données et aussi elles sont caractérisées par *la distribution, l'autonomie, et l'hétérogénéité.*

- **Distribution**

Les données du datawarehouse sont distribuées sur plusieurs bases de données. Ces bases de données peuvent être stockées sur un ou plusieurs systèmes informatiques, ces derniers sont dispersés géographiquement mais reliés par un réseau de communication, ces bases de données distribuées sont gérées par des SGBD réparties.

Système de gestion de bases de données réparties

Un SGBD répartie est un système gérant une collection de bases de données logiquement reliées, réparties sur différents sites en fournissant un moyen d'accès rendant la distribution transparente.

Architecture de schéma d'un système répartie

La plupart des architectures qui décrivent les SGBD répartis sont des extensions de l'architecture à trois niveaux de schémas (interne, conceptuel, externe) définis dans le rapport de l'ANSI/X3/SPARC établi en 1977

Niveau interne qui décrit l'organisation physique de données.

Niveau conceptuel qui correspond au schéma conceptuel des données.

Niveau externe qui correspond aux vues des différentes catégories d'utilisateurs.

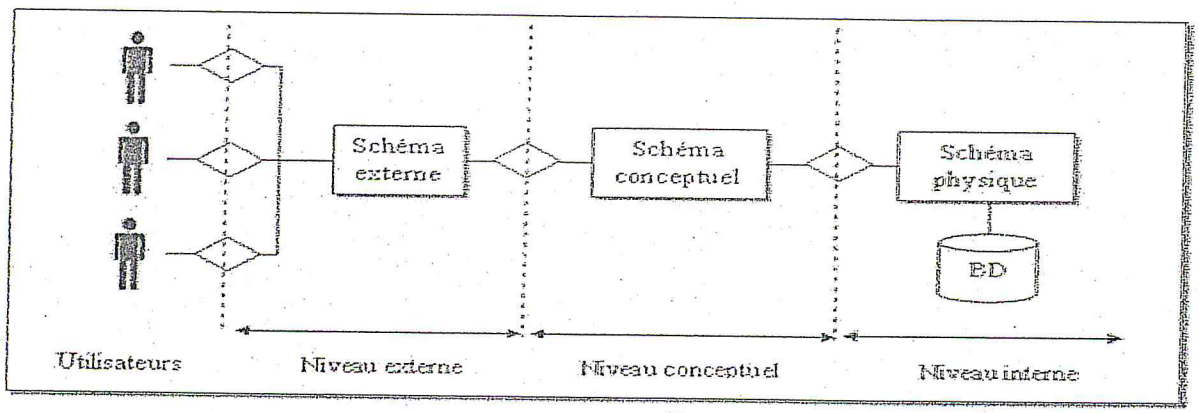


Figure1.1.7 Architecture d'un SGBD réparti [BAL 07]

Classification des approches de conception d'une base de données répartie

La classification de types génériques de SGBDR prend en compte les considérations concernant l'intégration des données. Les SGBDR peuvent être classées en fonction du fait que :

- La BDR est conçue suivant un processus ascendant ou descendant.
- Le schéma global existe ou n'existe pas.
- L'administrateur du schéma global intégré est centralisé ou non

On distingue essentiellement deux approches de conception, l'approche descendante et l'approche ascendante

Approche descendante : Le processus de développement descendant est utilisé lors de la décomposition d'une base de données en plusieurs bases locales ; le schéma local est construit à partir du schéma global (Figure1. 1.8). [RYM 07]

Approche ascendante : dans cette approche (Figure1. 1.9) on intègre des bases de données existantes et le schéma conceptuel global est créé à partir des schémas des sources, c'est dans cette approche que se pose généralement le problème de l'hétérogénéité des sources de données car elles ont été conçues indépendamment et ne découlent pas de la répartition d'une seule base de données. [RYM 07]

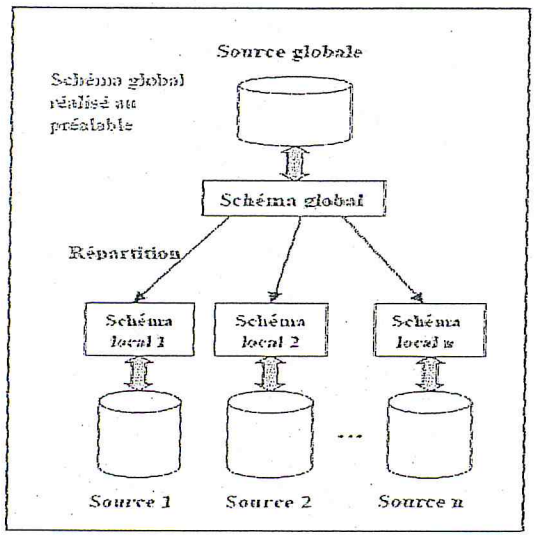


Figure1.1.8.Approche de conception Descendante [RYM 07]

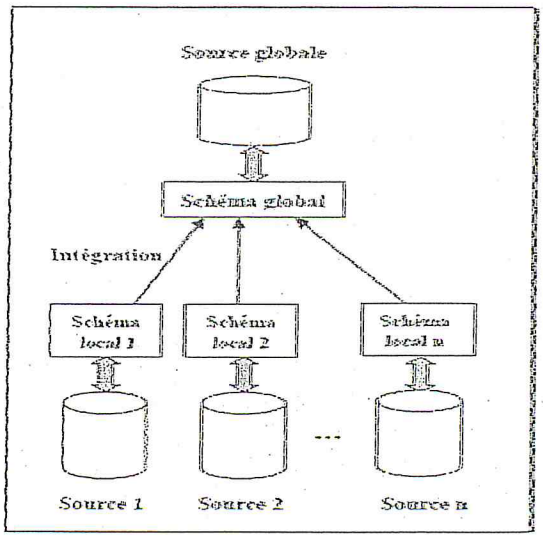


Figure1.1.9 Approche de conception Ascendante[RYM 07]

- **Autonomie**

L'autonomie permet à un système de devenir un composant d'une coopération sans que le partage de l'information ne remette en cause le fonctionnement local du système. [JOU 01]

L'autonomie se manifeste sous plusieurs aspects [BAL 07] :

- *L'autonomie de modélisation*

Qui implique que le site est capable de choisir ses propres modèles et langages d'interrogation de données.

- *L'autonomie de communication*

Qui implique que le système composant est capable de décider quant et comment répondre à une requête externe.

- *L'autonomie d'exécution*

Qui implique qu'un système composant doit être capable d'exécuter des requêtes locales sans interférences avec celles provenant d'un utilisateur externe

- *L'autonomie d'association*

qui implique que le site local est capable de décider de s'associer ou de se dissocier d'une coopération et d'appartenir à une ou plusieurs coopérations.

- **Hétérogénéité**

Les sources de données d'un datawarehouse peuvent présenter plusieurs types d'hétérogénéité:

- *Hétérogénéité des applications*

Les applications des entreprises ne sont pas faites nativement pour travailler ensemble. Ces applications peuvent être déployées dans des systèmes d'exploitation incompatibles (par exemple Windows avec Linux) et développées dans des plate-formes d'exécution propriétaires (.NET, Java, etc.). De plus, certaines entreprises utilisent encore des applications propriétaires et difficilement accessibles ce qui ne facilite pas la collaboration à ce niveau. [TOU07]

- *Hétérogénéité des SGBD*

Résultent dans l'utilisation des différents SGBD

Par exemple faire intégrer un SGBD Sql Server avec PostgreSQL [TOU 07]

- *Hétérogénéité des données*

Différents conflits de données entre sources existent, ces conflits sont le résultat du développement autonome selon des approches et des méthodologies différentes des systèmes d'informations et leurs sources de données, Les problèmes d'hétérogénéité de données sont présentés en détail dans la partie suivante.[TOU 07]

1.8 Problématique de l'intégration de données dans le processus de construction d'un Datawarehouse

Les données du Datawarehouse sont, pour la plupart, issues des différentes sources de données hétérogènes, il existe différentes raisons qui expliquent cette hétérogénéité, par exemple différences dans le matériel, le système logiciel , les systèmes des SGBD, et les données..etc.

Et puisque les données représentent les ressources les plus précieuses disponibles dans les différents systèmes d'informations, nous nous intéressons dans la suite du mémoire aux problèmes d'hétérogénéités des données dans le processus d'extraction, transformation et chargement (ETL).

La classification proposée par [JOU 01] présente les conflits des données en trois niveaux : conflits syntaxiques, conflits structurels et conflits sémantique

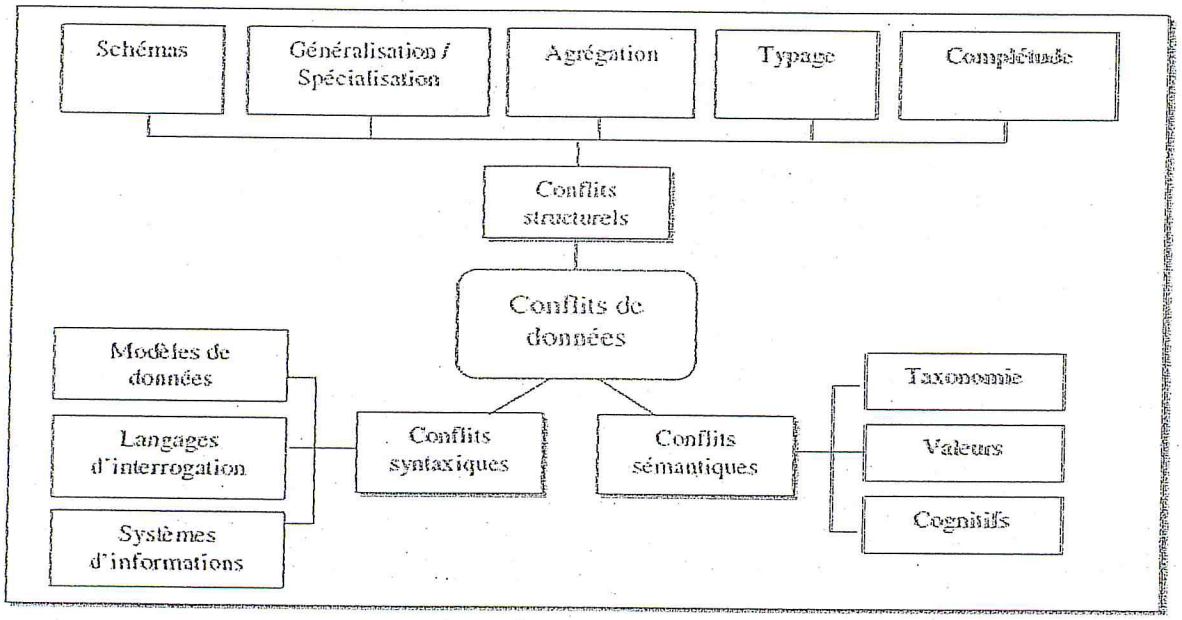


Figure1. 1.10 Classification des conflits de données [BAL 07]

1.8.1 Conflits syntaxiques

Résultent de l'utilisation de modèles de données différents d'un système à l'autre. Des concepts différents sont utilisés pour structurer la même information (relation dans le relationnel, classe dans le modèle objet, balise XML, etc.)[BAL 07]

Exemple Représentation d'une entité « Thèse » dans différents modèles

Schéma relationnel

```
CREATE TABLE [dbo].[thése](
    [Numthese] [varchar](10)NOT NULL
    [titre_t][varchar](20)NOT NULL
    [domaine_t][varchar](30)NOT NULL
```

Schéma XML

```
<!DOCTYPE These [
<!ELEMENT These (Numthese, titre_t, domaine)> (élément imbriqué)
<!ELEMENT Numthese (#PCDATA)> (élément simple)
<!ELEMENT titre_t (#PCDATA)>
<!ELEMENT domaine_t (#PCDATA)>]
```

Schéma orienté objet

```
Public Thése(String Numthese,String titre_t,String domaine)
```

1.8.2 Conflits structurels

Résultent d'une structuration et classification différente des informations. Ils sont étroitement liés aux choix de conception [BAL 07].

Exemple : Représentation de l'information « Téléphone » de manière différente dans le même modèle (Relationnel)

Les numéros de téléphone sont représentés par des attributs dans la relation

- Entreprise : Entreprise (Code, Nom, Adresse, tél1, tél2, tél3, Fax, e-mail)

Les numéros de téléphone sont regroupés dans une relation à part :

- Entreprise (Code, Nom, Adresse, Fax, e-mail)

  Tél_Entreprise (Code, Tél)

➢ **Les conflits de schémas**

résultent de l'utilisation de différents concepts pour représenter le même objet. Une information peut être représentée par une entité dans un système (S1) et par une relation dans un autre système (S2)[BAL 07].

Exemple

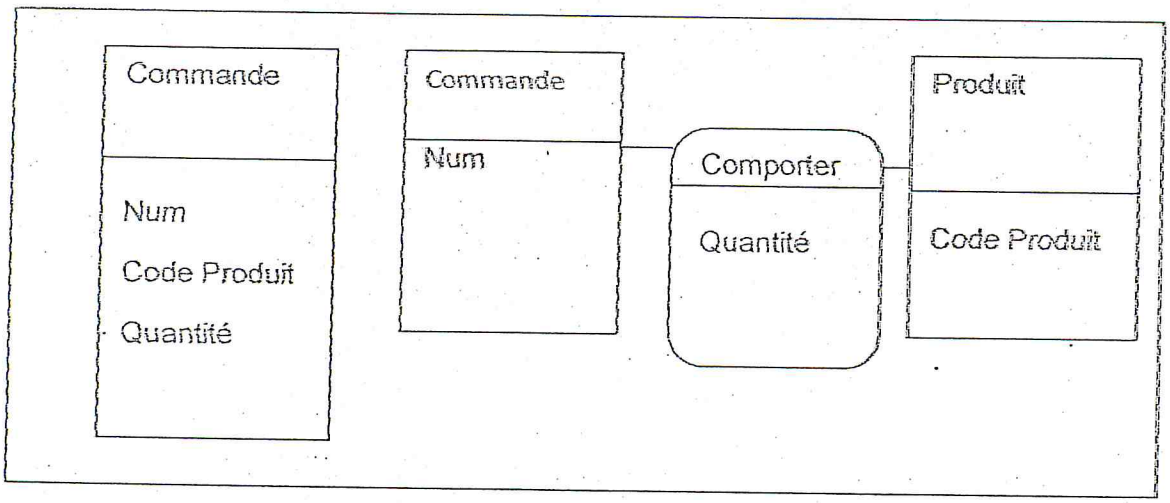


Figure1. 1.11. *Conflits schématiques*

➢ **Les *conflits de généralisation / spécialisation***

Résultent des différences de hiérarchisation des informations[BAL 07].

Exemple : Dans un système (S1) les personnes sont regroupées dans un seul objet PERSONNE alors que dans (S2) on utilise deux objets HOMME et FEMME pour représenter les personnes.

➢ **Les *conflits d'agrégation***

Résultent d'un niveau de granularité différent de deux systèmes. La valeur d'un attribut sur un système correspond à une agrégation des valeurs de plusieurs attributs sur un autre[BAL 07].

Exemple : Nous disposons des moyennes modulaires des étudiants dans un système (S1) et le détail des notes obtenues au cours des examens dans le deuxième (S2)

➢ **Les *conflits de typage***

Résultent des différences de typage pour le même objet dans les différents systèmes de la coopération[BAL 07].

Exemple : Le mois est représenté par une chaine de caractère sur le système S1,est représenté comme un entier sur le système (S2).

➢ **Les conflits de complétude**

Apparaissent lorsque des objets se correspondent partiellement sur les différents systèmes. C'est-à-dire qu'une partie d'un objet du système (S1) trouve une correspondance sur le système (S2) [BAL 07]

Exemple : (S1) : ETUDIANT (matricule, Nom, Prénom, Adresse)

(S2) : STAGIAIRE (N°ins, Nom, Adresse, Ville, Pays)

Dans (S1) l'attribut Adresse contient Rue, Ville et pays alors que dans (S2) l'adresse contient seulement la Rue. Dans (S2) le nom contient à la fois nom et prénom.

1.8.3 Conflits sémantiques

Proviennent des différences d'interprétation des informations partagées entre différents domaines d'application. Plusieurs types de conflits sémantiques apparaissent [TOU 07]

➢ **Les conflits de noms**

Se retrouvent lorsque les différents schémas utilisent des noms différents pour représenter le même concept ou propriété (synonyme), ou des nomsidentiques pour des concepts ou des propriétés différents (homonymes).

➢ **Les conflits de contexte**

Se retrouvent dans le cas où les concepts semblent avoir la même signification dans deux schémas mais sont différents à cause de leur contexte.Par exemple le poids d'une personne dépend de la date où elle se pèse.

➢ **Les conflits de mesure de valeur**

Sont liés à la manière de coder la valeur d'un concept du monde réel dans un système de mesure. Ils se retrouvent par exemple dans le cas où on utilise des unités différentes pour mesurer la valeur d'une même propriété dans différentes sources (dans une source, on utilise le dinar, et dans une autre on utilise l'euro).

Conclusion

Le Datawarehouse permet au décideur de travailler dans un environnement informationnel, référencé, homogène et historié. Cette technique l'affranchit des problèmes liés à l'hétérogénéité des systèmes informatiques.

Nous avons présenté au cours de ce premier chapitre quelques concepts de base sur les datawarehouse et les différents problèmes d'hétérogénéités des données dans le processus d'extraction, transformation et chargement (ETL).

Il reste à définir les différentes approches d'intégrations de données existantes, qui permettent de résoudre les différents conflits de données dans le Datawarehouse

Chapitre II Approches d'intégration de données

Introduction

Cette section présente comment un certain nombre d'approches classées par catégories résolvent les problèmes liés à l'intégration des données, En particulier les conflits de données détectés lors de l'intégration des données dans le Datawarehouse sont, des conflits syntaxiques, schématiques et sémantiques.

Il existe de nombreux travaux qui présentent des approches distinctes pour l'intégration des données, et chaque approche propose des solutions spécifiques à chacun des problèmes évoqués. On considère en général que ces approches appartiennent à deux grandes catégories : l'approche virtuelle et l'approche matérialisée

Approche virtuelle

L'approche virtuelle est apparue pour intégrer les sources de données hétérogènes. Elle propose un schéma de représentation unique qui permet de visualiser l'ensemble des données sources. Cette approche n'est pas utilisé pour le data warehouse, vu que les données du datawarehouse sont matérialisées.

La caractéristique la plus intéressante de cette approche est que les données restent dans leurs sources et elles sont récupérées lors des interrogations. .[GRI 07]

Approche matérialisée

L'approche matérialisée permet d'exporter les données des systèmes sources vers un entrepôt de données. Cette approche est dite matérialisée dans la mesure où les données sont physiquement présentes sur un support à part entière.

Grâce à l'approche matérialisée Le Data Warehouse permet d'organiser l'information différemment pour permettre des analyses transversales. .[GRI 07]

Ces deux approches doivent respecter différents critères

Autonomie des systèmes, extensibilité de l'architecture pour maîtriser l'ajout et le retrait de SI, scalabilité du système face à l'évolution du nombre de participants, composabilité pour combiner les informations de SI différents et transparence d'accès à la localisation et au format des données. [JOU 00]

Ce chapitre permettra de définir les bases de données réparties, les approches d'intégrations des sources de données réparties, déjà existantes, en se focalisant sur l'approche fédérée fortement couplé suivi pour l'intégration de données dans le datawarehouse.

1. Approche multi-bases (approche fédérée faiblement couplée)

1.1 Définition

Les systèmes multi-bases sont des systèmes dits faiblement couplés. On les caractérise de cette manière car ils n'offrent pas une vision unifiée des données. Il n'existe pas de schéma global permettant un accès transparent aux différentes sources de données.

L'objectif de la fédération faiblement couplée est l'intégration des informations des différents systèmes au coup par coup en fonction des besoins. Cette approche repose sur l'utilisation d'un langage d'interrogation et d'un modèle de représentation commun. Chaque SI exporte ses informations sous la forme d'un schéma décrit dans le modèle commun (généralement un modèle orienté objet), le langage multibases (extension de SQL ou OQL) permet une interrogation multisites.

L'interopérabilité des bases de données hétérogènes se fait autour d'un langage de requêtes puissant. La résolution des conflits schématiques et sémantiques reste entièrement à la charge de l'utilisateur. Le langage de requêtes a la particularité d'être doté de constructeurs pour la réconciliation sémantique (résolution des conflits) des données hétérogènes.[BAL 07]

1.2 Critiques

L'approche multi-base présente les avantages suivants :

- Ces systèmes sont extensibles et gardent une grande autonomie.
- Les sources peuvent évoluer de manière indépendante.

Cependant elle présente pas mal d'inconvénients qui ont conduit à l'apparition de nouvelles approches, entre autres on cite :

- La localisation des sources pertinentes reste à la charge de l'utilisateur.
- L'hétérogénéité sémantique et structurelle n'est pas traitée.
- Les correspondances entre les données ne sont pas prédéfinies.

1.3. Exemples de systèmes basés sur l'approche fédérée faiblement couplée

Les projets qui utilisent cette approche sont : Les projets GARLIC de Michael J. Carey , InfoMaster, D.Florescu, Glue, le système IM,Superviews et le système IRO-BD ...etc .

Nous allons présenter dans la suite le projet GARLIC .

Projet GARLIC :

GARLIC ressemble à un système de gestion de bases de données orienté objet. Réellement, GARLIC gère des données dispersées à travers plusieurs sources de données hétérogènes avec des formats variés : données structurées (bases de données), semi-structurées (telles que documents XML) et non structurées (images, textes, etc.). Deux principales caractéristiques de GARLIC : son modèle de données orienté objet GDL (Garlic Data Language) et sa capacité de stocker des objets complexes tels que les documents multimédias.[CAR 95]

- Modèle commun : GDL est une variante du langage ODMG-ODL (Object Data Language)
- Correspondances inter-schémas : ne sont pas pris en charge par GARLIC. l'intégration des informations est exprimée au niveau de la requête.
- Les sources de données locales : données structurées (bases de données) et données complexes (vidéo, image, etc.)
- Langage de requêtes : Puisque le modèle de données de GARLIC est un modèle objet (GDL) et puisque SQL est le langage universel d'interrogation des

bases de données, le choix est porté sur une extension orientée objet de SQL (OQL).

Dans cette solution, seule la résolution des conflits syntaxiques est prise en charge. L'intégration des informations est réalisée dans la formulation d'une requête.

2. Approche fédérée fortement couplée

2.1 Définition

L'objectif de la fédération fortement couplée est la construction d'un *schéma fédéré* qui intègre les différents schémas d'exports des différents systèmes de la coopération. Les mises à jour et requêtes lancées par les utilisateurs sont traitées sur ce schéma fédéré. Cette approche respecte les cinq niveaux proposés par Sheth [SHE 90]. Le schéma fédéré résout les conflits syntaxiques et schématiques grâce à l'utilisation d'un modèle pivot et aux assertions de correspondance inter-schémas.

2.2 Architecture

Pour les systèmes fédérés, Sheth et Larson proposent une architecture en cinq niveaux : schéma local, schéma composant, schéma export, schéma fédéré, schéma externe [SHE 90]

- Les *schémas locaux* : il s'agit des schémas conceptuels initiaux des différentes BD source. Il en existe autant qu'il y a de systèmes sources à intégrer. Ces schémas locaux peuvent être exprimés dans des modèles différents.
- Les *schémas pivots (Composant)*: il s'agit des schémas locaux traduits dans le *modèle commun* (ou *modèle canonique*), c'est-à-dire le modèle utilisé pour la fédération.
- Les *schémas d'export* : ils correspondent à un extrait des schémas pivots. Seuls les éléments que les administrateurs des bases sources souhaitent fédérer sont exportés.
- Les *schémas fédérés* : il s'agit des schémas d'export intégrés. Il peut en exister plusieurs. Ils offrent un vue unifiée des schémas exportés selon le modèle canonique.

- Les *schémas externes* : il s'agit de vues définies pour des groupes d'utilisateurs particuliers du système fédéré (reposant sur le modèle canonique). Ils n'offrent l'accès qu'à un sous-ensemble des données.

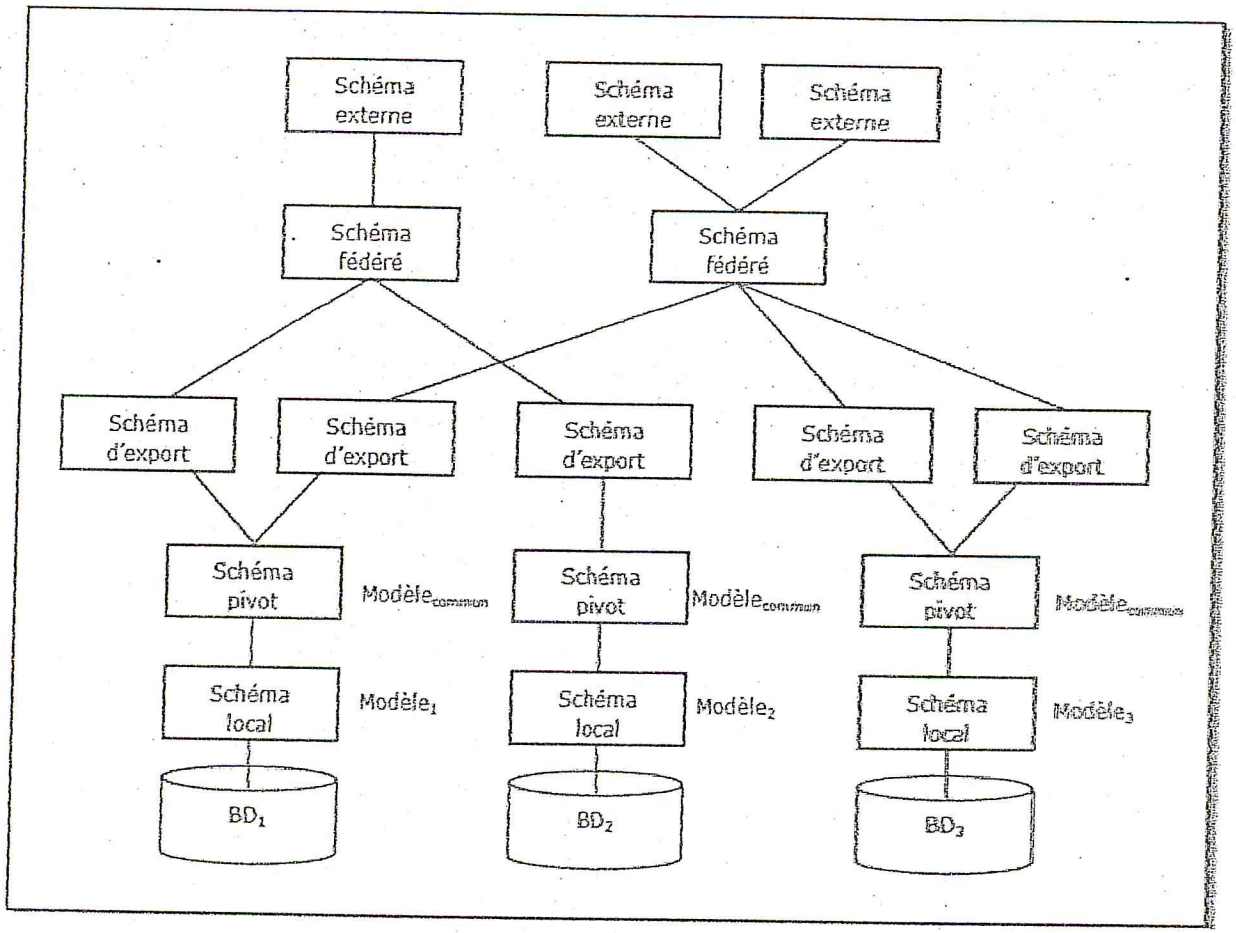


Figure1. 2.1 : Architecture de fédération à cinq niveaux [SHE 90]

2.3 Processus d'intégration dans les bases de données fédérées

L'intégration de schémas permet d'obtenir les schémas fédérés à partir des schémas exports (schéma export/schéma composant). L'intégration des schémas est l'étape clé du processus de fédération [PAR 96].

On distingue trois étapes principales :

- *Traduction de schémas(Pré-intégration)*
- *Recherche des correspondances (Mappings)*
- *Intégration*

Dans ce qui suit , nous expliquons en détail les trois étapes

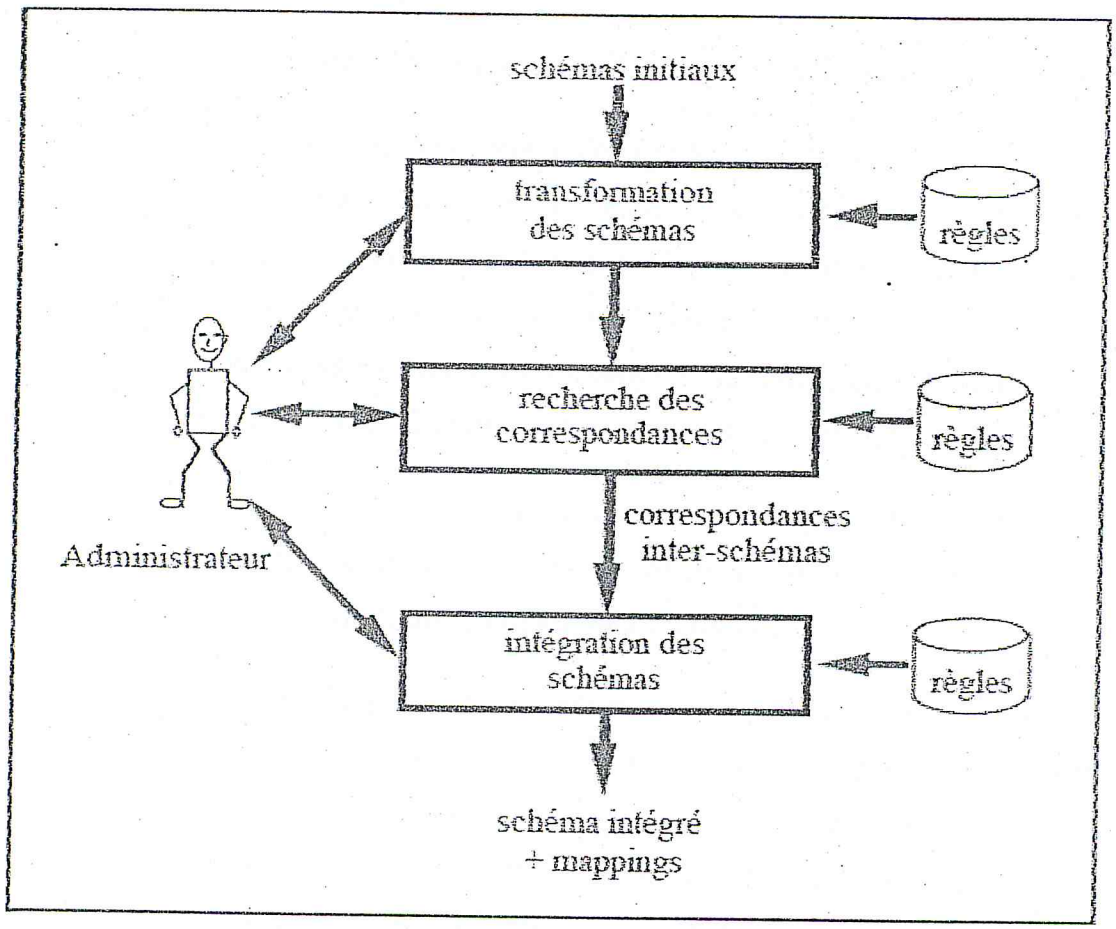


Figure1. 2.2: Le processus global d'intégration [PAR 96]

2.3.1 Traduction de schémas

La traduction de schémas permet d'obtenir les schémas composants en partant des schémas locaux des bases composantes. La traduction de schémas consiste à transformer un schéma décrit dans un modèle source vers un autre schéma décrit dans un modèle commun ou pivot

La traduction de schéma est essentielle puisqu'elle permet de résoudre l'hétérogénéité syntaxique due à une grande variété de modèles utilisés dans les bases de données composantes. [PAR 96]

Les assertions de correspondances

Pour construire le schéma intégré, la relation entre les différents éléments des sources de données, doit être connue, Étant données que les extensions sont considérées comme étant des ensembles d'objets du monde réel , Chaque assertion décrit la relation ensembliste pertinente des deux ensembles :

- équivalence (≡),
- Inclusion (⊆),
- Disjonction (≠)
- Intersection (∩)

Exemple

- on a S2.article ⊆ S1.article puisque tous les articles représentés dans S2 le sont dans S1 et que la réciproque est fausse.
- on a: S1.*Conférence*≡ S2.*Conférence* parce que les deux éléments représentent le même ensemble de conférences dans S1 et S2.

Chaque assertion doit comprendre une clause de spécification de la correspondance entre instances : nous appelons ceci la clause « avec Identifiants Correspondants » (AIC)

Exemple chaque article de S2 correspond à l'article de S1 qui a le même titre

S2.article⊆ S1.article AIC *titre*.

Les représentations des éléments en correspondance possèdent généralement des propriétés communes, au delà de celles utilisées pour leur identification.

Exemple S1.*Conférence* et S2.*Conférence*, partagent, en plus de l'identifiant *ISN*, les propriétés *année* et *N°*. Pour éviter la duplication des propriétés dans le schéma intégré, des assertions décrivent de telles correspondances en utilisant une clause « Avec Attributs Correspondants » (AAC). Ainsi, l'assertion de correspondance interschéma s'écrit:

S1.*Conférence* ≡ S2.*Conférence* AIC *ISN* AAC *année, N°*

2.3.3 Intégration des schémas

Une fois que les correspondances ont été décrites, l'intégration proprement dite peut commencer. Chaque assertion est analysée pour déterminer quelle est la représentation des éléments en correspondance qui doit être incluse dans le schéma intégrer et pour définir quelles sont les règles de traduction entre le schéma intégré et les schémas initiaux. [PAR 96]

La plupart du temps, toutefois, les types en correspondance vont présenter des différences, que ce soit dans leurs représentations ou dans leurs populations. Cette situation est appelée un conflit. Plusieurs taxonomies des conflits on été proposées par [PAR 96]

- Conflit de structure(schéma)
- Conflit de classification (généralisation/Spécialisation)
- Conflit de description
- Conflit d'hétérogénéité
- Conflit de données

On a choisit de focaliser notre travail sur le premier conflit présenté ci-dessus.

Conflit structurel

Un conflit structurel survient lorsque les éléments en correspondance sont décrits par des concepts de niveaux de représentation différents ou soumis à des contraintes différentes.

Exemple : l'assertion suivante montre un conflit structurel

Auteur $\supset$ Article. auteurs AIC nom (Auteur dans S1 est une relation et dans S2 est un attribut)

Solution pour les conflits structurels

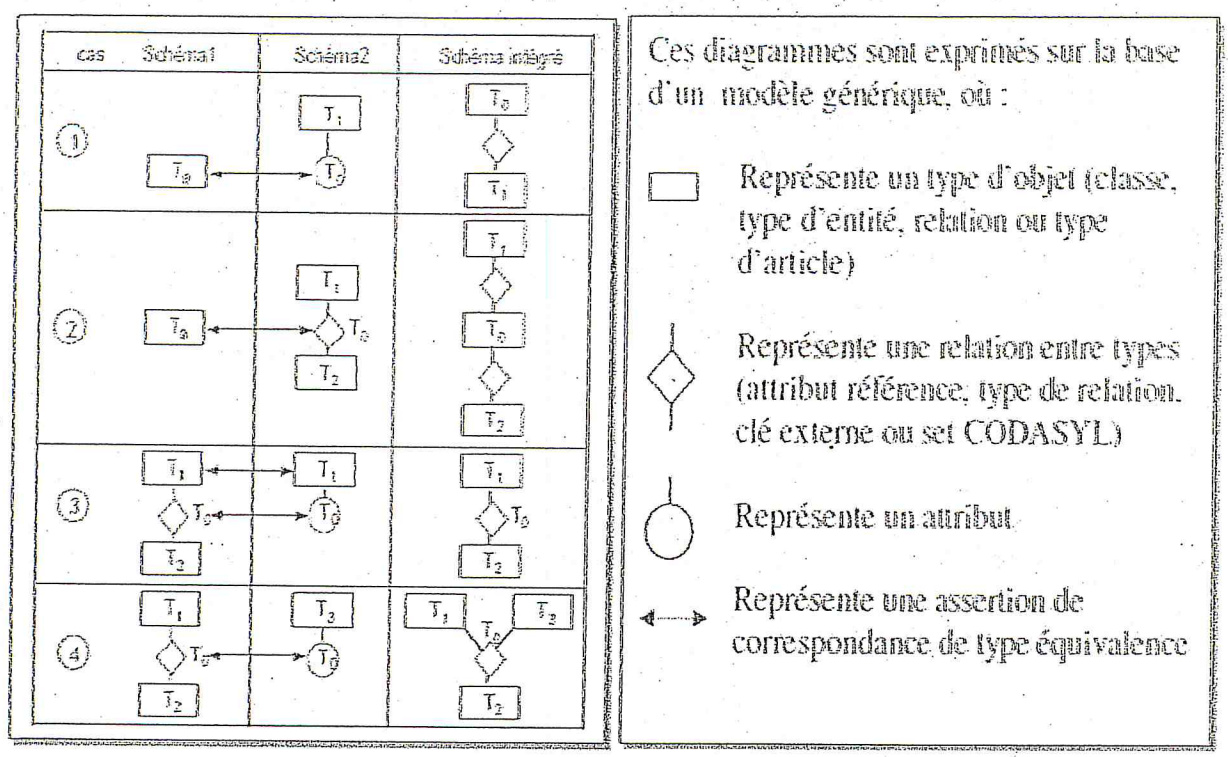


Figure1. 2.3 : Résolution des conflits structurels [PAR 96]

2.4 Critiques

Après l'apparition des approches fédérées fortement couplées, on a pu résoudre plus ou moins les conflits syntaxiques et schématiques par le processus d'intégration, parmi les avantages qu'elles présentent on trouve :

- La transparence de la localisation des données
- Les bases sources gardent leurs autonomies et restent sous contrôle de leurs administrateurs.

Toute fois elle expose l'inconvénient suivant :

- Un changement de schéma dans les bases sources doit se répercuter dans le schéma fédéré, ou bien cette approche n'est pas extensible.

2.5 Exemples de systèmes basés sur l'approche fédérée fortement couplée

Les projets qui utilisent cette approche sont : Le projet MIND (Metu INteroperable DBMS) de Dogac et O.LARAB .

Nous allons présenter dans la suite le projet MIND

Le projet MIND

La solution proposée dans ce projet consiste en une intégration de schémas en quatre niveaux et repose sur une architecture à objets distribués. Les niveaux de Sheth sont repris sauf que le deuxième et troisième niveau (schéma composant et schéma d'export) sont combinés dans un seul niveau qui correspond au schéma d'export.[DOG 98]

- Modèle commun : Le modèle commun utilisé dans ce projet est un langage de définition d'objets (ODL), une extension du langage de définition d'interface (IDL) de l'OMG. Il permet la description des schémas d'export dans un modèle orienté objet.
- Le langage IDL permet de décrire une interface (objet générique de type Base de Données) où chaque instance correspond à une source de données particulière dans la fédération.
- Correspondances inter-schémas : des règles de mise en correspondance dérivées de requêtes OQL relient les éléments du schéma fédéré aux schémas d'export.
- Les sources de données locales sont encapsulées dans des objets CORBA.

  CORBA est une technologie standard de l'OMG. Elle permet à des objets distribués d'interopérer. Ceci permettra d'inclure dans la fédération même les systèmes ne disposant pas de SGBD (sources de données semi-structurées et non structurées)

- Plusieurs types d'agents coopèrent : les agents locaux et les agents globaux.

  Un agent global (Global Database Agent) est chargé de décomposer une requête sur le schéma global en sous requêtes et de gérer les transactions correspondantes pour recomposer dynamiquement les réponses aux sous requêtes locales. L'agent global utilise deux services composés d'agents de transactions et de gestion des requêtes.

Un agent local fait le lien entre le schéma d'export et le schéma local d'un SGBD en traduisant les requêtes canoniques (exprimées dans le modèle commun) en requêtes locales et les résultats locaux en données canoniques.

Le système MIND permet un accès uniforme à n'importe quelle combinaison des sources de données disponibles à travers une requête globale exprimée en langage SQL. La requête globale est décomposée en sous requêtes et soumises à travers le bus ORB (Object Request Broker) vers le Système disposant de la source de données concernée.

3. Approche médiateur

L'approche d'intégration par médiation constitue sans doute aujourd'hui la solution la plus courante pour relier différentes sources.
Les solutions multibases, en particulier les approches fédérées, ont réussi à faire partager des informations appartenant à plusieurs systèmes autonomes, distribués et hétérogènes. Néanmoins certains aspects telles que l'**extensibilité** des solutions, la **transparence** de localisation restent faibles.

L'approche de médiation cherche à fournir, au mécanisme d'intégration des SI, la transparence de l'approche fédérée fortement couplée et la flexibilité de la fédération faiblement couplée.[BAL07]

3.1 Caractéristiques [BAL07]

La médiation repose sur un composant essentiel, appelé médiateur, et un autre composant appelé « wrapper »

Médiateur : Un médiateur est un logiciel qui offre une *interface unique et transparente* à plusieurs bases de données hétérogènes et distribuées. Il est chargé de la localisation des données pertinentes et de la résolution des conflits (structurels et sémantiques) , chargé de répondre à des besoins à partir de connaissances mises à sa disposition.

Pour accéder aux bases de données locales, le médiateur fait appel à des Wrappers.

Wrapper : Un Wrapper de données est un logiciel qui

- Convertit les requêtes exprimées dans le modèle de médiation provenant d'un ou de plusieurs médiateurs vers le modèle des bases de données locales.
- Convertit les données, résultats d'une requête, du modèle des bases de données locales vers le modèle de médiation.
- Un Wrapper de données offre une interface homogène locale d'accès aux données sources (résolution des conflits syntaxiques).

Langage de médiation

Le médiateur adopte un langage unique appelé langage de médiation pour permettre aux différents utilisateurs et applications d'interroger les différentes sources de données de manière uniforme en leur masquant les détails d'hétérogénéité et de localisation.

3.2 Architecture

L'approche de type médiation se présente en trois niveaux :[BAL07]

Niveau système d'information disposant des sources de données locales.

Niveau médiation composé du couple Médiateur/Wrapper, ainsi que la base de définition des connaissances nécessaires à l'interopération des sources de données

Niveau application qui correspond aux applications.

Critiques

- Les systèmes médiateurs offrent une *meilleure extensibilité* et souvent une meilleure *scalabilité*

Cependant elles présentent l'inconvénient majeur suivant :

L'accroissement des sources de données en nombre et en volume pose principalement deux problèmes pour cette approche :

- La complexité de conception du schéma global qui deviendra volumineux pour contenir tous les concepts des sources locales
- la difficulté d'optimisation et de la réécriture des requêtes.

Conclusion

Les systèmes multi-bases de données fournissent aux utilisateurs un langage d'accès multi-bases de données (habituellement de type SQL), les systèmes de bases de données fédérées basent leurs services sur une description intégrée des données qu'ils gèrent. Ceci permet à leurs utilisateurs d'accéder aux données comme sur une base de données centralisée, sans avoir à s'inquiéter de la localisation physique des données accédées ni du type de SGBD qui les gère localement, et cela explique pourquoi chercheurs et entreprises sont très intéressés par l'approche fédérée. Néanmoins l'extensibilité des solutions, reste faible, pour cela d'autres approches ont apparu comme l'approche de médiation, cette dernière promette d'étendre les approches fédérées en apportant davantage de flexibilité , mais cette approche a encore beaucoup de problèmes si le système à intégrer est caractériser par un nombre important de sources . *Il serait utopique de croire qu'il existe une approche idéale. Il faut rechercher l'approche la plus adaptée à son contexte.*

Nous verrons dans ce qui suit que notre étude s'inscrit dans un processus d'intégration destiné à concevoir ***un système fédéré fortement couplée pour intégrer, et matérialiser les données dans le datawarehouse avec une meilleure extensibilité(ajout d'une nouvelle source).***

Partie II
Conception du Système

Introduction

Intégrer des applications les unes avec les autres revient à rendre possible un dialogue entre elles, alors qu'au départ aucune ne parle le même langage. Partant de là, plusieurs approches d'intégrations, s'offrent à l'entreprise qui souhaite se lancer dans un tel chantier. Parmi celles-ci : l'ETL (extraction transformation loading),l'ETL se positionne sur l'intégration de données.

Notre système **ETL Data** *basé sur l'approche fédérée fortement couplé a pour but d'atteindre toutes les fonctionnalités d'un outil ETL, afin d'alimenter un datawarehouse.*

Les outils ETL existant sur le marché permettent ::

- ✓ Extraire les données à partir des sources hétérogènes et répartie,
- ✓ transformer les données afin de les nettoyer, unifier le format de données..etc.
- ✓ charger les données dans le datawarehouse.

Notre prototype doit être capable:

- ✓ *Extraire des données décrites dans des bases de données relationnelles sous la plateforme SQL Server ou PostgreSQL et les documents XML*
- ✓ transformer les schémas afin d'obtenir un schéma fédéré , qu'il permet d'unifier tous les schémas sources de données , et à partir de ce schéma l'administrateur peut construire la table de fait et les tables de dimensions de son datamart.
- ✓ charger les données dans le datawarehouse, grâce aux métadonnées stockées dans le Catalogue_ETL,

Le tableau suivant résume la différence entre Notre Système et les outils ETL du marché.

Notre Système (ETL Data)	ETL Commercial
Extrait les données sources hétérogènes, de deux types de source de données (Structuré, semi structuré)	Extrait les données sources hétérogènes, de plusieurs types de source de données (Structuré, semi structuré, et non structuré)
Le transformateur se charge de préparer les 4 schémas de SHETH	Le transformateur se charge de nettoyer les données, unifier le format de données, filtrer les données..etc
Existence d'un schéma fédéré qui englobe tous les schémas sources de données.	Existence d'une Table de correspondances pour le mapping entre les tables sources de données et les tables datamarts
Résolution des conflits syntaxique est faite dans la phase « Créer le schéma Fédéré »	Résolution des conflits syntaxique est faite par l'intervention humaine, dans la table de correspondance
Si les données retournent plusieurs types de données, le systéme convertit les types à un seul type « varchar»qui permet d'inclure tous les types de données.	Unifier le format de données, dans le cas ou il existe des données synonyme entre les sources
Le schéma Fédéré permet de traiter les jointures entre les bases de données, dans le cas ou les tables de dimensions ou de fait sont construite a partir de plusieurs sources .	Une seul source de données est interrogé lors de l'alimentation, pas de jointure entre les bases de données .

Tableau 2 différence entre Notre Système et les outils ETL du marché

D'après le tableau présenté ci-dessus nous remarquons que notre prototype ajoute une nouvelle fonctionnalité aux outils ETL du marché, il traite les jointures entre les bases sources et cela revient à l'existence d'un schéma fédéré fournit par l'approche fédérée fortement couplée.

La figure suivante montre les trois fonctionnalités d'un outil ETL(Extraction, transformation et chargement) dans notre systéme

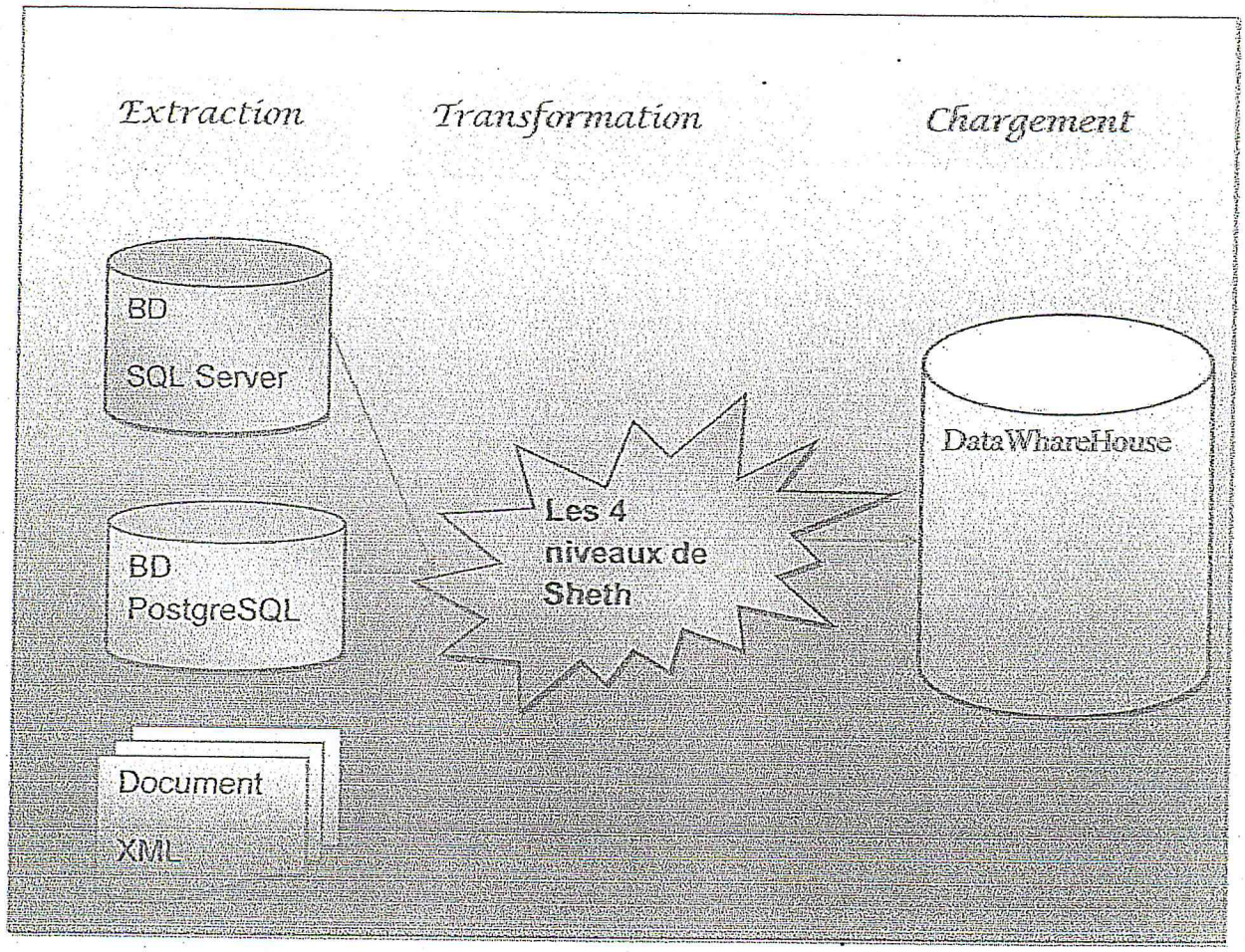


Figure2 Les étapes d'un outil ETL sur notre système

Chapitre 1 Sources de données traitées et le modèle pivot

Introduction

Depuis toujours, trouver un meilleur moyen pour stocker les informations sur un support matériel pour leurs réutilisations futures, était un vrai souci, pour répondre à trois besoins, non exclusifs les uns des autres :

- conserver l'information en lieu sûr pour répondre à une contrainte légale ou conventionnelle (archivage des données).
- rendre l'information disponible .
- réutiliser l'information (traitement des données).

Pour répondre à ce problème beaucoup de recherché ont été faites et qui ont conduit à l'apparition de plusieurs formats de données : les données structurées tel que les bases de données relationnelles, les bases de données objet, les données semi-structurées, les donnée non structurées...ainsi que plusieurs plateformes de gestion et d'analyse de données.

On a choisit de limiter le cadre de notre étude sur le traitement de deux formats de données qui sont considéré comme des données sources qui vont servir à alimenter le datawarehouse.

- ✓ Les bases de données relationnelles comme données structurées sous les deux plateformes :
  - ✓ SGBD SQL SERVER 2005
  - ✓ SGBD PostrgresSQL

- ✓ Les données semi-structurées :
  - ✓ Fichiers XML

Nous détaillons dans ce qui sui :

- ➢ les principes de base de chaque type de données ainsi que les plateformes choisies, déjà cité.
- ➢ Les métas donnés sur les SGBDR.
- ➢ Le model commun choisit dont les schémas sources seront traduit (model pivot).

1. Les sources de données traitées

1.1 Les bases de données

Une base de données relationnelle est une base de données structurée suivant les principes de l'algèbre relationnelle, Le concept permet de stocker et d'organiser une grande quantité d'informations. La théorie est due à Edgar Frank Codd. Elle est mise en œuvre au moyen d'un Système de gestion de base de données relationnelles (SGBDR).

Un système de gestion de base de données (abrégé *SGBDR*) est un ensemble de logiciels qui sert à la manipulation des bases de données. Il sert à effectuer des opérations ordinaires telles que consulter, modifier, construire, organiser, transformer, copier, sauvegarder ou restaurer des bases de données. Il est souvent utilisé par d'autres logiciels ainsi que les administrateurs ou les développeurs. Les trois ténors du marché des SGBD sont IBM DB2, Oracle et Microsoft SQL Server. Nous décrivons par la suite deux SGBDR [W5]

1.1.1 SGBD retenu pour l'extraction de schéma

➢ SGBD SQL SERVER

SQL server est une suite de produits et de technologies qui répondent aux exigeants de stockage des données

SQL Server est un système de gestion de base de données relationnelle (SGBDR) qui :

- Gère le stockage des données pour les transactions et l'analyse ;
- Répond aux requêtes des applications clientes.
- Utilise Transact-SQL pour envoyer des requêtes entre un client et SQL Server

Le Système de gestion de base de données relationnelle de SQL Server est responsable des taches suivantes :

- Gérer les relations entres les données de la base de données ;
- Vérifier que les données sont stockées correctement et que les règles qui définissent les relations entre les données ne sont pas transgressées.
- Récupérer toutes les données à un état de cohérence antérieur en cas de panne du système.

➢ SGBD PostgreSQL

PostgreSQL est un SGBD relationnel objet Open Source implémenté par l'université de Berkeley , il offre un certain nombre de fonctionnalités modernes telles que les requêtes complexes, les clés étrangères, l'intégrité des transactions et la gestion des accès simultanés à la base de données.[W6]

Les systèmes de gestion de base de données relationnels orientés objet proposent des fonctions plus puissantes que les systèmes relationnels classiques.Ces fonctions permettent ainsi une conception logicielle plus complexe.

Les fonctions clés du modèle orienté objet de PostgreSQL :

- Les classes : Une classe correspond à un ensemble d'objets possédant un identificateur unique.
- L'héritage : La notion d'héritage correspond à une organisation hiérarchique des tables.

Par exemple, si deux tables se trouvent dans une relation parent/enfant, les informations contenues dans la table parent sont également disponible dans la table enfant.

- La surcharge : On parle de "surcharge de fonction" lorsqu'une fonction peut être définie plusieurs fois avec des paramètres différents.

1.1.2 Les Métas Données

Chaque SGBD stocke des informations (métadonnées) relatives aux systèmes et aux objets dans les bases de données. Les métadonnées sont des informations relatives aux données.

Les métadonnées données comprennent des informations sur les propriétés des données, telle que le type de données d'une colonne (numérique, texte, etc.) ou la longueur d'une colonne. Il peut également s'agir d'informations sur la structure de données, ou d'informations qui spécifient les conceptions des objets. [MIC 00]

1.1.3 Catalogue de base de données (SQL Server et PostgreSQL)

Chaque base de données contient un ensemble de tables systèmes qui stockent les métadonnées relatives à cette base de données. [MIC 00]

Cet ensemble de tables système constitue le catalogue de la base de données. Il contient la définition de tous les objets de la base de données.

Voici quelques tables systèmes utilisées lors de l'extraction de schéma à partir de SGBD Sql Server, et postgreSQL :

Table système sur PostgreSql	Table systéme sur SQL Server	Fonction
pg_database	Sysdatabases	Contient une ligne pour chaque base de données sur le SGBD
pg_class	Sysobjects	Contient une ligne pour chaque table sur SGBD.
pg_attribute	Syscolumns	Contient les attributs d'une table sur SGBD.
Columns		Contient une ligne pour chaque objet d'une base de données (le type, la longueur, le nom d'une colonne.)
pg_constraint	SysindexeKeys	Fait correspondre à chaque clé étrangère d'une table la clé primaire qu'elle référencie dans une autre table.
data_type	Systypes	Contient les types des colonnes.
key_column_usage	Sysindexes	Contient les contraintes de clés primaires et étrangères pour chaque table.
Table		Contient le nom et le schéma de chaque table.

Tableau2.1.1 Table système de postgreSQL et SQL Server

1.2 Les données semi-structurées

Les données semi structurées sont les données qui possède une structure flexible et qui n'ont pas un schéma à priori mais plutôt dont le schéma peut être extrait à partir de la données. La plupart du temps, un ensemble de données semi structurées est représenté sous la forme d'un graphe dont les feuilles contiennent les données et dont les nœuds et les liens représentent la structure de l'ensemble .[KEZ 07]

La modification, l'ajout ou la suppression d'une donnée entraine une modification du graphe, c'est-à-dire la structure de l'ensemble.

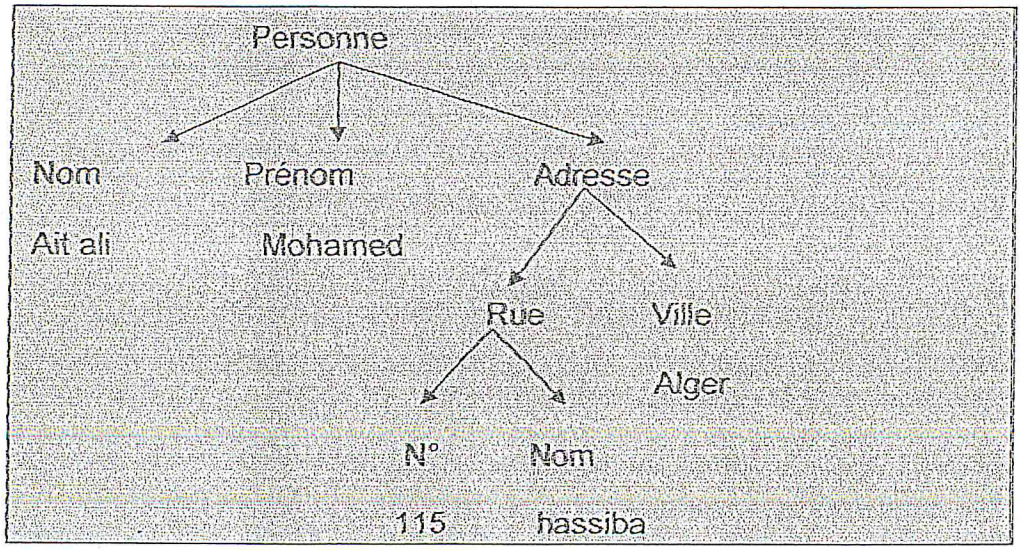


Figure2.1.1 Exemple de graphe de données semi-structuré

1.2.1 Le Model de données XML

XML (eXtended Markup Language) est un format textuel extensible de description de document définit par le W3C (World Wide Web Consortium). De la famille des langages de marquage SGML (Standard Geberalizerd Markup Language) définit par ISO (ISO 8879), il permet de s'adapter à quasiment tous les domaines ou l'on a besoin de structurer de l'information de façon portable. [KEZ 07]

C'est un format de données universel, qui permet de gérer toutes les structures

» Fichiers plats

» Tables relationnelles

» Arbres

» Réseaux

» Objets

1.2.1.1 La syntaxe d'un document XML

Un fichier XML est présenté sous la forme d'un simple fichier texte contenant des balises de type <balise> , son composant principal est l'élément « Element ».

Un document XML est composé d'éléments complexes (en contenant d'autres) et d'éléments simples contenant du texte, les éléments sont délimités par des balises ouvrantes et fermantes et peuvent être précisés par des attributs. [HES 05]

```
<personne>
    <nom> Ait Ali </nom>
    <prenom>Mohamed</prenom>
    <Adresse
        <Ville> Alger </ville>
        <rue>
            <N°>115</N°>
            <Nom> Hassiba</Nom>
        </rue>
    </Adresse>
```

Figure2. 1 .2 Exemple d'un fichier XML

La déclaration d'entête d'un document XML a la forme suivante :

```
< ? xml version = "1.0" encoding ="ISO-8859-1" standlone=yes ?>
```

Figure2.1.3 Exemple d'entête d'un fichier XML

Cette expression indique au processus qui va traiter le document : La version du langage XML utilisé dans le document, le codage des caractères utilisés dans le document , si le document XML fait référence à une DTD ou à un schéma xml définit dans un autre fichier, il faut mettre "no". Sinon, on utilise "yes".

1.2.1.2 Les qualifications d'un fichier XML

Un document XML peut avoir deux qualifications, il peut être :

- ✓ ***Bien formé*** : quand il respecte la syntaxe du langage XML définie par le W3C
- ✓ ***Valide*** : quand il est associé à une définition de type de document et qu'il la respecte (nom des éléments, type, répétition et ordre d'apparition dans le document).

Un document XML bien formé est un document XML qui respecte certaines règles simples :

- Il existe un et un seul élément racine qui contient tous les autres éléments.
- Les balises sont correctement imbriquées : chaque balise ouvrante a une balise fermante associée.
- Le nom des balises est libre mais il contient au moins une lettre.
- Les attributs des balises, lorsqu'ils existent, doivent comporter obligatoirement une valeur qui doit toujours apparaître entre double apostrophes.
- Quand un élément est vide, les balises peuvent être simplifiées :

<balise></balise> est identique à <balise/>.

1.2.2 Les métas données pour le model XML [DON 03]

Les données semi structurées sont autos descriptives, le schéma est définit dans les données, et aucune structure n'est précisée a priori. Ceci permet une grande flexibilité dans le traitement (requêtes, stockage, chargement) et aussi dans les mises à jour et le changement structurels de telles données

En générale les grammaires définissent la syntaxe d'un langage. En XML, elle définit une liste de balises qui sont utilisables, la syntaxe correspond à l'organisation de ces balises et elle représente un méta donné pour les données semi structurées.

Afin de décrire les données semi structurés sans utiliser un schéma prédéfini rigide, deux approches ont été proposées

- ***Un schéma déduit:*** est calculé automatiquement à posteriori à partir des données. Ce schéma permet de décrire les structures de données d'une manière assez précise, et sans mis à jour régulière. A titre d'exemple on cite les guides de données (Data Guides).

- ***Un formalisme de schéma flexible :*** définit à priori, il permet de décrire les données d'une manière ouverte, pour pallier aux structures irrégulière des données. Un tel formalisme nous permet aussi bien de décrire un schéma rigide et typé qu'un schéma n'imposant aucune structuration dans les données. On cite l'exemple des schémas XML et DTDs.

1.2.2.1 Les DTD [W3]

Le W3C propose une définition de document type appelée DTD (Document Type Definition), c'est-à-dire une grammaire permettant de vérifier la conformité du document XML.

– Une DTD est un fichier permettant de vérifier qu'un document XML est conforme à une structure donnée.

– La norme XML n'impose pas l'utilisation d'une DTD pour un document XML, mais elle impose par contre le respect exact des règles de base de la norme XML.

Cependant, les DTD, comme langage de schéma utilisé pour spécifier le contenu et la structure de documents XML, souffrent de quelques déficiences :

- Les DTD ne sont pas écrites en XML, ce qui signifie que les technologies existantes pour manipuler des documents XML telles que DOM ou SAX ne peuvent être utilisées pour « parser » des schémas de documents.
- Les DTD n'offrent qu'un typage très limité des données.

```
<!ELEMENT Etudiant (Id_emp, Nom, Prénom, Adresse)>
<! ELEMENT Id_Etud (#PCDATA)>
<!ELEMENT Nom (#PCDATA)>
<!ELEMENT Prénom (#PCDATA)>
<!ELEMENT Adresse (#PCDATA)>
```

Figure2.1.4 Exemple de définition d'une DTD

1.2.2.2 Les Schémas XML [W3]

Conscient de ces grandes limitations des DTD, le W3C a proposé un nouveau langage de définition de schéma de documents qu'est ***XML Schema***.

Conçu pour palier aux déficiences précitées des DTD, XML Schema propose, en plus des fonctionnalités fournies par les DTD, plusieurs nouveautés à savoir :

- Un grand nombre de types de données intégrées comme les booléens, les entiers, les intervalles de temps, etc. De plus, il est possible de créer de nouveaux types par ajout de contraintes sur un type existant.
- Des types de données utilisateurs qui nous permettent de créer notre propre type de données nommé.

- La notion d'héritage : Les éléments peuvent hériter du contenu et des attributs d'un autre élément. C'est sans aucun doute l'innovation la plus intéressante de XML Schéma, car elle permet la réutilisation.
- Les indicateurs d'occurrences des éléments peuvent être tout nombre non négatif.
- Une grande facilité de conception modulaire de schémas.

Le document XML Schéma fournit deux types de définition de type de données :

A. Type simple

Types simples incluent les types de base inclus dans la bibliothèque standard des XML Schéma, les listes et les unions.

Les types de données simples les plus courants sont représentés par : chaînes de caractères, entiers, dates, réels, etc. Les types de données des DTD ont été repris par XML Schéma pour des raisons de compatibilité ascendante.

Les types de données simples ne permettent pas aux éléments de contenir des sous-éléments. Pour cela, XML Schéma définit le type de données complexes, pouvant être créé par un auteur de schéma.

B. Type Complexe

Un type de données complexe peut être caractérisé par son modèle de contenu ; c'est-à-dire comment ses sous-éléments sont susceptibles d'apparaître. Un type complexe est défini à l'aide de l'élément <xsd:complexType name="..."> qui pourra contenir, entre autres, une séquence d'éléments, une série d'attributs, etc.

XML Schéma propose un ensemble de connecteurs permettant de représenter n'importe quel modèle de contenu exprimable à l'aide d'une DTD :

- Le connecteur de séquence :

Le connecteur de séquence permet de définir un type complexe comportant des suites d'éléments. Le connecteur séquence définit une liste ordonnée de sous éléments. Pour pouvoir rendre des éléments optionnels, il faut appliquer des contraintes d'occurrences sur les éléments.

- Le connecteur de choix

Le connecteur de choix permet de définir un type complexe comportant des éléments variables. Il n'autorise qu'un seul de ses fils soit présent dans le document instance. Pour pouvoir autoriser l'un ou l'autre ou les deux, il faut appliquer des contraintes d'occurrences sur l'élément « xsd: choice.».

- L'élément all

L'élément all représente un connecteur supplémentaire par rapport aux DTD. Il permet à ses éléments fils d'apparaître une fois (ou pas du tout) et dans n'importe quel ordre.

```
<xsd:schema>
        <xsd:complexType name="Etudiant">
                <xsd:sequence>
                <xsd:attribute name="Nom" type="xsd:string"/>
                <xsd:attribute name="Prénom" type="xsd:string"/>
                <xsd:attribute name="Adresse" type="xsd:string"/>
                <xsd:attribute name="Matricule" type="xsd:positiveInteger>
            </xsd:sequence>
        </xsd:complexType>
</xsd:schema>
```

Figure2.1.5 Exemple d'un schéma xml (ficher xsd)

Toute fois, il est possible d'exprimer les contraintes des clés primaires et clés étrangères sur les attributs dans un schéma xml.

Lors de l'extraction d'un schéma xml, nous intéressons plus particulièrement aux contraintes de type Primary Key(PK), Foreign Key (FK):

La déclaration d'une clé primaire dans une structure XML est définit comme suit :

```
<xsd:key name="IDdepartement">
    <xsd:selector xpath=".//departement"/>
    <xsd:field xpath="./@idDep"/>
</xsd:key>
```

Figure2.1.6 Exemple d'une clé primaire

Ou : - IDdepartement est le nom de la Primary Key
- departement est le nom de la table contenant cette clé.
- idDep est le nom de la clé primaire dans la table departement

La déclaration d'une clé étrangère dans une structure XML est définit comme suit :

```
<xsd:keyref name="cle_site" refer="num_site">
    <xsd:selector xpath=".//page"/>
    <xsd:field xpath="./cle_site"/>
</xsd:keyref>
```

Figure2.1.7 Exemple d'une clé étrangère

Ou : - cle_site est le nom de la colonne foreign Key.
- Num_site : est le nom de la colonne Primary Key.
- Page : la table ou clé_site est définit comme Foreign Key

Remarque : la définition des contrainte PK et FK d'un ou plusieurs éléments ne peuvent pas être définit que dans le schéma de l'élément qui contient ses éléments.

1.2.3 Traduction d'un schéma XML en un schéma relationnel

La traduction d'un schéma xml en un schéma relationnel se fait comme suit :

- Toute élément <xsd :ComplexType> sera représenté soit par une base de données si le schéma xml est composé d'au moins un élément <xsd :ComplexType> , sinon il sera représenté comme une table dans le modèle relationnel.
- Tout élément <xsd:attribute name="Nom" type="xsd:string"/> est représenté par une colonne :
- name : définit le nom de la colonne.
- type : définit le type de la colonne. Afin de présenter les types primitifs xml dans le modèle relationnel, on doit définir une conversion de type sur chaque type xml.

Nous présentons ci-dessous une traduction en modèle relationnel des deux modèles de schémas xml qu'on a traité dans notre processus d'intégration , la première concerne un fichier xml simple ou aucune contrainte n'a été exprimer sur ses attributs, et la 2ème concerne un fichier xml ou des contraintes de primary key et foreign key ont été exprimées :

Pour le 1ier cas nous reprenons le schéma qui à été présenté dans la Figure2.1.5, sa traduction en une base de données relationnelle sera comme suit :

On a un seul élément <xsd:complexType>, donc il sera traduit en une table qui aura comme nom Etudiant. On obtiendra le schéma relationnel suivant :

Etudiant (Matricule,Nom,Prénom,Adresse) ou :

Matricule est de type « int »

Nom est de type « Varchar »

Prénom est de type « Varchar »

Adresse est de type « Varchar »

2ème Cas :

Soit le fichier XSD présenté ci-dessous ou :

(3) Représente l'élément de type « type_annuaire » qui contient deux éléments :

page : qui est de type « type_page » présenté par (1)

Site : qui est de type « type_site » présenté par (2)

(4) Représente la définition des contraintes PK de l'élément page dans l'élément de type « type_annuaire ».

(5) Représente la définition des contraintes PK de l'élément site dans l'élément de type « type_annuaire».

(6) Représente la définition des contraintes FK.

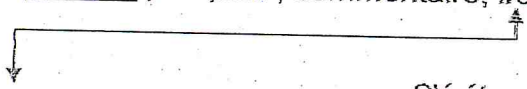
```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsd:schema>
 <xsd:complexType name="type_page">
        <xsd:sequence>
                <xsd:attribute name="titre" type="xsd:string"/>
                <xsd:attribute name="lien" type="xsd:string"/>
                <xsd:attribute name="commentaire" type="xsd:string"/>
                <xsd:attribute name="cle_site" type="xsd:positiveInteger"/>
                <xsd:attribute name="cle_page " type="xsd:positiveInteger" use="required"/>
                <xsd:attribute name="langue" type="xsd:string" use="required"/>
        </xsd:sequence>
</xsd:complexType>
 <xsd:complexType name="type_site">
        <xsd:sequence>
                <xsd:attribute name="nom" type="xsd:string"/>
                <xsd:attribute name="url" type="xsd:string"/>
                <xsd:attribute name="numero" type="xsd:positiveInteger" use="required"/>
        </xsd:sequence>
 </xsd:complexType>
<xsd:complexType name="type_annuaire">
 <xsd:sequence>
   <xsd:element name="page"
                maxOccurs="unbounded"
                type="type_page">
    <xsd: key name="cle_page">
    <xsd:selector xpath=".//page"/>
    <xsd:field xpath="./@cle_page"/>
    </xsd:Key>
   </xsd:element>
   <xsd:element name="site"
                maxOccurs="unbounded"
                type="type_site">
    <xsd:key name="numero ">
    <xsd:selector xpath=".//site"/>
    <xsd:field xpath="./@numero"/>
    </xsd:key>

    <xsd:keyref name="cle_site" refer="numero">

    <xsd:selector xpath=".//page"/>

    <xsd:field xpath="./cle_site"/>

    </xsd:keyref>

   </xsd:element>

 </xsd:sequence>

</xsd:complexType>

 <xsd:element name="annuaire" type="type_annuaire"/>

</xsd:schema>
```


Figure2.1.8 Modèle de schéma XML(XSD)

En Obtiendra dans le modèle relationnel le schéma suivant :

Base de données : annuaire qui sera composé de ces tables :

page (numero ,titre,lien , commentaire, #cle_site ,langue)



Clé étrangère

site (numero , url, type)

2. Le modèle de données pivot (modèle relationnel)

Le modèle relationnel a été formalisé par CODD en 1970. Quelques exemples de réalisation en sont : DB2(IBM), INFORMIX, INGRES, ORACLE, SQL Server .....etc. Dans ce modèle, les données sont stockées dans des tables, sans préjuger de la façon dont les informations sont stockées dans la machine. Un ensemble de données sera donc modélisé par un ensemble de tables.

Le succès du modèle relationnel auprès des chercheurs, concepteurs et utilisateurs est dû à la puissance et à la simplicité de ses concepts. En outre, contrairement à certains autres modèles, il repose sur des bases théoriques solides, notamment la théorie des ensembles et la logique mathématique (théorie des prédicats d'ordre 1). [W5]

2.1 Les objectifs du modèle relationnel

- proposer des schémas de données faciles à utiliser.
- améliorer l'indépendance logique et physique.
- Mettre à la disposition des utilisateurs des langages de haut niveau pouvant éventuellement être utilisés par des non informaticiens.
- optimiser les accès à la base de données.
- améliorer l'intégrité et la confidentialité.
- fournir une approche méthodologique dans la construction des schémas.

De façon informelle, on peut définir le modèle relationnel de la manière suivante [W5]

- Les données sont organisées sous forme de tables à deux dimensions, encore appelées relations et chaque ligne n-uplet ou tuple,
- les données sont manipulées par des opérateurs de l'algèbre relationnelle,
- l'état cohérent de la base est défini par un ensemble de contraintes d'intégrité.

2.2 Le langage des requêtes SQL

L'utilisation du langage SQL (sturctured Query language) permet de normaliser le développement des applications de base de données dans les entreprises SQL est un langage évolué de manipulation de base de données relationnelle, il opère sur des ensembles logiques de données appelés des relations

SQL comprend un ensemble de commandes permettant de définir, de mettre à jour et d'afficher les informations des tables.

SQL a été développé chez IBM dans le milieu des années 70 adopté des sa parution par de nombreuses sociétés, il est considéré aujourd'hui comme une norme dans les environnements de grandes et moyennes entreprises.[MIC 00]

Conclusion

Le modèle relationnel a fait ses preuves et il a démontré, surtout depuis les années 80, qu'il avait des points forts indiscutables.

Le modèle relationnel est complètement adapté aux applications de gestion grâce à son système de modélisation formé de tables dont les colonnes prennent des valeurs alphanumériques. Les concepts du modèle relationnel (tables, enregistrements, relations, champs, etc...) sont simples et aisément compréhensibles par n'importe qui. De plus, la longue expérience dont bénéficient maintenant les SGBD relationnels a permis de les optimiser et donc de les rendre très performants.

Un autre point fort du modèle relationnel est l'existence de SQL, un langage standardisé de création et de manipulation des bases de données. Le langage SQL est bien adapté aux architectures client-serveur, il intègre la gestion des transactions, indispensable lors d'accès concurrents aux données ainsi que pour leur sécurité.

Après cette étude détaillée des sources de données retenues pour alimenter le datawarehouse , et le modèle de données relationnel , nous présenterons dans la partie suivante la modélisation du système ETL Data.

Chapitre 2 Modélisation d'ETL Data

Introduction

L'objectif d'un système d'intégration est de fournir une interface qui permet d'accéder de manière unifiée et transparente à différentes sources de données .Ces sources peuvent être hétérogènes et réparties sur un réseau informatique.

ETL Data offre une solution d'intégration de données dans le cadre de construction d'entrepôts de données pour les systèmes décisionnels, elle permet d'intégrer des bases de données hétérogènes structurées (base de données relationnelles), semi-structurées (XML) basée sur l'approche fédérée fortement couplée.

Pour ce faire, notre solution propose un processus qui se réalise de la manière suivante :

- Extraire les schémas des différents sources de données distribuées, hétérogènes et les traduire vers le modèle commun le modèle relationnel.
- Définir les schémas exports des différentes sources de données .
- Identifier les correspondances entre les différents schémas exports, cette étape elle a pour but de construire un outil qui effectue automatiquement l'intégration .et qui permette de résoudre les conflits schématique entre les base de productions.
- Après l'intégration des schémas (construction du schéma fédéré), l'administrateur devra cibler, à partir du schéma fédéré, les données jugées pertinentes pour des applications d'analyse (schéma externe). Celles-ci seront donc retenues pour la construction du datamart (*magasin de données*).

Dans ce chapitre nous allons présenter l'architecture de la solution proposés .Ainsi nous expliquons les étapes de la conception par les différents diagrammes d'UML.

1. Architecture du système d'intégration proposé

Notre approche s'inspire des différentes architectures citées dans la PARTIE I. Elle est basée sur l'approche fédérée fortement couplé.

Dans notre architecture les données sont extraites à partir des bases de données relationnelles (sous la plateforme SQL Server et PostgreSQL)et données semi-structurées (documents XML) , ce choix est dû à l'utilisation fréquente de ces deux types de représentation de données.

La Figure suivante illustre les différents niveaux de la solution proposée , les niveaux de Sheth sont repris sauf que le premier et deuxième niveau (schéma local et schéma pivot) sont regroupés dans un seul niveau qui correspond au schéma pivot.

Le schémas pivot, correspond au schéma d'une source de données traduit dans le modèle pivot qui est le modèle relationnel.

Le schémas export, représente le schéma de données pertinent retenues pour l'analyse.

Le schéma fédéré, c'est le schéma global obtenu après l'intégration des schémas exports.

Le schéma externe , représente les tables de dimensions et table de fait,qui serviront à l'alimentation d'un datamart.

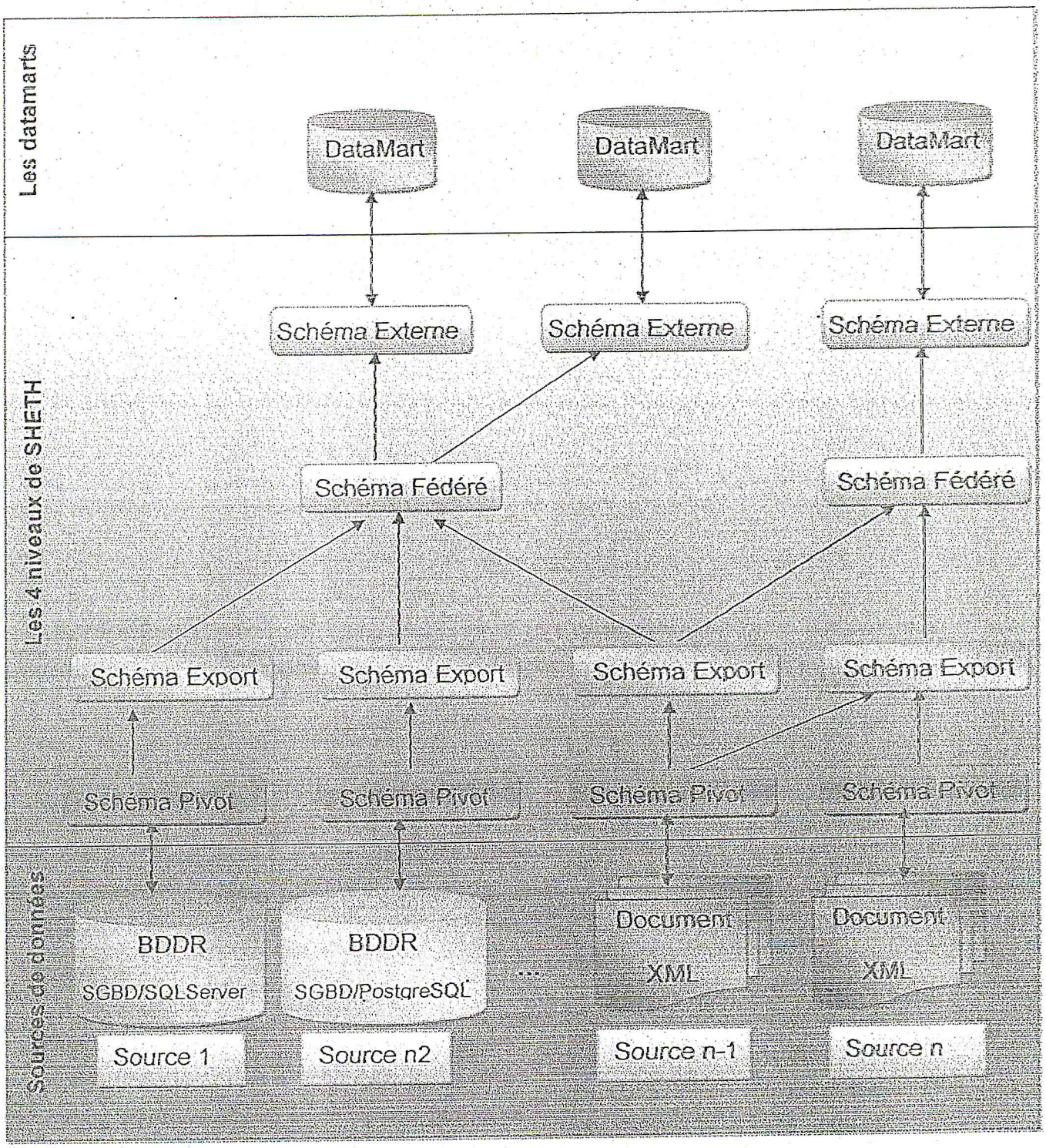


Figure2.2.1 Architecture d'un système d'intégration de données

Diagramme de classes

Le diagramme de classes est généralement considéré comme le plus important dans un développement orienté objet. Il représente l'architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs liens, que ceux-ci représentent un emboîtage conceptuel (héritage) ou une relation organique (agrégation).

Diagramme d'objets

Le diagramme d'objets permet d'éclairer un diagramme de classes en l'illustrant par des exemples. Il est, par exemple, utilisé pour vérifier l'adéquation d'un diagramme de classes à différents cas possibles.

Diagrammes de composants

Les diagrammes de composants décrivent les composants et leurs dépendances dans l'environnement de réalisation. En général, ils ne sont utilisés que pour des systèmes complexes.

Diagrammes de déploiement

Les diagrammes de déploiement montrent la disposition physique des différents matériels (les nœuds) qui entrent dans la composition d'un système et la répartition des instances de composants, processus et objets qui « vivent » sur ces matériels.

Les diagrammes de déploiement sont très utiles pour modéliser l'architecture physique d'un système.

2.1.2.2 Les vues dynamiques

Diagramme d'états-transitions

Le diagramme d'états-transitions représente la façon dont évoluent les objets appartenant à une même classe. La modélisation du cycle de vie est essentielle pour représenter et mettre en forme la dynamique du système.

2. Modélisation du système «ETL Data»

2.1Le langage de modélisation UML

2.1.1Définition

UML est un langage de modélisation (Unified Modeling Language), dont la mise au point est supervisée par un consortium de plusieurs entreprises (essentiellement américaines) : l'Object Management Group (OMG).[UML05]

À l'origine, UML est le résultat de la fusion de trois méthodes objets préexistantes :

- OMT de James Rumbaugh ;
- OOSE de Ivar Jacobson ;
- BOOCH de Grady Booch.

UML n'est pas une méthode de conception mais notation graphique normalisée de présentation de certains concepts pour modéliser des systèmes objets. En particulier, UML ne précise pas dans quel ordre et comment concevoir les différents diagrammes qu'il défini. Cependant, UML est indépendant de toute méthode de conception et peut être utilisé avec n'importe lequel de ces processus.

2.1.2 Diagrammes UML

Il existe 2 types de vues du système qui comporte chacune ses propres diagrammes :

2.1.2.1 Les vues statiques

Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système. C'est le premier diagramme du modèle UML, celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre.

Diagramme d'activités

Le diagramme d'activités montre l'enchaînement des activités qui concourent au processus.

Diagramme de séquence

Le diagramme de séquence représente la succession chronologique des opérations réalisées par un acteur. Il indique les objets que l'acteur va manipuler et les opérations qui font passer d'un objet à l'autre.

Diagrammes de collaboration

Un diagramme de collaboration fait apparaître les interactions entre des objets et les messages qu'ils échangent.

Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation présenté au dessus permet de recenser les grandes fonctionnalités fournit par notre système, et montre l'interaction entre l'administrateur d'un DWH et les principaux fonctionnalités offertes par notre système.

2.2 Diagramme des cas d'utilisations

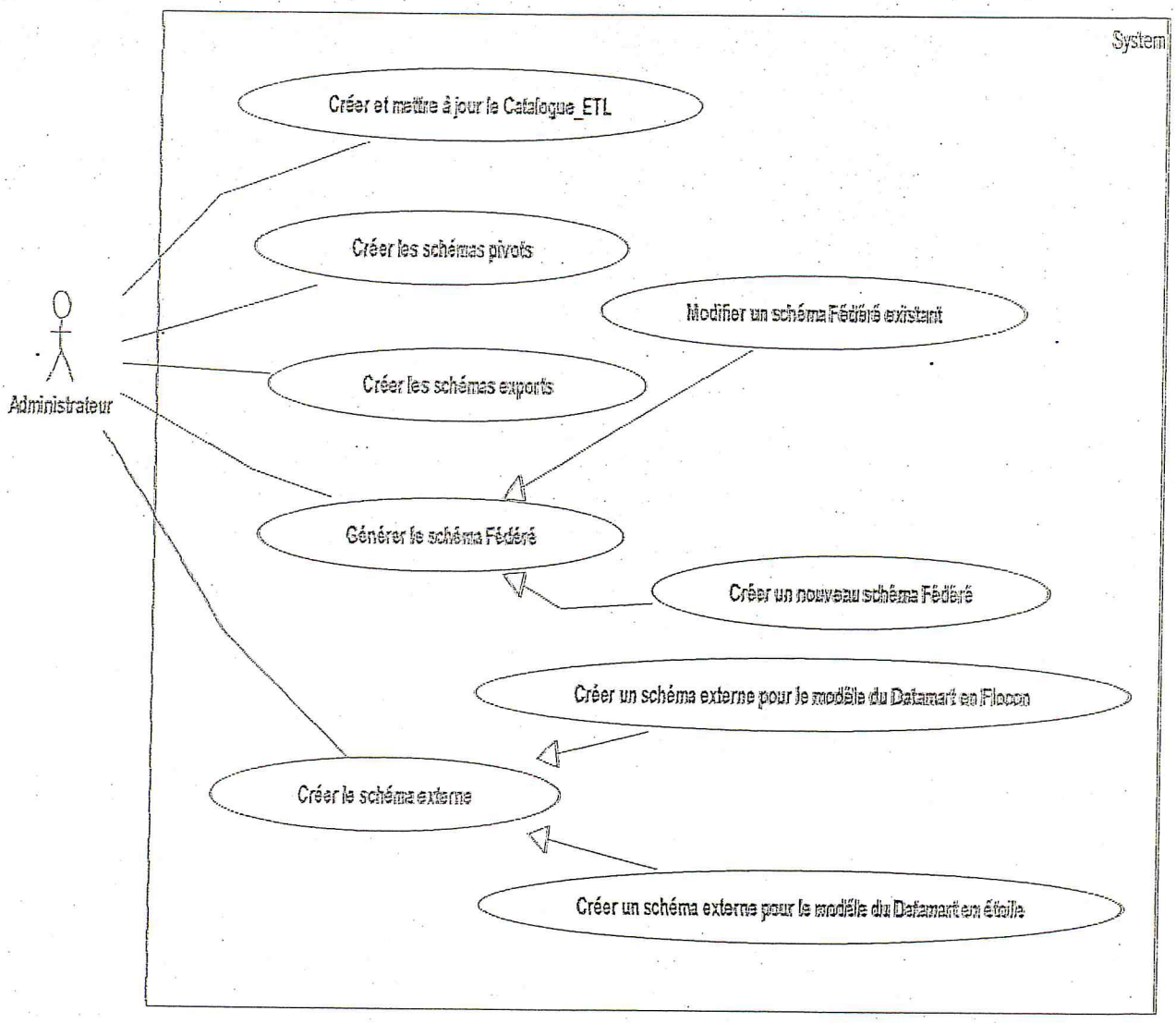


Figure2.2.2 Diagramme de cas d'utilisation «Intégration des bases de données hétérogènes dans le processus de construction d'un datawarehouse.

Description des acteurs

Nom de l'acteur	Rôle
Administrateur du DataWreHouse	*Acteur principale qui prend en charge toutes les phases du processus d'intégration de l'approche fédéré pour le but d'alimenter les différents datamarts*

Tableau 2.2.1 Description de l'acteur

Nous détaillons par la suite chaque fonctionnalité, dans un package.

Créer et mettre à jour le Catalogue_ETL

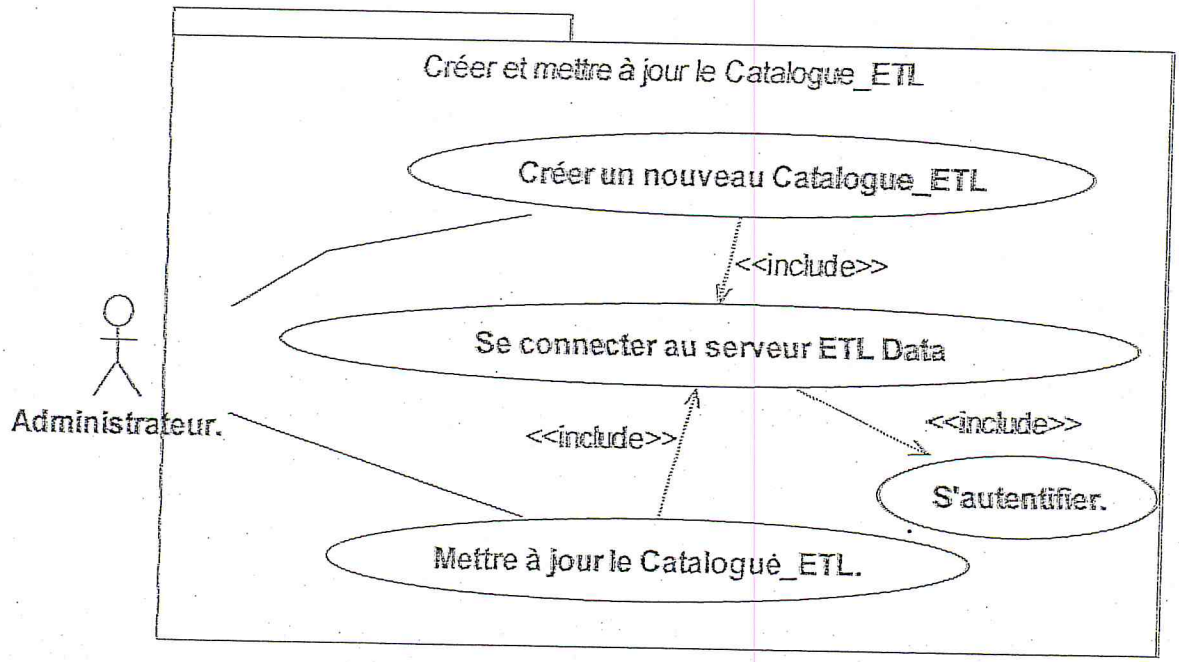


Figure2.2.3 Diagramme de cas d'utilisation « Créer et mettre à jour le Catalogue_ETL»

Description textuelle des cas d'utilisation

Cas d'utilisation	Description
Créer le Catalogue_ETL	L'administrateur doit se connecter au SGBD SQL Server pour qu'il puisse créer le Catalogue-ETL. dans cette étape l'administrateur doit définir, l'adresse du serveur, le nom d'utilisateur, le mot de passe et le port pour qu'il puisse accéder au Serveur ETL Data.
mettre à jour le Catalogue_ETL	Cette étape permette l'ajout d'une source dans le Catalogue_ETL, lors de la création d'un nouveau datamart. Après avoir se connecter au Serveur ETL Data.

Tableau 2.2.2 Description textuelle « Créer et mettre à jour le Catalogue_ETL »

2.2.1 Diagramme de cas d'utilisation «Créer les schémas pivots »

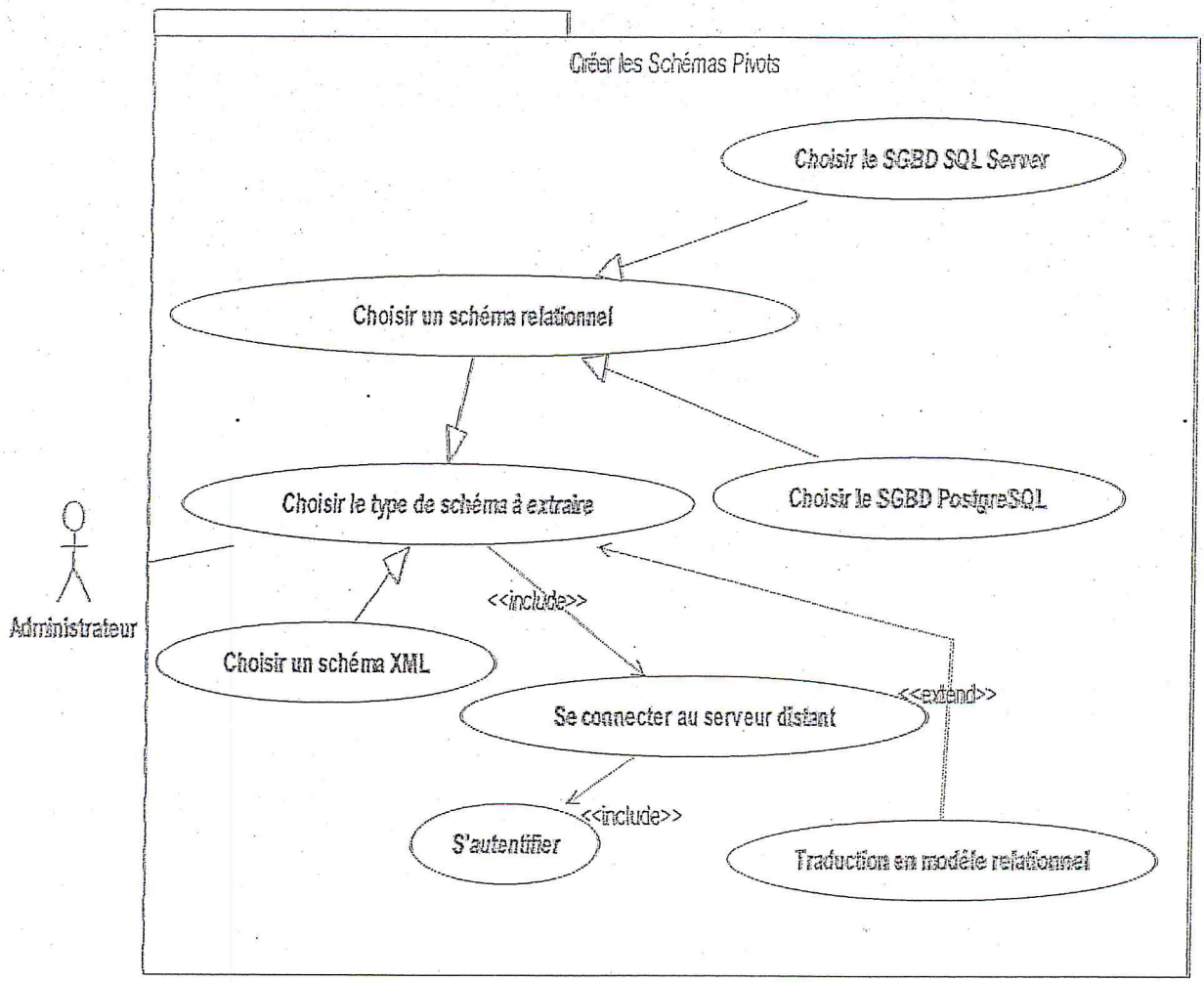


Figure2.2.4 Diagramme de cas d'utilisation «Créer les schémas pivots»

Description textuelle des cas d'utilisation

Cas d'utilisation	Description
Choisir le type de schéma à extraire	*Notre système permet d'extraire deux types de schémas : les schémas pour les fichiers de type XML et les schémas pour les fichiers de type base de données relationnelles sur les deux plateformes SQL Server et PostgreSQL.* Dès que l'administrateur choisit le schéma pertinent pour l'analyse, le système résout les conflits syntaxiques, en traduisant le schéma de la source vers *le modèle relationnel.*
Choisir un schéma relationnel	L'administrateur doit définir, l'adresse du serveur distant ou local (pour les plateforme SQLServer , PostgreSQL), le *nom d'utilisateur, le mot de passe et le* port pour qu'il puisse. extraire le fichier pertinent pour l'analyse.
Choisir un schéma XML	*Si le type de fichier à extraire est de type XML, l'administrateur doit définir le* chemin de fichier, pour qu'il puisse. extraire le fichier pertinent pour l'analyse .

Tableau 2.2.3 *Description textuelle « Créer les schémas pivots»*

2.2.2 Diagramme de cas d'utilisation «Créer les schémas Exports »

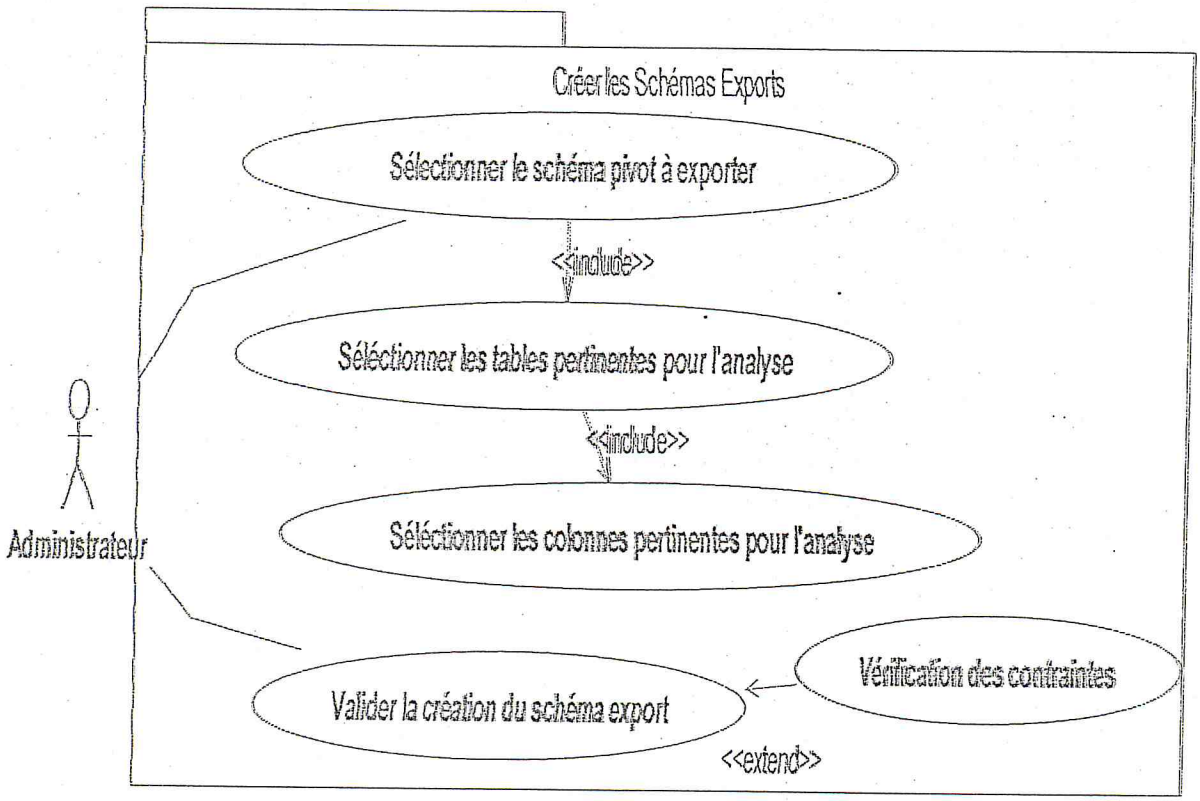


Figure2.2.5 Diagramme de cas d'utilisation «Créer les schémas exports»

Description textuelle des cas d'utilisation

Cas d'utilisation	Description
Sélectionner le schéma pivot à exporter	Afin de créer un nouveau schéma export , l'administrateur doit sélectionner le schéma pivot ,table pivot et pour chaque table les colonnes pivots
Valider la création du schéma export	Après avoir exporté le schéma pivot, le système déclenche une vérification de l'existence d'une colonne de type Primary Key pour chaque colonne de type foreign Key, déjà prise.

Tableau 2.2.4 Description textuelle « Créer les schémas exports»

2.2.3 Générer un schéma Fédéré

Ce cas d'utilisation englobe deux cas d'utilisation principaux, qui seront représentés dans les deux packages suivants.

2.2.3.1 Créer un nouveau schéma Fédéré

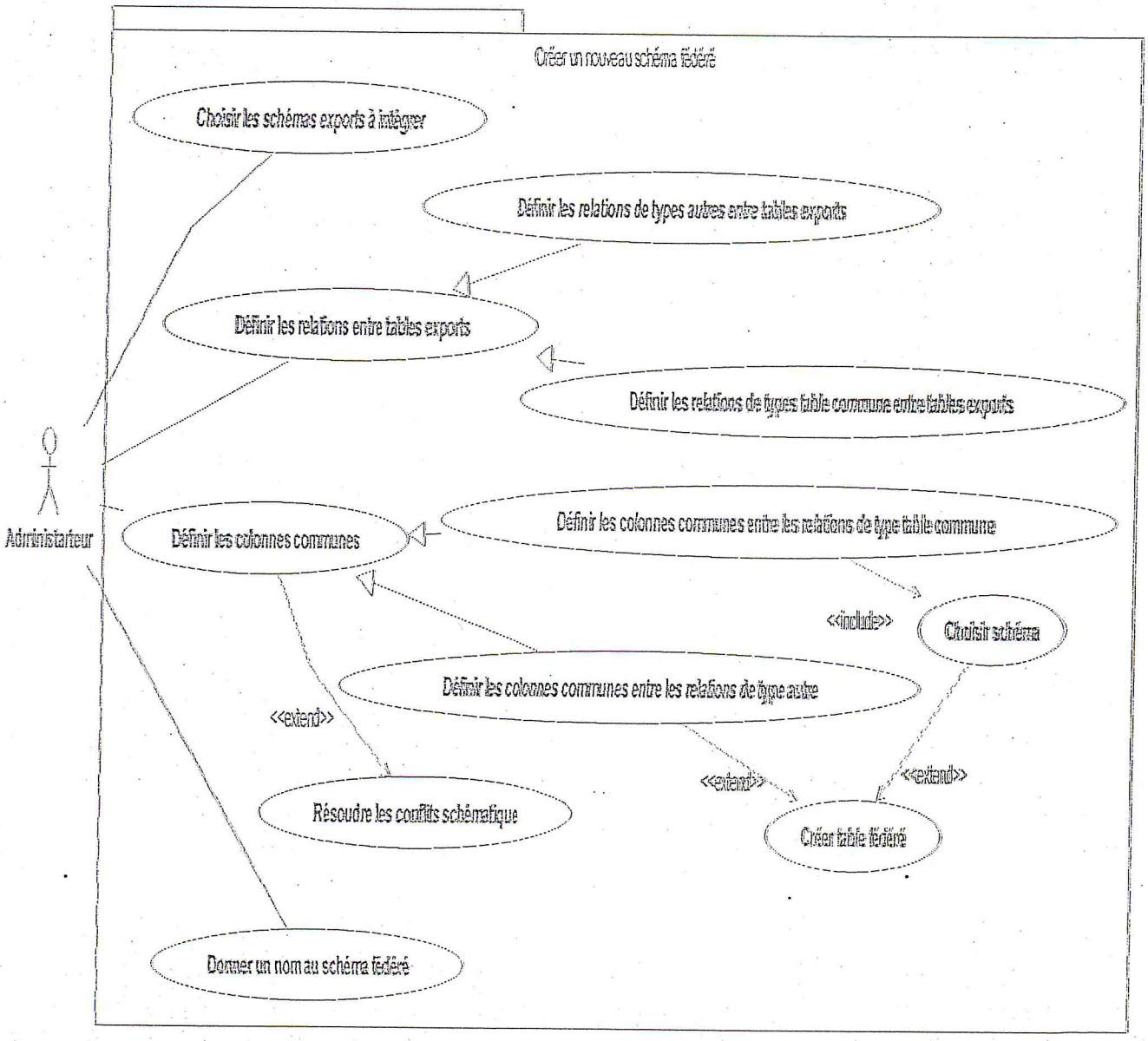


Figure2.2.6 Diagramme de cas d'utilisation «Créer un nouveau schéma fédéré»

Description textuelle des cas d'utilisation

Cas d'utilisation	Description
Donner un nom au schéma Fédéré	Dans cette étape l'administrateur donne un nom au schéma Fédéré qu'il souhaite créer.
Choisir les schémas exports à intégrer	L'administrateur doit choisir au minimum un schéma export à intégrer.
Définir les relations de type « table commune » entre les tables exports	L'administrateur doit définir les tables communes entre les schémas exports choisis.
Définir les relations de type « autre» entre les tables exports	L'administrateur doit définir les cardinalités de type 1..*,1..* ou 1..*,1entre les tables des schémas exports.
Définir les colonnes communes entre les tables exports ayant une relation déjà définit de type « table commune »	Pour résoudre les conflits schématiques présents dans les schémas exports, l'administrateur fusionne les schémas des tables reconnu comme « table commune » en éliminant les colonnes en double.
Définir les colonnes communes entre les tables exports ayant une relation déjà définit de type « autre »	Pour résoudre les conflits schématiques présents dans les schémas exports, l'administrateur doit préciser les cardinalités entre les tables export, ce qui va permettre de déterminer les relations entre les tables du schéma fédéré.

Tableau 2.2.5 Description textuelle « Créer un nouveau schéma Fédéré»

Table Commune : Représente une relation existante entre les tables exports pour dire qu'on parle pratiquement de la même représentation du même objet dans le monde réel. Dans les schémas qui les regroupe

Autre : représente une relation existante entre deux tables de deux schémas exports , qui sera précisé par des cardinalités de type 1..*,1..* ou 1..*,1.

2.2.3.2 Modifier un schéma Fédéré existant

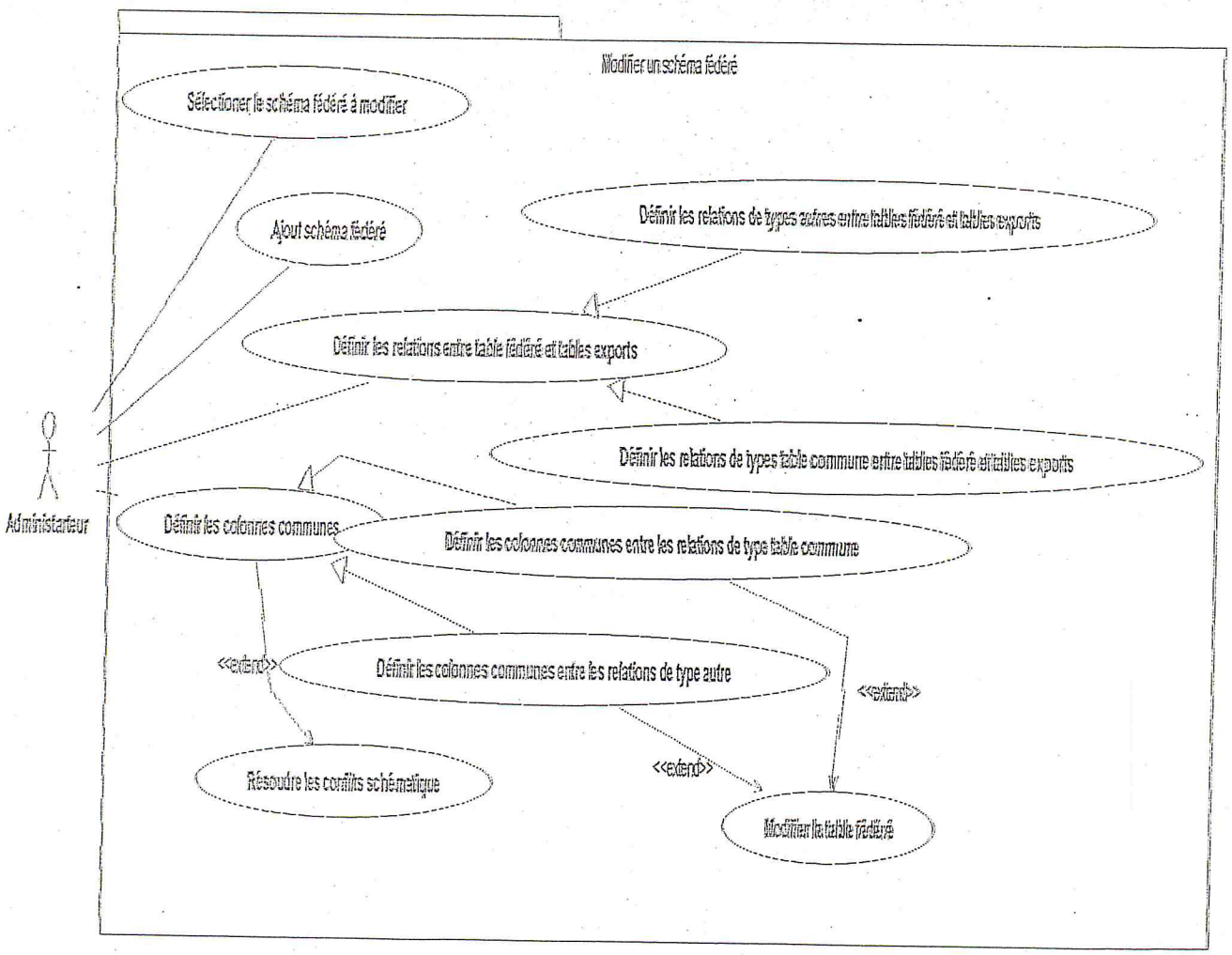


Figure2.2.7 Diagramme de cas d'utilisation «Modifier un schéma fédéré existant»

Description textuelle des cas d'utilisation

Cas d'utilisation	Description
Sélectionner le schéma fédéré à modifier	Dans cette étape l'administrateur choisit un schéma fédéré auquel il veut exprimer de nouveaux besoins.
Ajout des schémas exports	Afin de modifier un schéma fédéré l'administrateur doit sélectionner les nouveaux schémas exports qu'il veut intégrer.
Définir les relations de type « table commune » entre les tables exports et table fédéré	L'administrateur doit définir les tables communes entre les schémas exports choisis et les tables du schéma fédéré à modifier.
Définir les relations de type « autre» entre les tables exports et les tables fédéré	d tiod ruetartsinimda'Léfinir les cardinalités de type 1..*,1..* ou 1..*,1entre les tables des schémas exports choisis et les tables du schéma fédéré à modifier.
Définir les colonnes communes entre une tables export et une table fédéré ayant une relation déjà définit de type « table commune »	Pour résoudre les conflits schématiques entre les tables exports et les tables fédérées existantes déjà, l'administrateur va modifier la structure de la table fédérée, en ajoutant les colonnes non communes.
Définir les colonnes communes entre une tables export et une table fédéré ayant une relation déjà définit de type «autre»	Pour résoudre les conflits schématiques entres les tables exports et les tables fédérées existantes déjà, l'administrateur doit préciser les cardinalités entres les tables exports et les tables fédérées.

Tableau 2.2.6 Description textuelle « Modifier un schéma Fédéré existant»

2.2.4 Diagramme de cas d'utilisation «Créer un Schéma Externe»

Ce cas d'utilisation englobe deux cas d'utilisation principaux, qui seront représentés dans les deux packages suivants.

2.2.4.1 Diagramme de cas d'utilisation «Créer un Schéma Externe pour le modèle en Etoile»

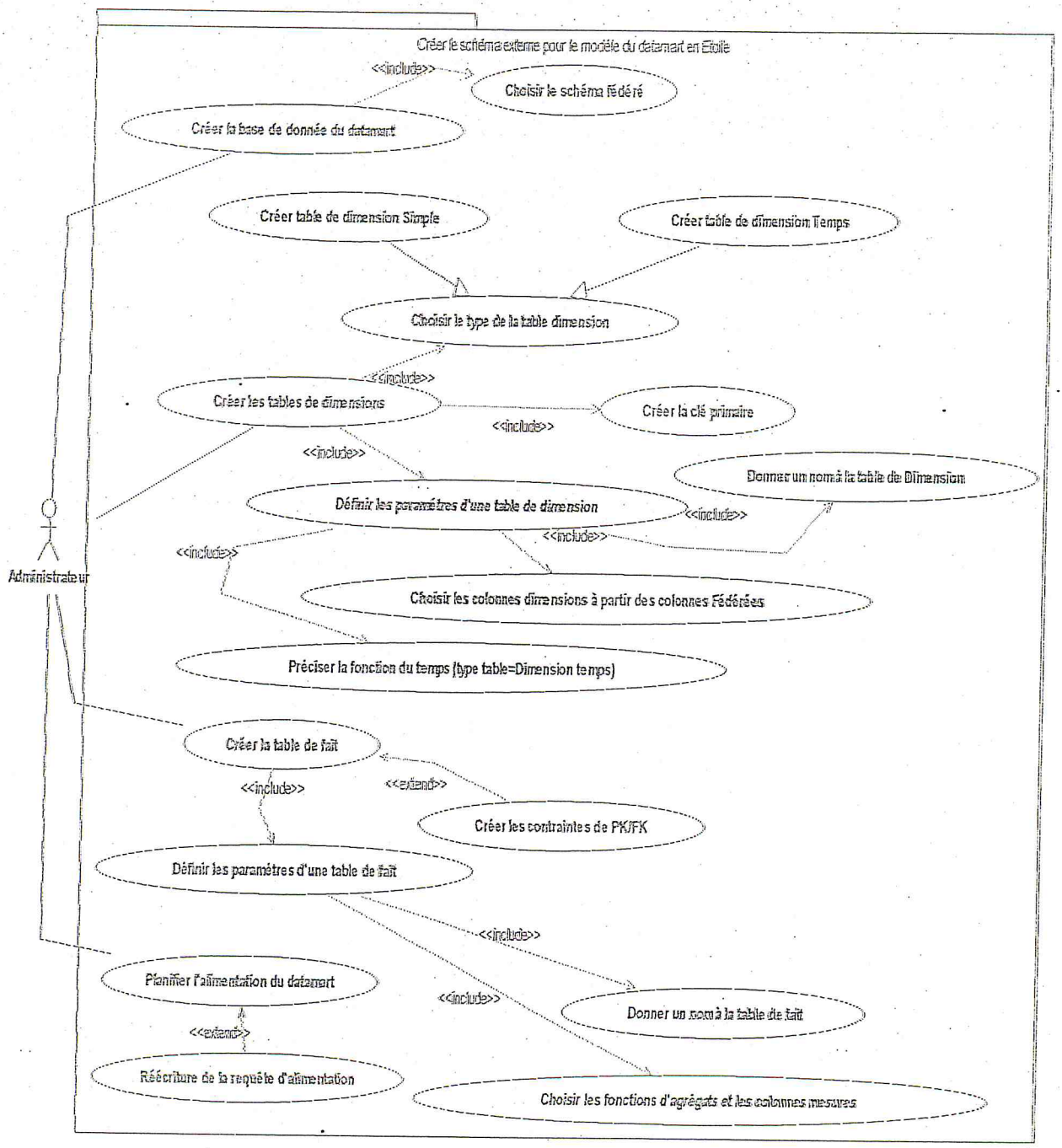


Figure2.2.8 Diagramme de cas d'utilisation «Créer le schéma externe pour le modèle du datamart en étoile»

Description textuelle des cas d'utilisation

Cas d'utilisation	Description
Créer la base de données du datamart	Pour créer la base de données datamart , l'administrateur doit choisir le schéma fédéré .
Créer les tables de dimensions	Pour créer la table de dimension, l'administrateur doit : • choisir entre la création d'une table de dimension simple et une table de dimension temps • après le chois de la table de dimension il donne un nom à la table de dimension, et il sélectionne à partir des colonnes fédérées, les colonnes de dimensions. Et si le type de la table dimension est de type dimension temps, l'administrateur doit choisir la fonction du temps, pour les colonnes de type date • définir la clé primaire d'une table de dimension
Créer la table de fait	Pour créer la table de fait, l'administrateur doit : • Donner un nom à la table de fait • Choisir les colonnes de mesure de fait à partir des colonnes fédérées, pour chaque mesure , il doit spécifier la fonction d'agrégat. Dans cette étape, le système crée la contrainte PK/FK pour la table de fait (il fusionne les clés primaire des tables de dimensions)

Planifier l'alimentation du datamart	Pour alimenter le datamart l'administrateur, doit planifier le système, pour rafraichies les données périodiquement dans le datamart.cette tache déclenche la réécriture de la requête d'alimentation pour insérer les données qui sont dispatchées sur des bases de productions.

Tableau 2.2.7 «Créer le schéma externe pour le modèle du datamart en étoile»

2.2.4.1 Diagramme de cas d'utilisation «Créer un Schéma Externe pour le modèle Flocon de neige»

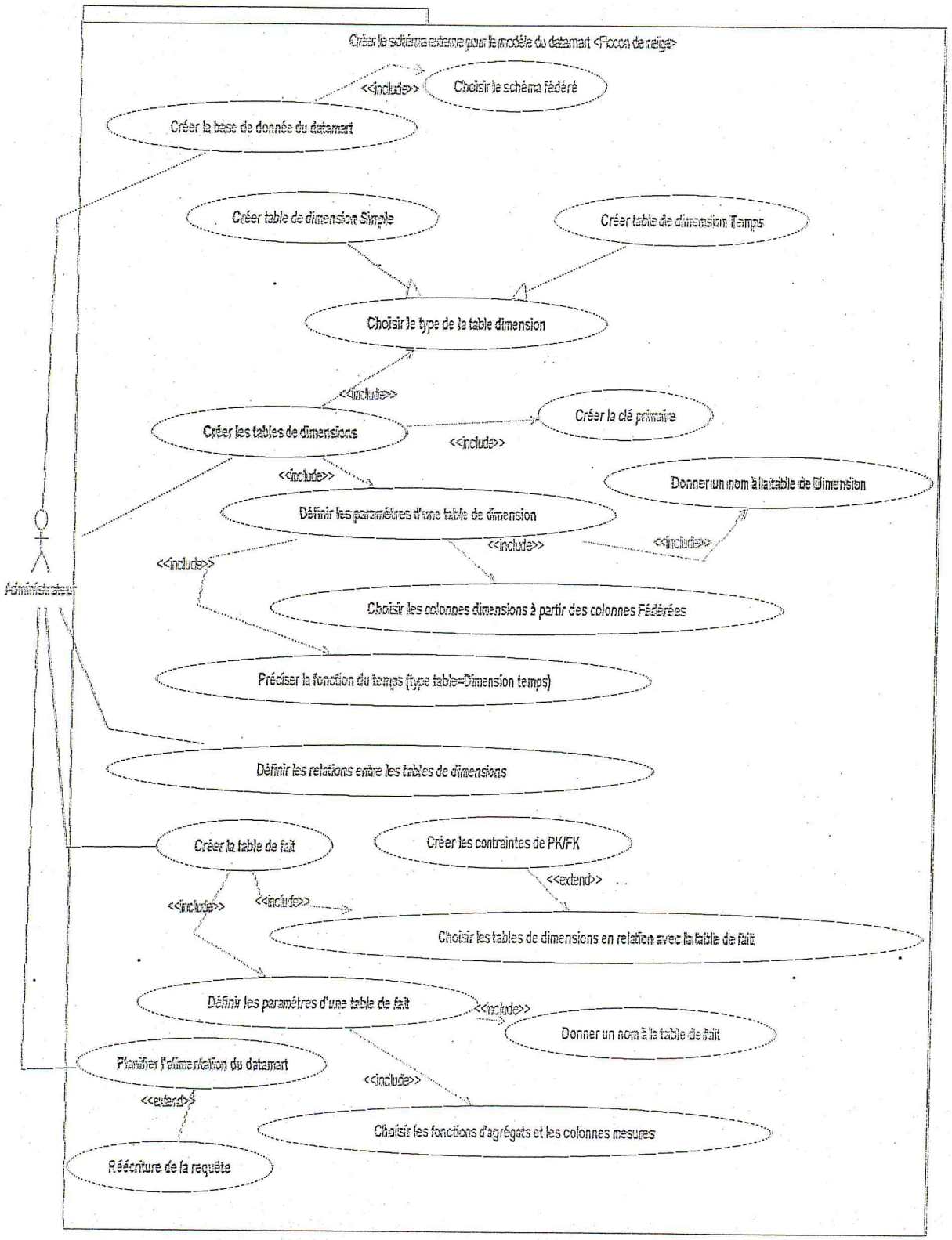


Figure2.2.9 Diagramme de cas d'utilisation «Créer le schéma externe pour le modèle du datamart flocon de neige»

Description textuelle des cas d'utilisation

Cas d'utilisation	Description
Créer la base de données du datamart	Pour créer la base de données datamart ; l'administrateur doit choisir le schéma fédéré .
Créer les tables de dimensions	Pour créer la table de dimension, l'administrateur doit : • choisir entre la création d'une table de dimension simple et une table de dimension temps • après le chois de la table de dimension il donne un nom à la table de dimension, et il sélectionne à partir des colonnes fédérées, les colonnes de dimensions. Et si le type de la table dimension est de type dimension temps, l'administrateur doit choisir la fonction du temps, pour les colonnes de type date • définir la clé primaire d'une table de dimension
Définir les relations entre les tables de dimensions .	l'administrateur doit définir les relations entre les tables de dimensions (créer les clés étrangères).
Créer la table de fait	Pour créer la table de fait, l'administrateur doit : • Donner un nom à la table de fait • Choisir les colonnes de mesure de fait à partir des colonnes fédérées, pour chaque mesure, il doit

	spécifier la fonction d'agrégat. • Définir les tables de dimensions en relation avec la table de fait, pour que le système peut créer automatiquement la contrainte de PK/FK pour la table de fait.
Planifier l'alimentation du datamart	Pour alimenter le datamart l'administrateur, doit planifier le système, pour rafraichies les données *périodiquement dans le datamart.cette* tache déclenche la réécriture de la requête d'alimentation pour insérer les données qui sont dispatchées sur des différents sites.

Tableau 2.2.8 Description textuelle «Créer le schéma externe pour le modèle du *datamart Flocon de neige*»

2.2 Catalogue d'ETL Data

Le catalogue de notre système «ETL Data » est une base de données relationnelle(de type SQL Server) qui reprend fidèlement les niveaux et les schémas tels que présentés sur l'architecture .le catalogue permet au système de traiter les différentes sources de données depuis la traduction en modèle relationnel jusqu'à la construction du datamarts. La Figure suivante illustre le diagramme de classes de la base de données qui contient les tables nécessaires pour définir les correspondances entre , les schémas pivots, les schéma exports, le schéma fédéré, et le schéma externe.

L'intérêt du Catalogue-ETL est de conserver la trace des données produites, Cela est rendu possible grâce aux méta-données qui permettent de stocker des informations telles que le nom de la base de production dont la donnée est extraite, le type de SGBD, le chemin des fichiers XML.....etc.

En outre le catalogue de notre système est considéré comme un support des informations portant sur la localisation et la disposition de ces éléments au sein des bases de production

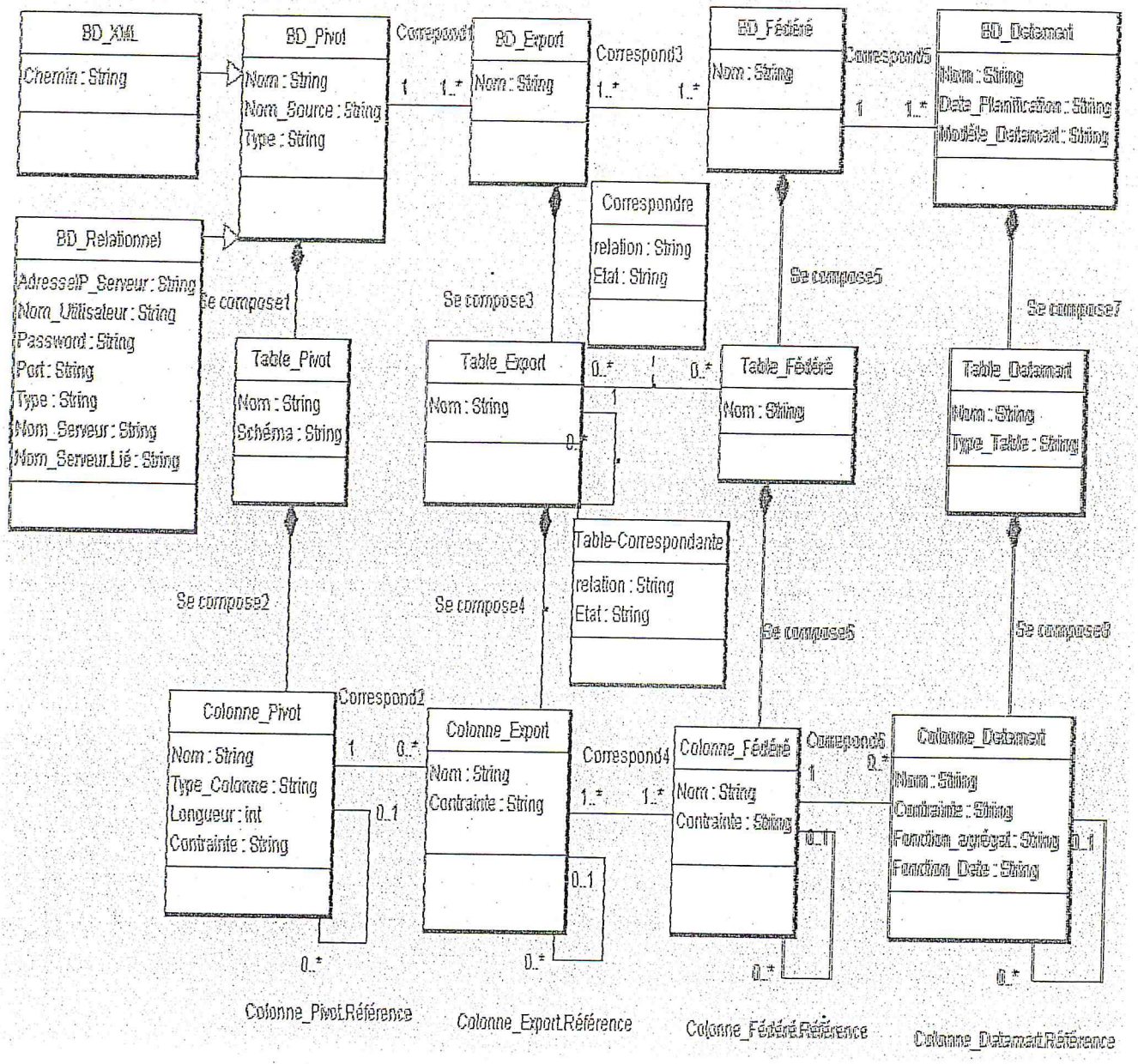


Figure2.2.10 Diagramme de Classes

2.2.2.2 La liste des attributs de chaque table

Nom de la Table	Nom de l'attribut	Description
BD_Pivot	Nom	Détermine le nom de la base de données pivot
	Nom_Source	détermine le nom du fichier source
	Type	Détermine le type de fichier à extraire (Relationnel, XML)
BD_relationnelle	AdresseIP_Serveur	Détermine l'adresse du serveur
	Nom_Utilisateur	Détermine le nom d'utisateur de chaque serveur
	Password	Détermine le mot de passe de chaque serveur
	Port	Détermine le port du serveur
	Type	Détermine le type de Système de Gestion de la base de données (PostgreSQL,SQL Server ..)
	Nom_Serveur	Détermine le nom du serveur pour chaque base de données.
	Nom_Serveur.Lié	Détermine le nom du serveur lié au Serveur ETL Data
BD_XML	Chemin	Contient les paramètres qui servent à localiser la Base de données Source
Table_Pivot	Nom	Détermine le nom de la table pivot
	Schéma	Détermine le schéma de la table si la base de données à extraire appartient à PostgreSQL (public , private,ou bien autre schéma)

Colonne_Pivot	Nom	Détermine le nom de la colonne pivot
	Type	Détermine le Type de la colonne pivot
	Longueur	Détermine la longueur de la colonne pivot
	Contrainte	Détermine la contrainte de clé primaire ou bien clé étrangère pour chaque colonne
BD_Export	Nom	Détermine le nom de la base de données Export
Table_Export	Nom	Détermine le nom de la table export
Colonne_Export	Nom	Détermine le nom de la colonne export
	Contrainte	Détermine la contrainte de clé primaire ou bien clé étrangère pour chaque colonne Export (PK,FK,PK/FK)
BD_Fédéré	Nom	Détermine le nom pour chaque base de données fédérée
Table_Fédérée	Nom	Détermine le nom de la table fédéré
Colonne_Fédérée	Nom	Détermine le nom de la colonne Fédéré
	Contrainte	Détermine la contrainte de clé primaire ou bien clé étrangère pour chaque colonne Fédéré (PK,FK,PK/FK)
BD_Datamart	Nom	Détermine le nom pour chaque datamart
	Date_Planification	Détermine la dernière date de planification du datamart
	Modèle_Datamart	Détermine le modèle du datamart (étoile ou bien flocon de neige)
Table_Datamart	Nom	Détermine le nom de la table datamart
	Type_Table	Détermine le type de la table datamart (Dimension ou bien fait).
Colonne_Datamart	Nom	Détermine le nom de la colonne datamart
	Contrainte	Détermine la contrainte de clé primaire ou bien clé étrangère pour chaque colonne

		datamart (PK,FK,PK/FK)
	Fonction_agrégat	Détermine la fonction d'agrégat d'une colonne de mesure d'une table de fait
	Fonction-date	Détermine la fonction de temps pour une colonne datamart de type date (Day ,Month,Year)

Tableau 2.2.10 liste des attributs de chaque table

2.2.2.3 La liste des associations

Association	Tables correspond a l'association	Description de l'association
Colonne_Pivot. Référence	Colonne_Pivot , Colonne_Pivot	Fait correspondre à chaque clé étrangère d'une table la clé primaire qu'elle référencie dans une autre table.
Se compose1	BD_Pivot, Table_Pivot	Elle fait correspondre à chaque Table pivot, La base de données pivot auquel elle appartient
Se compose2	Table_Pivot, Colonne_Pivot	Elle fait correspondre à chaque Colonne pivot, La table pivot auquel elle a appartient
Correspond1	BD_Pivot, BD_Export	Elle fait correspondre à chaque Base de données pivot, une base de données Export
Correspond2	Colonne_Pivot,Colonne_Export	Elle fait correspondre à chaque colonne export, une colonne pivot
Correspond3	BD_Export,BD_Fédéré	Elle fait correspondre à

		chaque Base de données Export, une base de données Fédéré
Se compose 3	BD_ Export, Table_Export	Elle fait correspondre à chaque Table export, La base de données export auquel elle appartient
Se compose 4	Table_Export, Colonne_Export	Elle fait correspondre à chaque Colonne export, La table export auquel elle appartient
Correpond4	Colonne_Export, Colonne_Fédéré	Elle fait correspondre à chaque colonne Fédéré, une colonne export
Table_Correspendante	Table_Export, Table_Export	Elle fait correspondre à chaque table export d'un schéma une autre table export d'un autre schéma, selon une *relation* (Commune,autre) , et un *état* qui définit si ce cas a été pris en charge ou non .
Correspondre	Table_Export , Table_Fédéré	Elle fait correspondre à chaque nouvelle table export d'un schéma export une table fédérée déjà définit dans un schéma fédéré, selon une *relation* (Commune,autre) , et un *état* qui définit si ce cas a été pris en charge ou non .

Se compose5	BD_Fédérée,Table_Fédéré	Elle fait correspondre à chaque table fédérée, La base de données fédérée auquel elle appartient
Se compose6	Table_Fédérée,Colonne_Fédérée	Elle fait correspondre à chaque Colonne fédérée, La table fédérée auquel elle appartient.
Colonne_Fédéré e.Référence	Colonne_Fédéré, Colonne_Fédéré .	Fait correspondre à chaque clé étrangère d'une table Fédérée la clé primaire qu'elle référencie dans une autre table Fédérée
Correspond5	BD_Fédéré,BD_Datamart	Elle fait correspondre à chaque Base de données Fédéré, une base de données Datamart.
Se compose7	BD_Datamart,Table_Datamart	Elle fait correspondre à chaque table datamart(fait ou bien dimension), La base de données datamart auquel elle appartient.
Se compose 8	Table_Datamart,Colonne_Datamart	Elle fait correspondre à chaque Colonne datamart, La table datamart auquel elle appartient
Correpond6	Colonne_Fédéré,Colonne_Datamart	Elle fait correspondre à chaque colonne Datamart, une colonne Fédéré.
Colonne_Datama rt.Référence	Colonne_Datamart ,Colonne_Datamart	Fait correspondre à chaque clé étrangère d'une table datamart la clé

		primaire qu'elle référencie dans une autre table datamart

Tableau 2.2.11 liste des associations

2.2.3 Traduction UML-Relationnel

2.2.3.1 Règles de passage

Chaque **classe** devient une **relation** les attributs de la classe deviennent attributs de la relation, l'identifiant de la classe devient clé primaire de la relation

- ***Traduction des associations 1-N***

Chaque association 1-N est prise en compte en incluant la clé primaire de la relaion dont la multiplicité maximale et cette derniere est définie comme clé étrangére .

- ***Traduction des associations M-N***

Chaque association M-N est est prise en compte en incluant en créant une nouvelle relation dont la clé primaire et la concaténation des clés primaires des relations participantes

- ***Traduction de la relation d'héritage***

Les sous-classes ont méme clé primaire que la superclasse , ou bien hérite la clé primaire de la classe mére .

- ***Traduction de la composition***

une relation composition est traduit comme une association 1-N (défininit précédemment)

- ***Traduction de la relation réfléxive M-N***

Une relation réflexive se traduit par l'ajout d'une table dont la clé primaire est la duplication de la clé primaire de la classe par concaténation.

2.2.3.2 Passage du diagramme de classe au modèle relationnel

1. BD_Pivot(Nom_BD.Pivot, Nom_BD.Source, Type)
2. BD_Relationnel(Nom_BD.Pivot #, AdresseIP_Serveur.Source, Nom_Utilisateur, Password, Port, Type,Nom_Serveur,Nom_Serveur.Lié)
3. BD_XML(Nom_BD.Pivot#, Chemin)
4. Table_Pivot(Nom_Table.Pivot, Nom_BD.Pivot #,Schéma)
5. Colonne_Pivot(Nom_Colonne.Pivot, Nom_Table.Pivot#, Nom_BD.Pivot#, Type, Longueur, Contrainte)
6. Colonne_Référence(Nom_Colonne.FK#, Nom_Table.FK#, Nom_BD.FK#, Nom_Colonne.PK#, Nom_Table.PK#, Nom_BD.PK #)
7. BD_Export(Nom_BD.Export, Nom_BD.Pivot#)
8. Table_Export(Nom_Table.Export, Nom_BD.Export#)
9. Table_Correspendante(Nom_Table.Export#,Nom_BD.Export#,Nom_Table.Correspendante#,Nom_BD.Correspendante#,Relation,Etat)
10. Correspondre(Nom_Table.Export#, Nom_BD.Export #, Nom_Table.Fédéré#, Nom_BD.Fédéré#,Relation,Etat).
11. Colonne_Export(Nom_Colonne.Export, Nom_Table.Export#, Nom_BD.Export#, Nom_Colonne.Pivot#, Nom_Table.Pivot#, Nom_BD.Pivot#, Contrainte).
12. Colonne_Export.Référence(Nom_Colonne.FK#, Nom_Table.FK#, Nom_BD.FK#, Nom_Colonne.PK#, Nom_Table.PK#, Nom_BD.PK #)
13. BD_Fédéré(Nom_BD.Fédéré)
14. Table_Fédéré(Nom_Table.Fédéré, Nom_BD.Fédéré#)
15. Colonne_Fédéré(Nom_Colonne.Fédéré, Nom_Table.Fédéré#, Nom_BD.Fédéré#, Contrainte)
16. Colonne_Fédérée.Référence(Nom_Colonne.FK #,Nom_Table.FK#,Nom_BD.FK,Nom_Colonne.PK#,Nom_Table.PK#,Nom_BD.PK#)
17. BD_Export_Fédérée(Nom_BD.Fédéré#, Nom_BD.Export#)
18. Colonne_Export_Fédérée(Nom_Colonne.Export#, Nom_Table_Export#, Nom_BD.Export #, Nom_Colonne.Fédéré#, Nom_Table.Fédéré#, Nom_BD.Fédéré#)

19. BD_Datamart(**Nom_BD.Datamart**, Date_Planification, Modéle_Datamart,Nom_BD.Fédéré#)
20. Table_Datamart(**Nom_Table.DataMart, Nom_BD.Datamart#**, Type_Table)
21. Colonne_Datamart(**Nom_Colonne.Datamart, Nom_Table.Datamart#, Nom_BD.Datamart#**, Nom_Colonne.Fédéré#, Nom_Table.Fédéré#, Nom_BD.Fédéré#, Contrainte , Fonction_agrégat,Fonction_Date).
22. Colonne_Datamart.Référence(**Nom_Colonne.FK#, Nom_Table.FK#, Nom_BD.FK#**, Nom_Colonne.PK#, Nom_Table.PK#, Nom_BD.PK#)

Légende

_____ :Clé primaire(PK)

:Clé étrangére (FK)

2.3 Diagramme de séquence

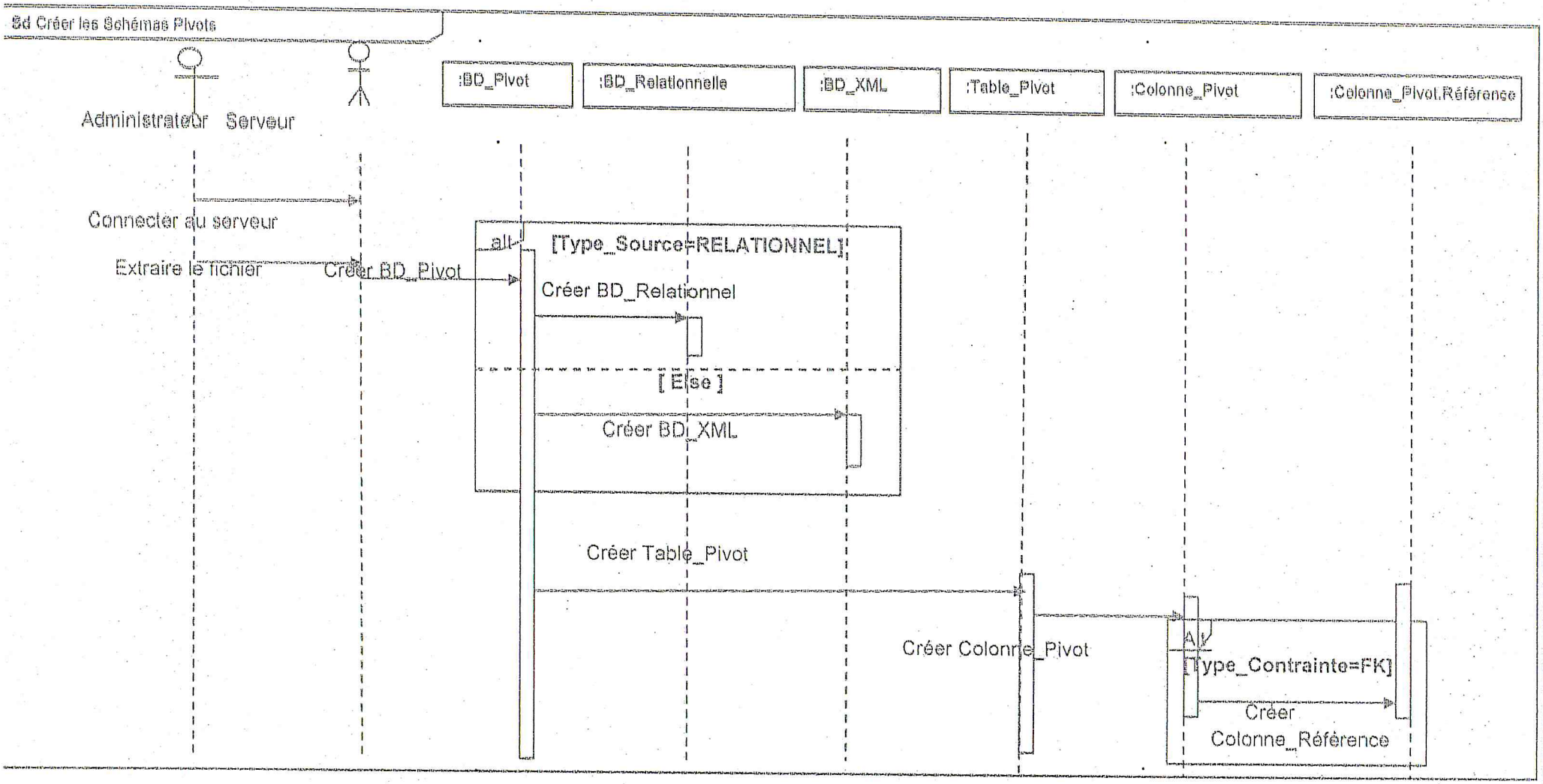
Bien que de nombreux diagrammes d'UML permettent de décrire les cas d'utilisations, on a utilisé deux types de diagrammes, le diagramme de séquence et le diagramme d'activité.

- *Le diagramme de séquence* montre comment des instances au cœur du système communiquent pour réaliser une certaine fonctionnalité.

2.3.1 Diagramme de séquence «Créer les Schémas Pivots »

Pour illustrer comment l'administrateur interagit avec les instances des classes BD_Pivot,BD_Relationnelle,BD_XML,Table_Pivot,Colonne_Pivot,Colonne_Référen*ce on définit le diagramme de séquence suivant.*

Figure 2.2.11 Diagramme de séquence «Créer les schémas pivots »

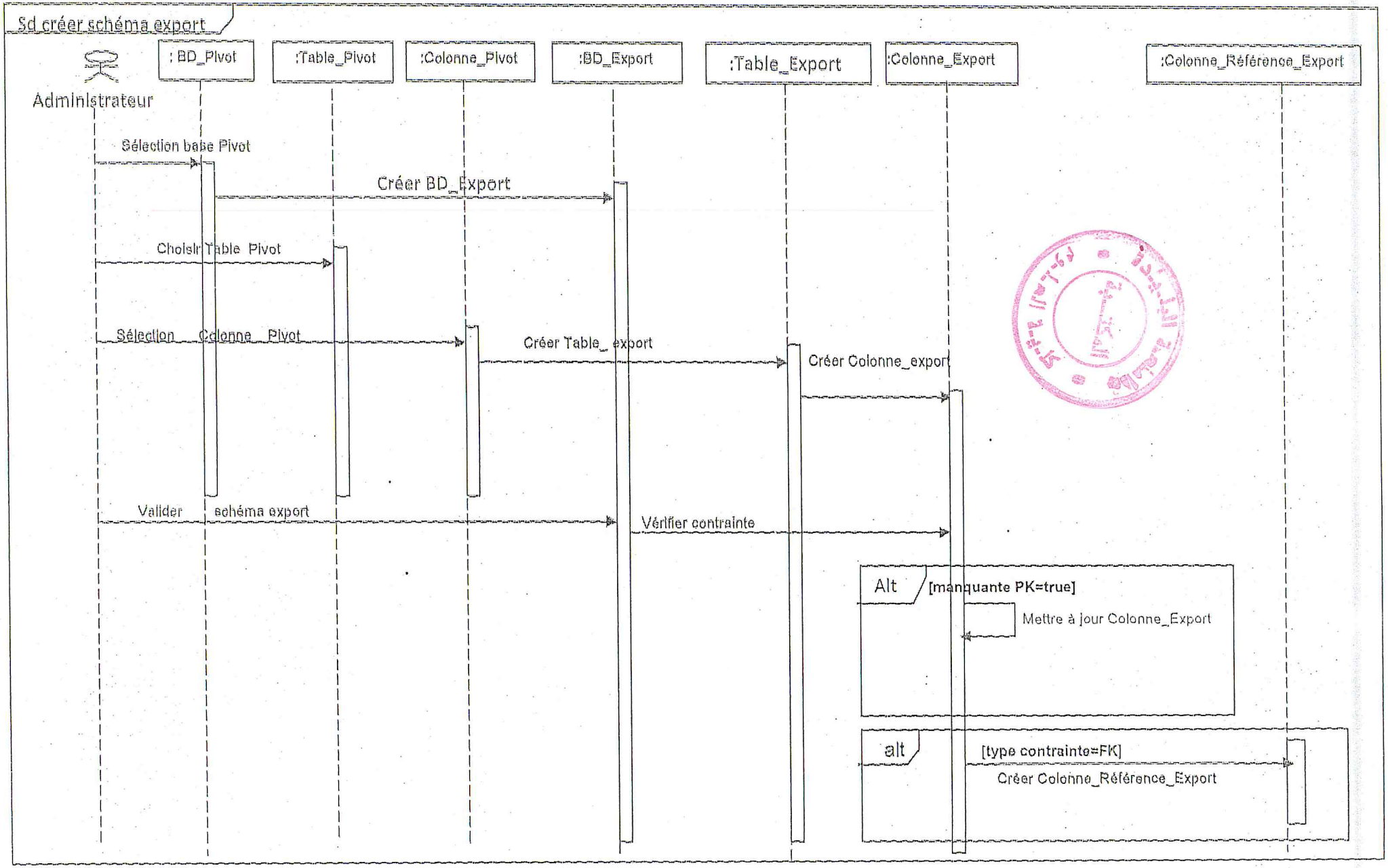


- La création de l'objet BD_Pivot dans la base de données Catalogue_ETL débute au moment où l'adminstrateur choisit le fichier à extraire.
- Selon le type de fichier à extraire, une instance BD_Relationnelle ou BD_XML est crée dans le Catalogue_ETL.
- Pour terminer la création du schéma pivot d'une base de données, les différentes tables de la base de données sont crées dans la Classe Table_Pivot, et pour chaque table, leurs colonnes dans la Classe Colonne_Pivot.
- Si le type de la contrainte d'une colonne est de type «Clé étrangère», alors une instance dans la classe Colonne_Référence est crée.

2.3.2 Diagramme de séquence «Créer les Schémas Exports »

Le diagramme suivant illustre l'interaction entre l'administrateur et les instances des classes concernant la création d'un schéma export

Figure 2.2.12. Diagramme de séquence « créer les schémas exports »

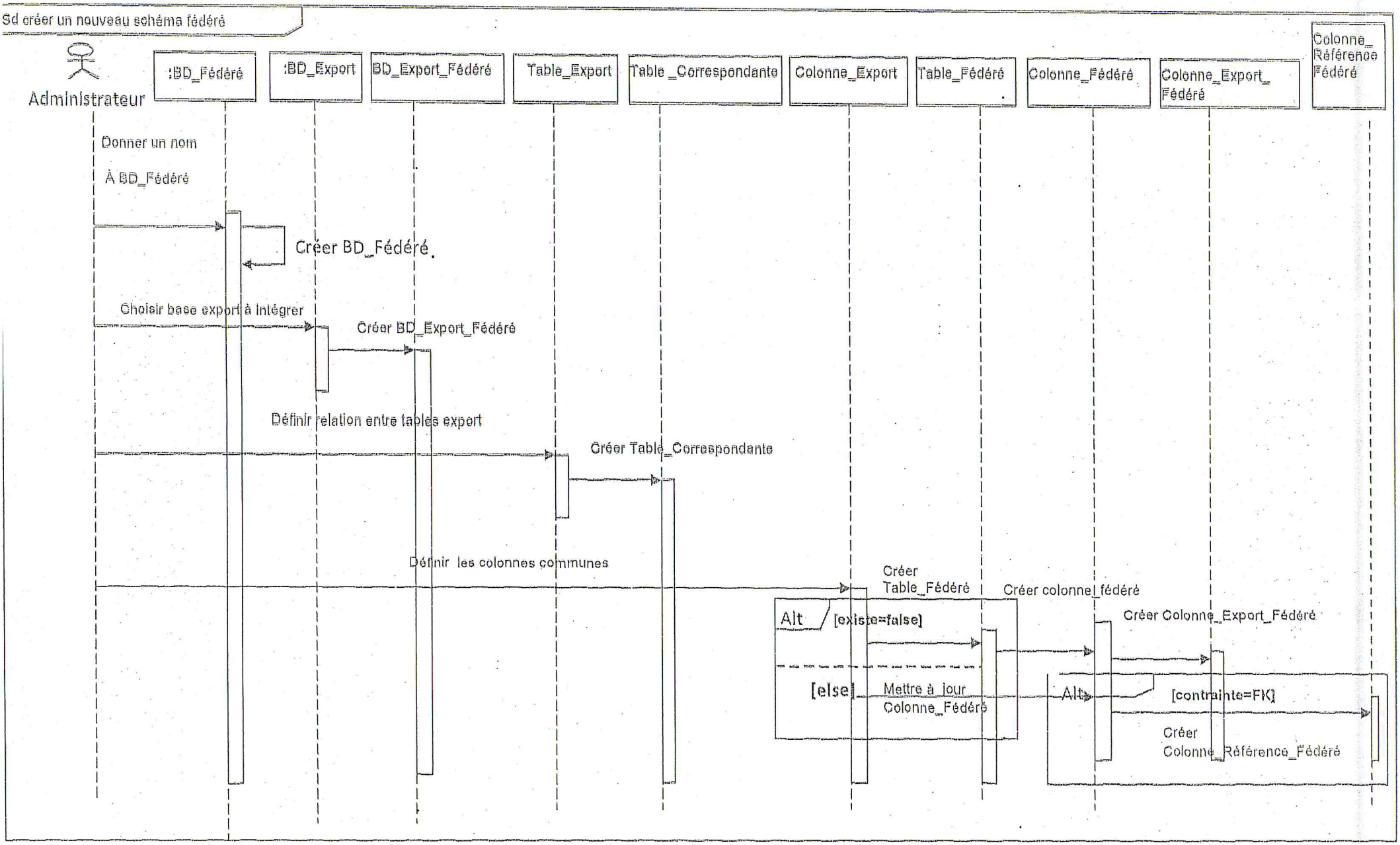


- Après avoir définit les schémas pivots, l'administrateur peut créer à partir de ces derniers, des schémas exports, pour ce fait l'administrateur doit choisir pour chaque schéma export qu'il veut créer, un schéma Pivot.
- Afin de définir les tables et colonne export pour son schéma export, L'administrateur doit choisir les tables pivot ainsi que leurs colonnes jugé pertinentes.
- L'administrateur doit valider son schéma export comme dernier pas, cette validation va déclencher une vérification au niveau colonne fédéré, le système va vérifier que les clés primaire de chaque table export pris, sont définit, ainsi qu'il va vérifier que la table référencé de chaque colonne définit comme FK dans une tables export a été définit, si elle n'est pas définit, le système offre à l'administrateur deux solution : soit d'ajouter la table référencé au schéma export, soit de supprimer la colonne définit comme FK, c'est-à-dire supprimer la relation entre ces deux table.
- Si une colonne a comme contrainte FK alors sa colonne PK référencé sera insérer dans la table Colonne_Référence_Export.

2.3.3 Diagramme de séquence «Créer un nouveau Schémas fédéré »

Le diagramme suivant illustre l'interaction entre l'administrateur et les instances des classes concernant la création d'un schéma fédéré.

Figure 2.2.13 diagramme de séquence « créer un nouveau schéma fédéré »

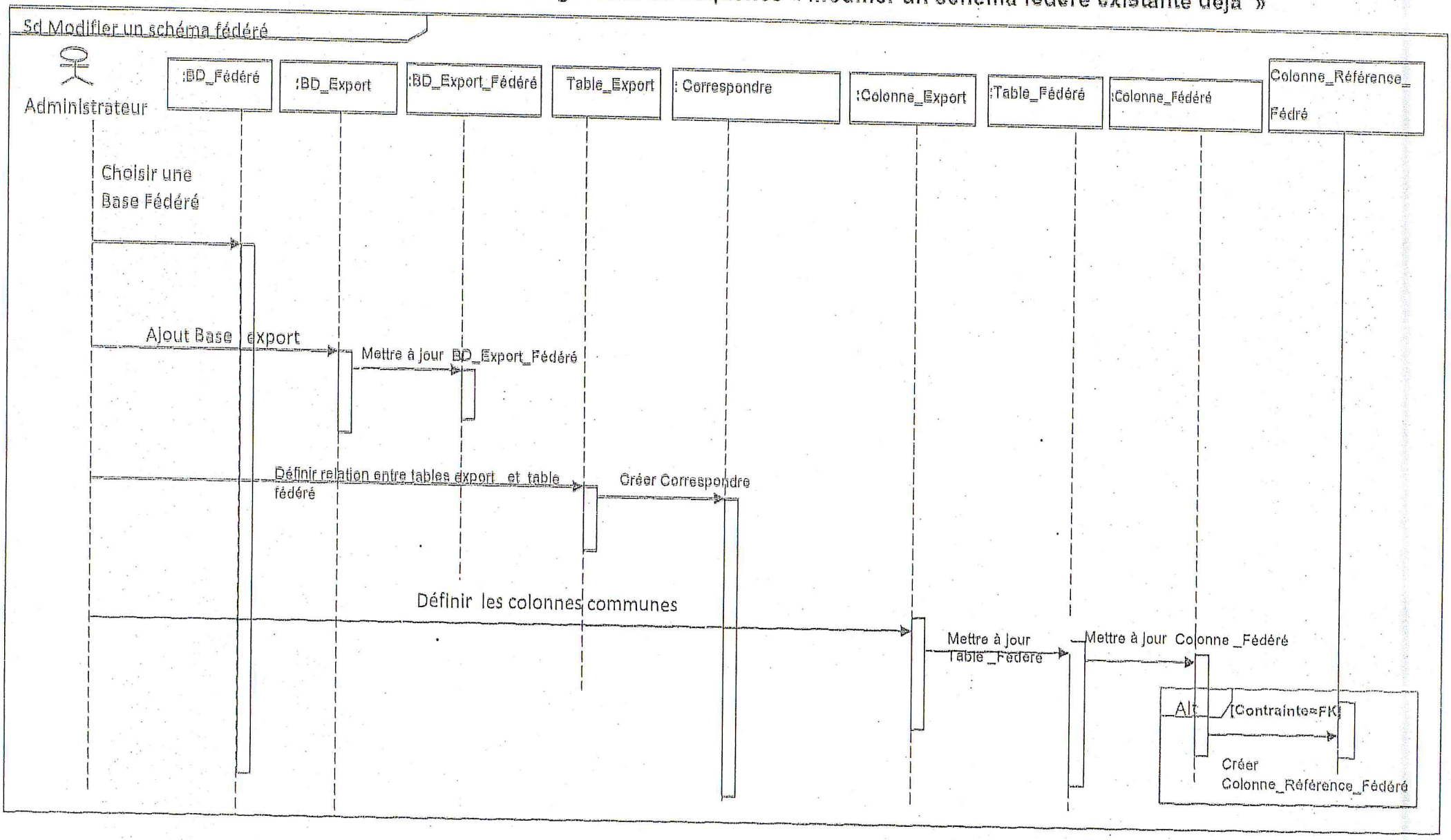


- L'administrateur va créer un schéma fédéré toute en lui spécifiant un nom.
- L'administrateur doit choisir les schémas exports qu'il souhaite intégrer dans son nouveau schéma fédéré, ce qui mène à une mise à jour de la table BD_Export_Fédéré.
- Afin d'intégrer et d'homogénéisé ces différents schémas exports déjà choisi, l'administrateur doit définir les types de relations existantes entre les tables exports (« commune », ou « autre »), c'est l'instanciation de la classe « Table_Correspondante ».
- Au tant qu'il existe des relations de type « table commune » l'administrateur doit définir les colonnes communes entre les tables qui ont cette relations, choisir un schéma de table pour sa table fédéré, le système va vérifier si la table fédérée existe déjà, si oui l'administrateur va modifier la table fédéré sinon la table fédérée sera créé et ses colonnes seront définit.
- Après avoir définit les colonnes communes entre les tables de type de relation « table commune », l'administrateur définit les colonnes communes entre les tables de type de relation « autre »,le système va vérifier si la table fédérée existe déjà, si oui l'administrateur va redéfinir les colonnes communes et la table fédérée sera modifiée , sinon la table fédéré sera créé et ses colonnes seront définit.
- Après avoir définit le schéma fédéré, ses tables et ses colonnes, le système crée une instance de type Table_Référence.Fédéré.

2.3.4 Diagramme de séquence «Modifier un Schémas fédéré existant déjà»

Le diagramme suivant illustre l'interaction entre l'administrateur et les instances des classes concernant la modification d'un schéma fédéré.

Figure 2.2.14 diagramme de séquence « modifier un schéma fédéré existante déjà »

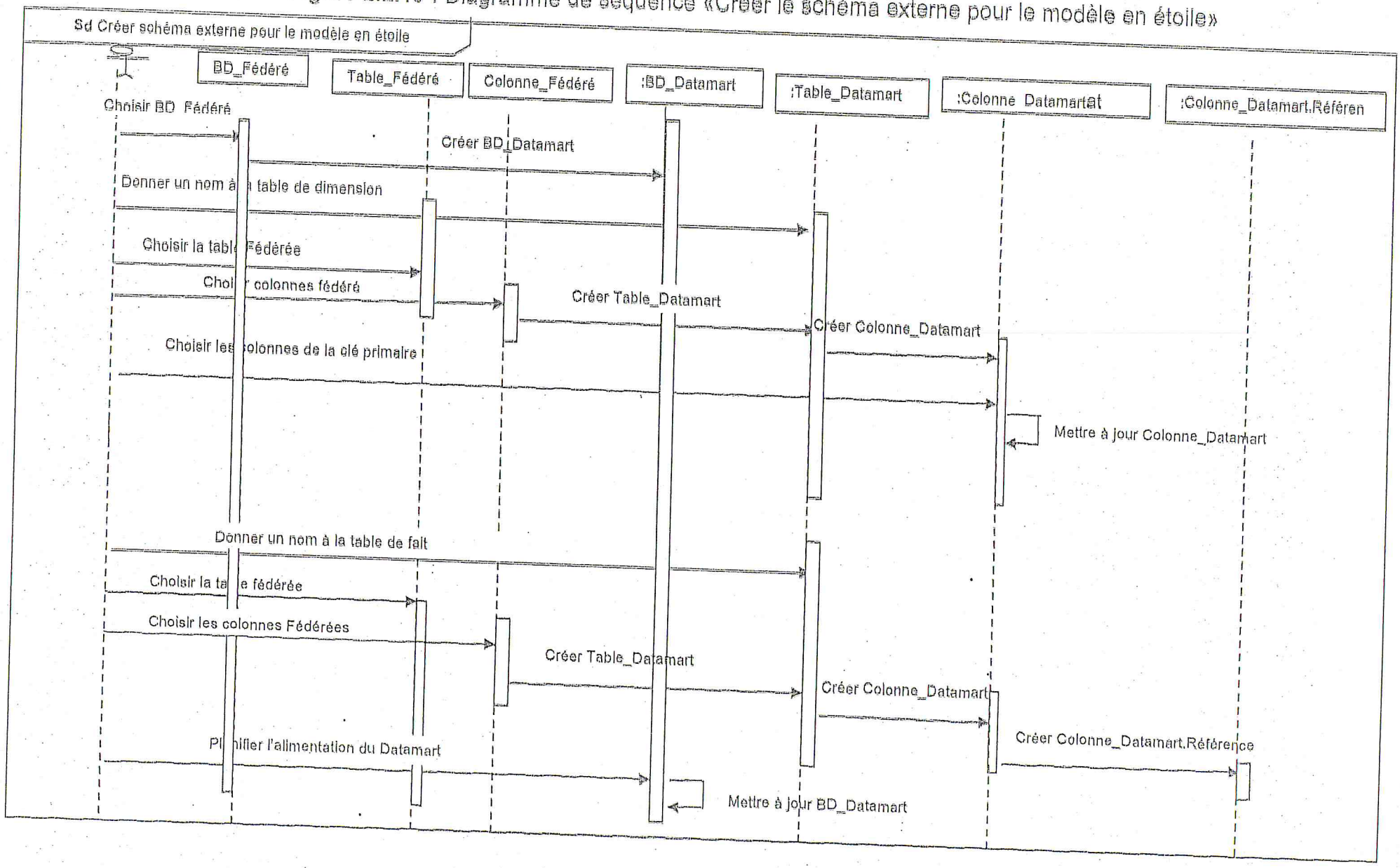


- Afin de modifier un schéma fédéré existant L'administrateur doit d'abord sélectionner le schéma à modifier.
- L'administrateur doit choisir les schémas exports qu'il souhaite ajouter à son schéma fédéré, ce qui mène à une mise à jour de la table BD_Export_Fédéré.
- Afin d'intégrer et d'homogénéisé ces différents schémas exports déjà choisi, l'administrateur doit définir les types de relations existantes entre les tables exports et les table fédérées (« commune » pour dire qu'on parle pratiquement de la représentation du même objet dans le monde réel dans les deux tables, ou « autre » pour définir une relation qui reste inconnu à ce stade là mais qui existe entre ces deux tables), c'est l'instanciation de la classe « Correspondre»
- Au tant qu'il existe des relations de type « table commune » l'administrateur doit définir les colonnes communes entre les tables qui ont cette relations, et il va modifier la table fédérée.
- Après avoir définit les colonnes communes entre les tables de type de relation « table commune », l'administrateur définit les colonnes communes entre les tables de type de relation « autre », le système va modifier la table fédérée et définir les colonnes fédérées.
- Après avoir définit le schéma fédéré, ses tables et ses colonnes, le système crée une instance de type Table_Référence.Fédéré.

2.3.5 Diagramme de séquence «Créer le Schéma Externe pour le modèle en étoile»

Le diagramme de séquence montre les interactions entre l'administrateur et les différentes instances, concernent le schéma externe.

Figure 2.2.15 : Diagramme de séquence «Créer le schéma externe pour le modèle en étoile»

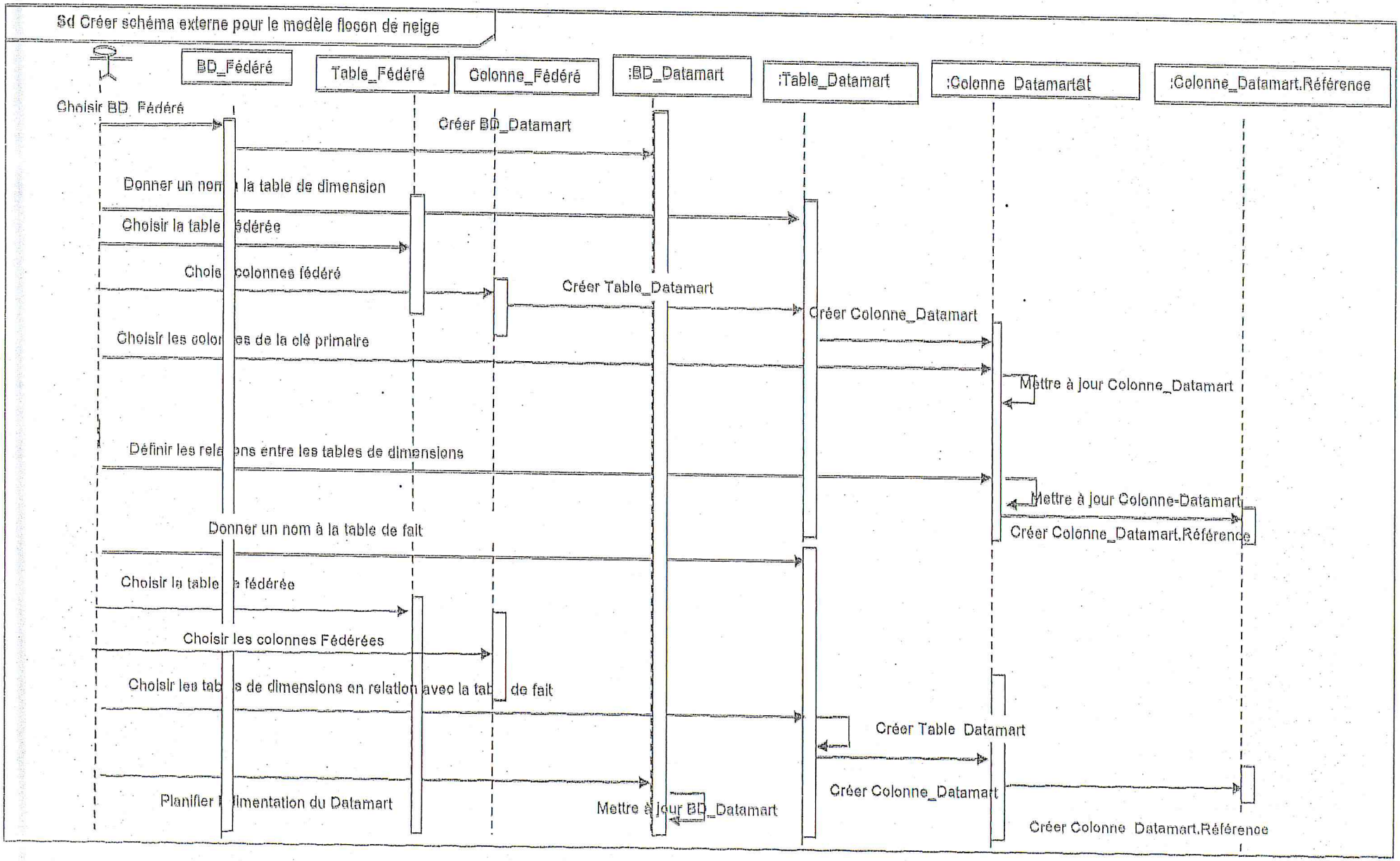


- L'administrateur doit choisir le schéma fédéré,qu'il souhaite créer son datamart ,une instance DB_Datamart est crée dans la classe BD_Datamart.
- Pour créer la table de dimension L'administrateur doit attribuer un nom à la table de dimension et choisir les colonnes de dimension à partir des colonnes fédérées, l'instance Table_Datamart est créée, et Colonne_Datamart aussi.
- Après la création de la table de dimension, l'administrateur doit choisir les colonnes de dimension, pour créer la clé primaire. ce qui mène à une mise à jour de la Colonne_Datamart
- Au moment où l'administrateur termine la création de toutes les tables de dimension, il passe à la création de la table de fait , pour ce faire , il donne un nom à la table de fait , il choisit les colonnes de mesures à partir des colonnes fédérées ,une instance de Table_Datamart est crée , et les objets Colonne_Datamart sont instanciés.
- Enfin l'administrateur doit planifier l'alimentation du datamart , ce qui mène à une mise à jour de la BD_Datamart.

2.3.6 Diagramme de séquence «Créer le Schéma Externe pour le modèle Flocon de neige»

Le diagramme de séquence montre les interactions entre l'administrateur et les différentes instances, concernent le schéma Externe.

Figure 2.2.16 : Diagramme de séquence «Créer schéma externe pour le modèle flocon de neige»



- L'administrateur doit choisir le schéma fédéré qu'il souhaite créer son datamart ,une instance DB_Datamart est crée dans la classe BD_Datamart.
- Pour créer la table de dimension L'administrateur doit attribuer un nom à la table de dimension et choisir les colonnes de dimensions à partir des colonnes fédérées, l'instance Table_Datamart est crée, et la Colonne_Datamart aussi.
- Après la création de la table de dimension, l'administrateur doit choisir les colonnes de dimension, pour créer la clé primaire. ce qui mène à une mise à jour de la Colonne_Datamart
- L'administrateur crée les relations entre les tables de dimensions , la Colonne_Datamart est mise à jour .
- Au moment où, l'administrateur termine la création de toutes ses tables de dimensions, il passe à la création de la table de fait , pour ce faire , il donne un nom à la table de fait , il choisit les colonnes de mesures à partir des colonnes fédérées.
- l'administrateur sélectionne les tables de dimensions en relation avec la table de fait, Les instances de la Table_Datamart , Colonne_Datamart sont créées , et Les objets Colonne_Datamart. référence sont instanciés.
- Enfin l'administrateur doit planifier l'alimentation du datamart , ce qui mène à une mise à jour de la BD_Datamart.

2.4 Les diagrammes d'activités

Le diagramme d'activité permettant de spécifier des traitements a priori séquentiels. Il est donc adapté à la spécification détaillée des traitements en phase de réalisation .On peut aussi l'utiliser pour décrire des enchainements d'actions de haut niveau, en particulier pour la description détaillée des cas d'utilisation.

2.4.1 Diagramme d'activité «Choisir un schéma relationnel»

Le diagramme d'activité «Choisir un schéma relationnel» détaille le cas d'utilisation « Choisir un schéma relationnel».

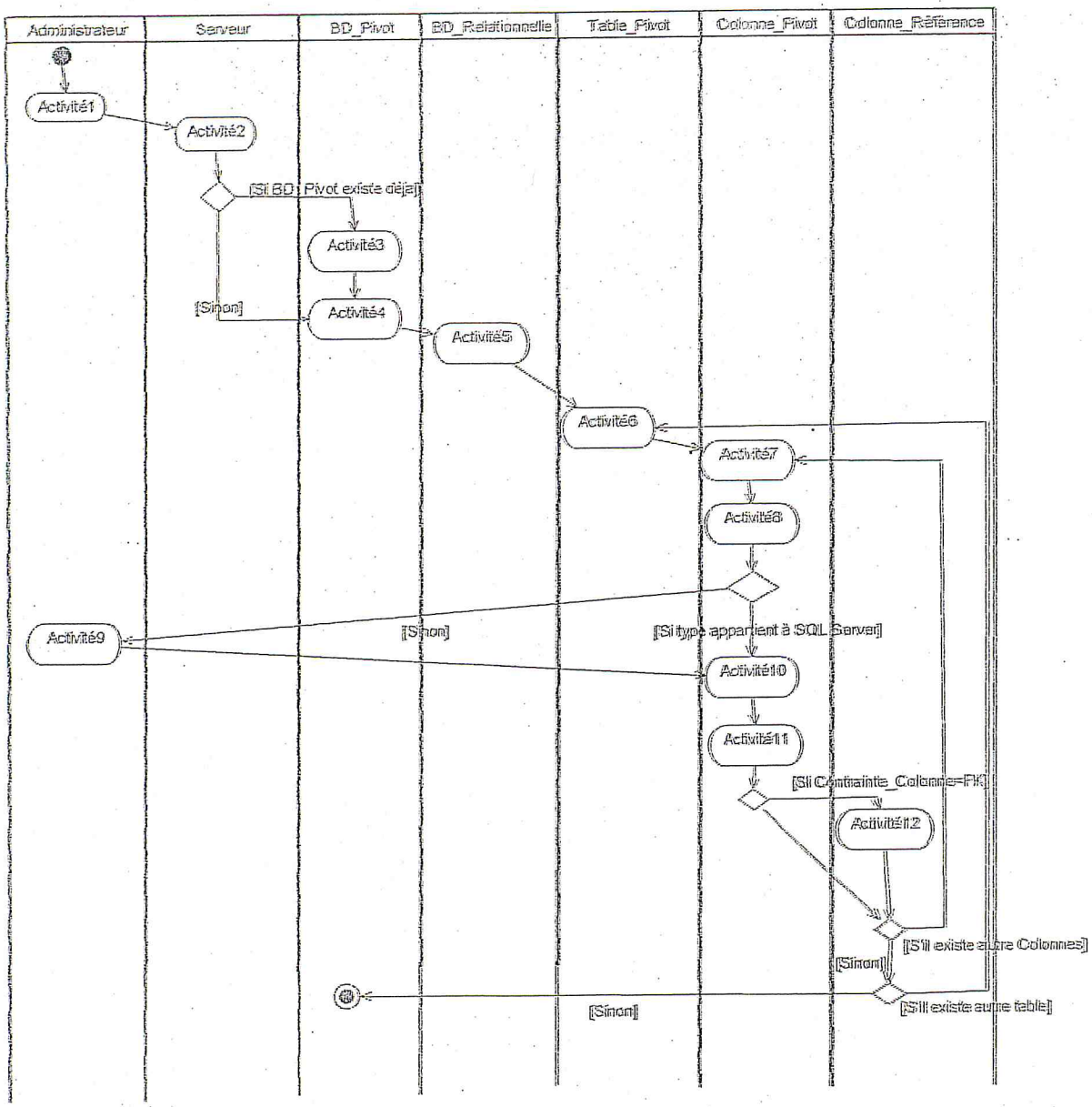


Figure 2.2.17 diagramme d'activité «Choisir un schéma relationnel»

Indice	Description
Activité1	Choisir le schéma relationnel
Activité2	Connecter au serveur
Activité3	Ajouter un index à la base de données pivot
Activité4	Créer BD_Pivot
Activité5	Créer BD_Relationnel
Activité6	Créer Table_pivot
Activité7	Rechercher les colonnes de la table source
Activité8	Rechercher le type de la colonne
Activité9	Donner un type proche, appartient aux types de SQL Server
Activité10	Définir une longueur pour chaque type
Activité11	Créer Colonne_Pivot
Activité12	Créer Colonne_Référence

Tableau 2.2.12 la description des activités pour le diagramme d'activité « Choisir un schéma relationnel »

2.4.2 Diagramme d'activité «Choisir un fichier XML»

Le diagramme d'activité «Choisir un schéma XML» détaille le cas d'utilisation « Choisir un schéma XML».

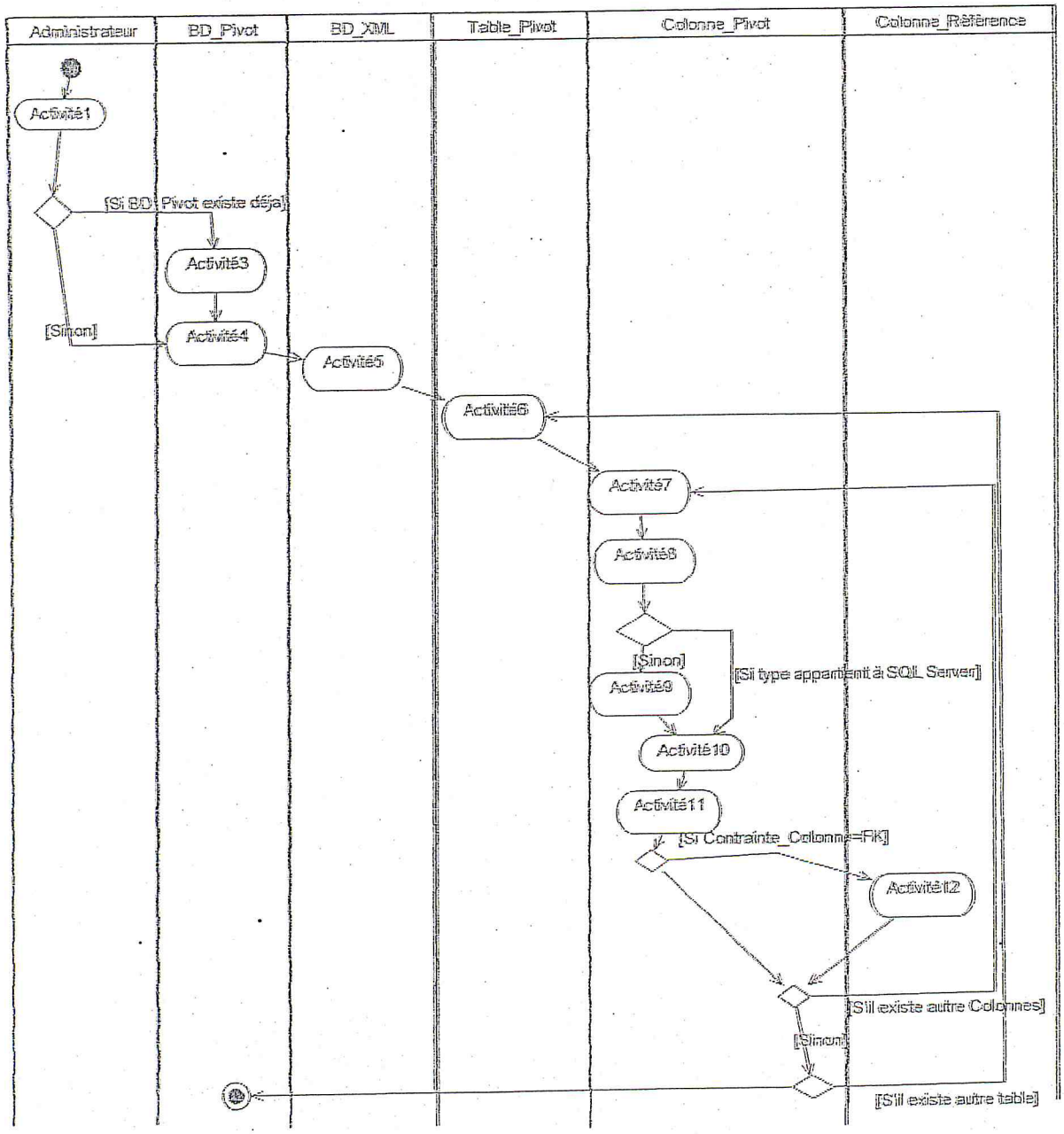


Figure 2.2.18 diagramme d'activité « Choisir un schéma XML »

Indice	Description
Activité1	Extraire un fichier XML
Activité2	Spécifier le schéma d'un fichier XML et son schéma XSD
Activité3	Ajouter un index à la base de données pivot
Activité4	Créer BD_Pivot
Activité5	Créer BD_XML
Activité6	Créer Table_pivot
Activité7	Rechercher les colonnes de la table source
Activité8	Rechercher le type de la colonne
Activité9	Définir un type proche, appartient aux types SQL Server
Activité10	Définir une longueur pour chaque type
Activité11	Créer Colonne_Pivot
Activité12	Créer Colonne_Référence

Tableau 2.2.13 la description des activités pour le diagramme d'activité

« Choisir un schéma XML »

2.4.5 Diagramme d'activité«Créer les schémas exports »

Le diagramme d'activité «Créer les schémas exports» détaille le cas d'utilisation Créer les schémas exports»

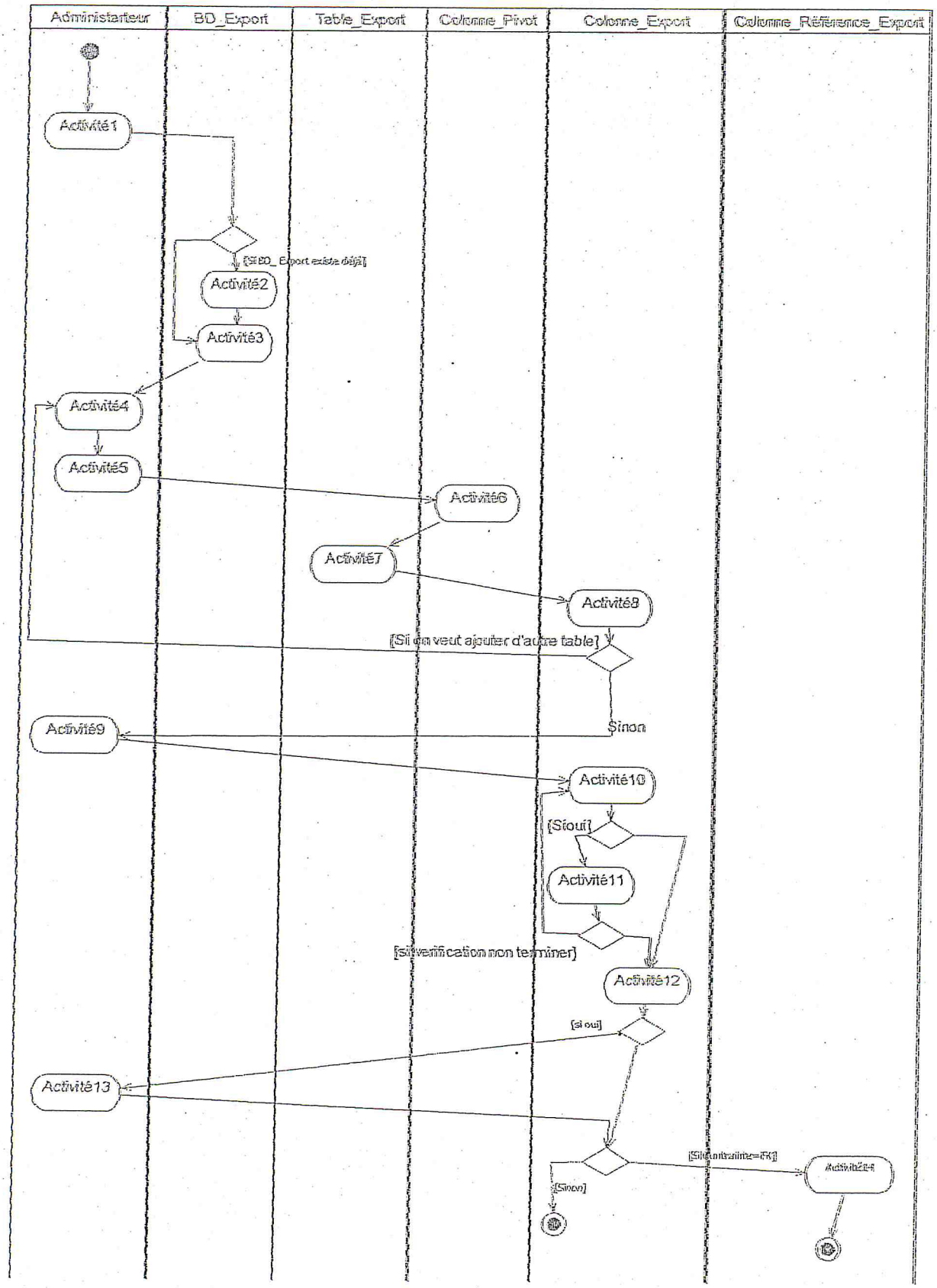


Figure 2.2.19 diagramme d'activité «Créer les schémas exports »

Indice	Description
Activité 1	Sélectionner un schéma pivot
Activité 2	Si le schéma existe Renommer le schéma export en joutant un index au nom du schéma pivot
Activité 3	Créer le schéma Export
Activité 4	Sélection les tables pivot juger pertinente.
Activité 5	Sélectionner les colonnes pivot s jugés pertinente pour l'analyse.
Activité 6	Définir contrainte colonne export
Activité 7	Définir Table_Export
Activité 8	Définir Colonne_Export
Activité 9	Valider le schéma export.
Activité 10	Vérifier pour chaque table du schéma export si une colonne de type Primary Key n'été pas ajouter à la table export qui lui comespond
Activité 11	Ajouter la colonne de type Primary Key
Activité12	Vérifier pour chaque table du schéma export si chauque colonne de type Foreign Key a une colonne référence PK .
Activité13	Ajouter la table qui contient la clé primaire référence de la clé étrangère , ou supprimer la colonne de type FK
Activité14	Mettre à jours la table Colonne_Référence_Export

Tableau 2.2.13 la description des activités pour le diagramme d'activité

2.4.6 Diagramme d'activité diagramme d'activité «Créer un nouveau schéma Fédéré »

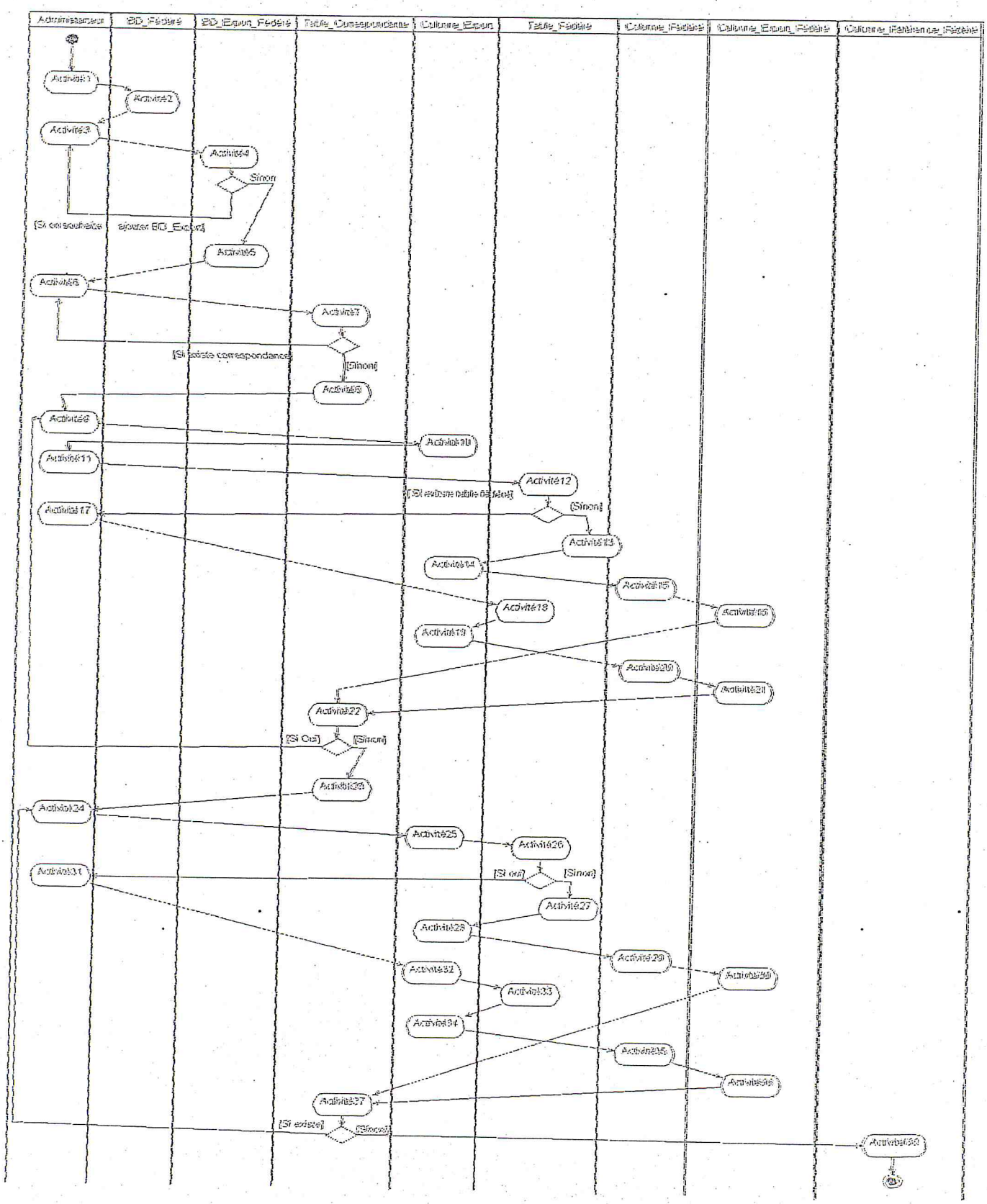


Figure 2.2.20 diagramme d'activité «Créer un nouveau schéma Fédéré »

Activité	Description
Activité1	Donner un nom au schéma fédéré
Activité2	Créer BD_Fédéré
Activité3	Sélectionner BD_Export à intégrer
Activité4	Mettre à jour BD_Export_Fédéré
Activité5	Sélectionner BD_Export qui ont été choisis pour être intégrer dans le schéma fédéré
Activité6	Définir les relations existantes entre les Table_Export des schémas exports déjà sélectioner
Activité7	Mettre à jours Table_Correspondante
Activité8	Sélectionner tables exports ou la relation égal à « table commune »
Activité9	Définir les colonnes communes
Activité10	Définir les colonnes non communes
Activité11	Choisir schéma de la table export qu'on souhaite garder dans la table fédéré
Activité12	Vérifier si la table fédérée à créer existe déjà suite à une autre définition de colonnes commune
Activité13	Créer la table fédérée
Activité14	Définir contraintes colonne fédéré
Activité15	Créer colonne_ fédérée
Activité16	Mettre à jour colonne _Export_Fédéré
Activité17	Redéfinir les colonnes communes
Activité18	Modifier la table Fédérée
Activité19	Définir contraintes colonne fédéré
Activité20	Mettre à jour Colonne_Fédéré
Activité21	Mettre à jour colonne _Export_Fédéré
Activité22	Vérifier s'il existe encore des relations de type « table commune » entre les tables exports ou Etat égal à non

Activité22	Vérifier s'il existe une encore des relations de type « Autre » entre les tables exports et les tables fédérées ou l'état égal à non
Activité23	Définir Colonne_Référence_Fédéré

Tableau 2.2.15 la description des activités pour le diagramme d'activité «Modifier un schéma fédéré existant déjà »

2.4.8 Diagramme d'activité «Créer Table de dimension»

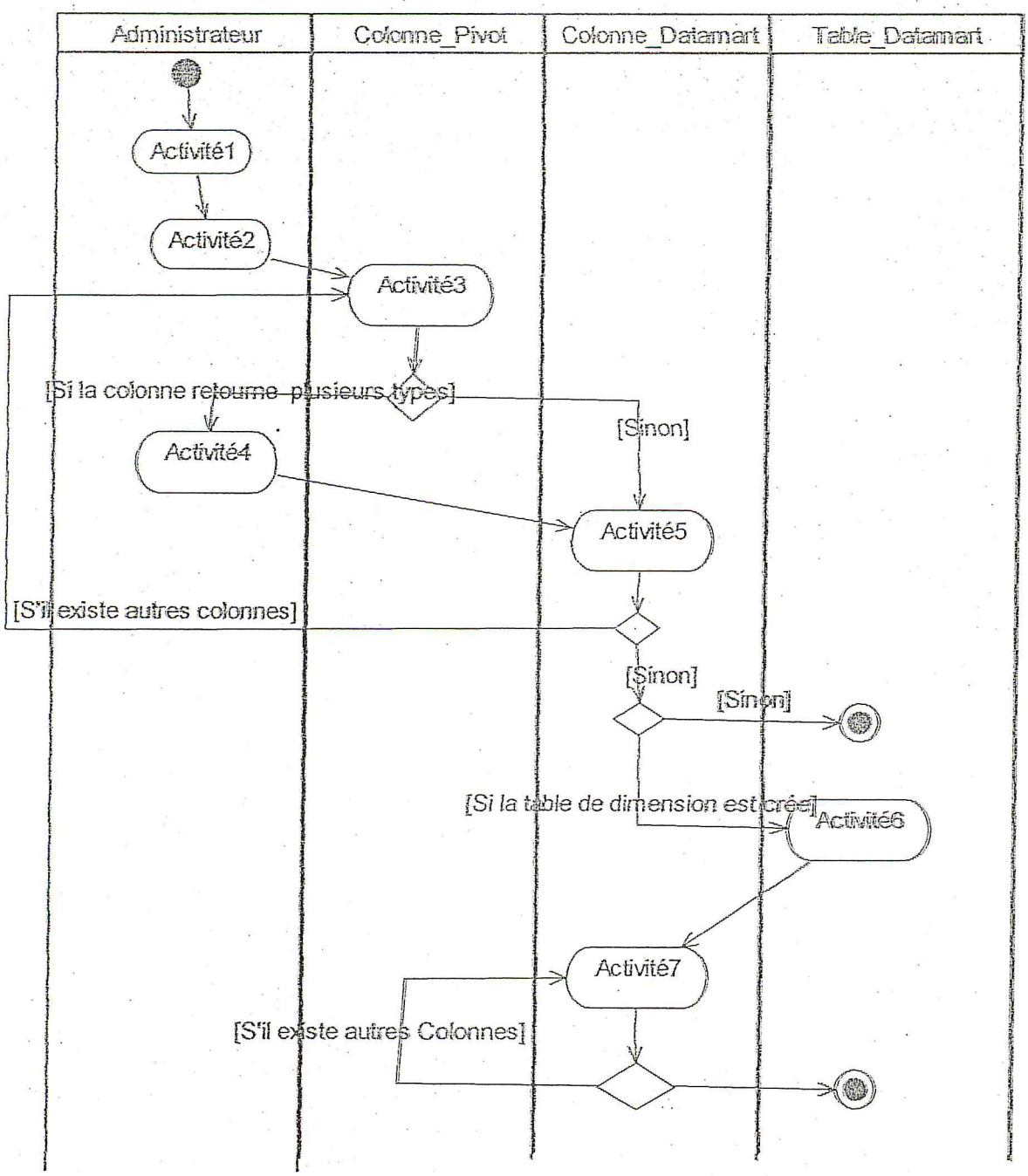


Figure 2.2.22 diagramme d'activité «Créer Table de dimension»

Indice	Description
Activité1	Donner un nom à la table de dimension
Activité2	Choisir les colonnes de datamart, à partir des colonnes Fédérées.
Activité3	Définir le type de la colonne
Activité4	Attribuer le type « varchar»à la colonne.
Activité5	Donner une longueur pour chaque type
Activité6	Créer Table_Datamart
Activité7	Créer Colonne_Datamart

Tableau 2.2.15 la description des activités pour le diagramme d'activité

«Créer Table de dimension»

2.4.9 Diagramme d'activité«Créer Table de fait»

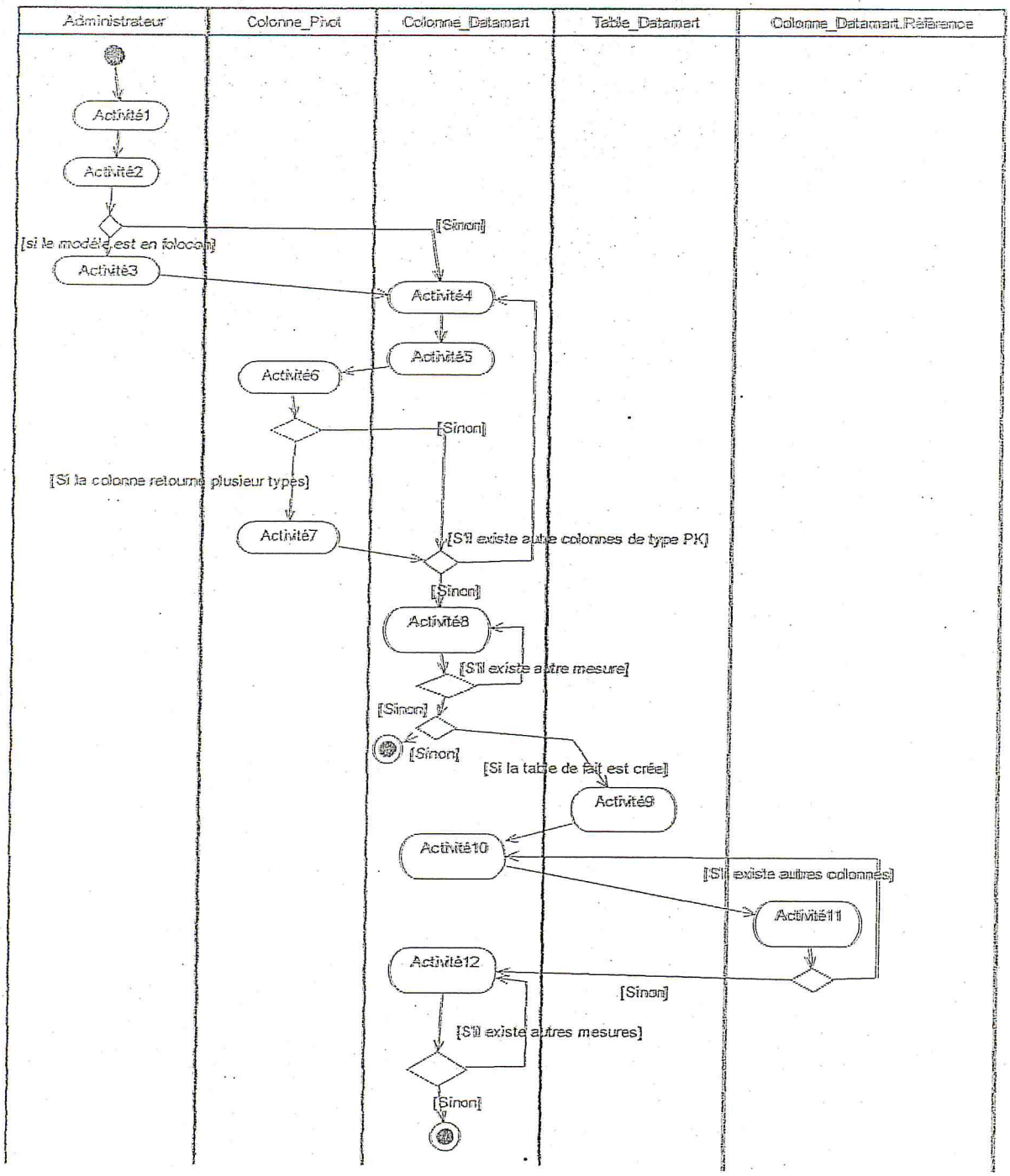


Figure 2.2.23 diagramme d'activité «Créer Table de fait»

Indice	Description
Activité1	Donner un nom à la table de fait
Activité2	Choisir les paramètres d'une colonne mesure de fait.
Activité3	Sélectionner les tables en relation avec la table de fait
Activité4	Rechercher la colonne de dimension de type PK.
Activité5	Renommer colonne(Colonne_Table).
Activité6	Rechercher le type, la longueur pour chaque colonne de dimension de type PK.
Activité7	Attribuer le type « varchar » a la colonne
Activité8	Ajouter le Type numeric et la longueur 38 à colonne mesure de fait.
Activité9	Créer Table_Datamart.
Activité10	Créer les Colonne_Datamart.
Activité11	Créer les Colonne_Datamart.Référence.
Activité12	Créer les Colonne_Datamart (les colonnes mesure).

Tableau 2.2.15 la description des activités pour le diagramme d'activité

«Créer Table de fait»

2.5 Diagramme de déploiement

Nous allons maintenant décrire l'implémentation de notre application grâce au diagramme de déploiement.

Le diagramme de déploiement montre la connexion de l'application cliente avec les bases de données réparties sous les SGBD SQL Server et PostgreSQL, et les documents XML.

La station d'administration ETL Data est connectée à un serveur SGBD SQL Server (Serveur ETL Data), par l'API. JDBC, cette connexion permette à l'administrateur de l'entrepôt de données , de créer son Catalogue_ETL , et même les datamart

Pour accéder à des bases de données implémentées sur PostgreSQL et extraire leurs schémas il suffit de connecter l'application ETL Data au SGBD PostgreSQL par JDBC PostgreSQL.

Pour accéder à des bases de données implémentées sur SQL Server et extraire leurs schémas il suffit de connecter l'application ETL Data au SGBD SQL Server par JDBC SQL Server.

Pour alimenter les datamarts , Notre système fait la migration des bases de données ,qui sont implémenté sur le SGBD de SQL Server ou PostgreSQL sur le serveur ETL Data.

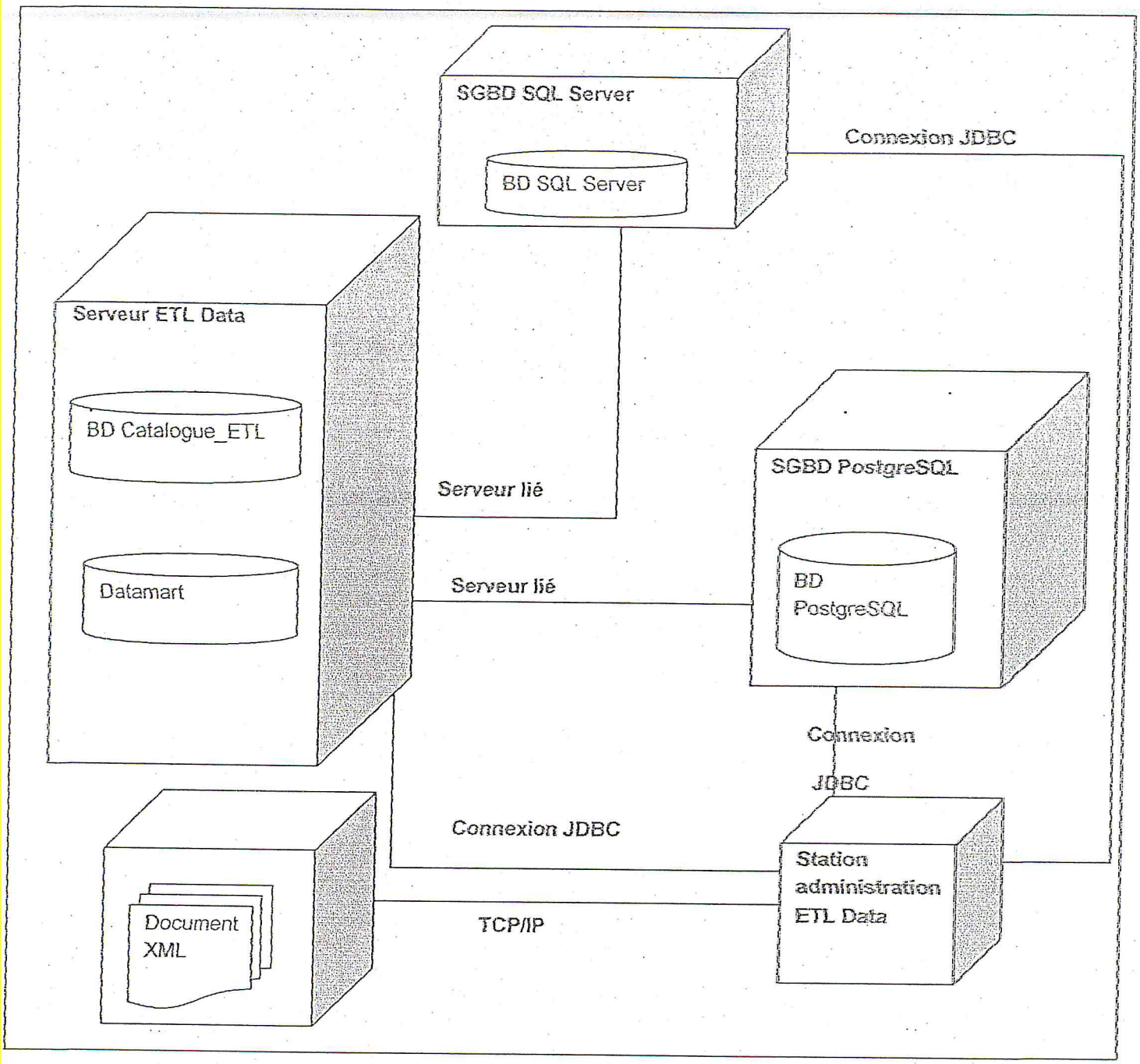


Figure 2.2.24 diagramme de déploiement

Conclusion

Nous avons illustré dans ce chapitre les grandes fonctionnalités offertes par notre système d'intégration qui repose sur l'architecture fédérée fortement couplet par le biais des diagrammes UML.

Partie 3

Implémentation

Introduction

L'étape d'implémentation est considérée comme le baromètre de sucés des étapes précédentes. C'est à dire si les étapes de l'analyse et de la conception sont bien faites, les programmeurs serons plus à l'aise.

Chapitre 1 Processus d'alimentation

Introduction

Dans cette partie nous allons décrire en détail les étapes du processus de traitement de la requête d'alimentation d'une table datamart à savoir l'analyse, la localisation des sources, la réécriture de la requête d'alimentation afin d'obtenir un résultat, pour alimenter la table de fait et les tables de dimensions du datamart. Dans cette étape on s'est basé sur les travaux [SAL 08].

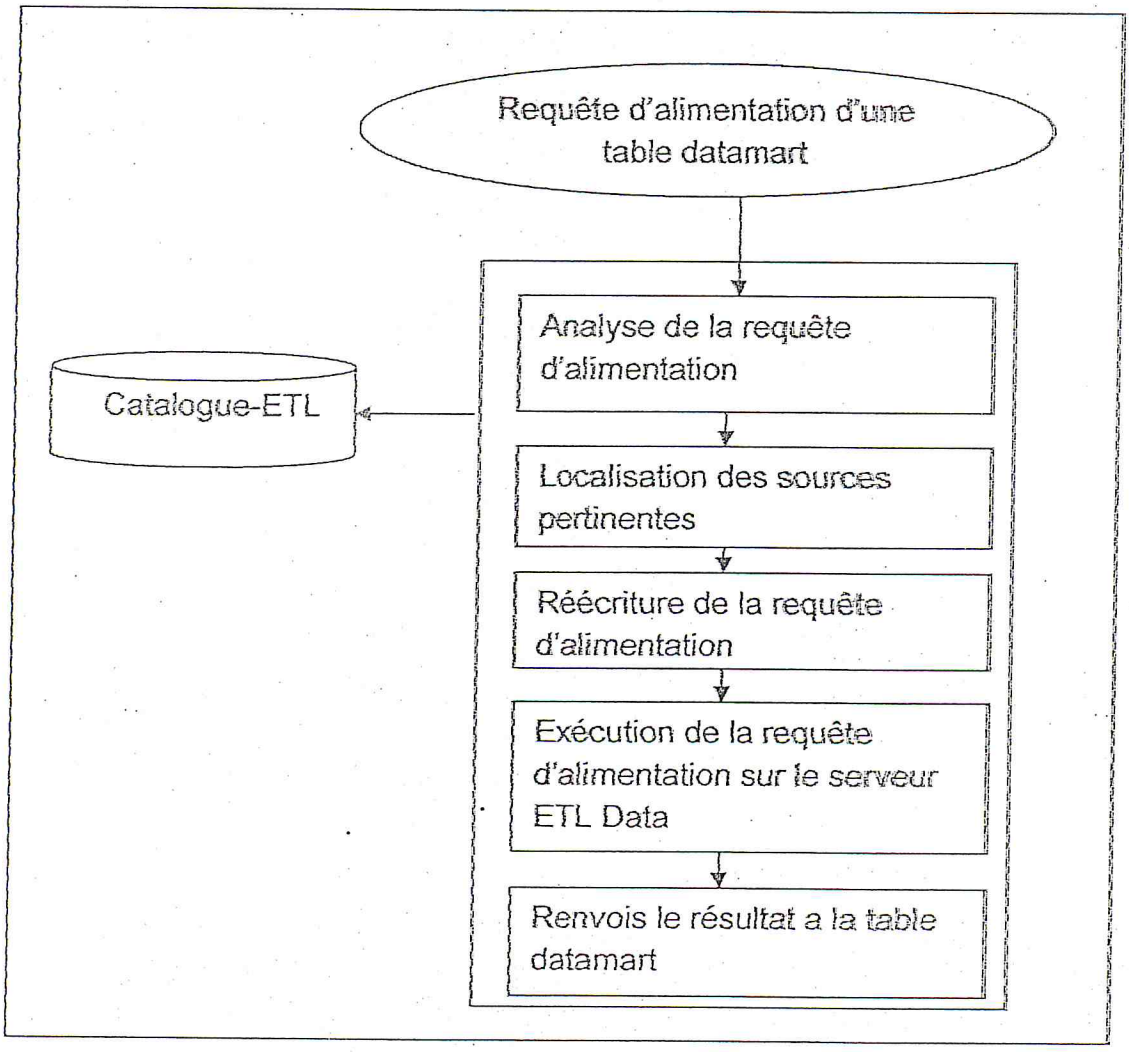


Figure3. 1.1 Architecture du processus de traitement de requêtes d'alimentation

1. Architecture du processus de traitement de requêtes d'alimentation

1.1 L'analyse de la requête

C'est la première étape dans le processus du traitement de la requete d'alimentation, elle consiste à

- Vérifier le modèle du datamart à alimenter, si le modèle du datamart est en étoile on commence par les tables de dimensions puis la table de fait. Mais si le modèle est en flocon de neige, il faut donner un ordre aux tables de dimensions, (on commence par l'alimentation des tables qui comportent les clés primaires puis les tables référencies).
- Retirer les différents éléments (attributs à projeter d'une table datamart) ; elle a pour but de localiser les sources pertinentes en déterminant les propriétés des sources locales sémantiquement équivalentes aux éléments extraits de la requête d'alimentation. Elle est considérée comme une étape préalable aux étapes suivantes.

1.2 La localisation des sources pertinentes

Cette étape qui vient après l'analyse consiste en la détermination des sources pertinentes capables de donner une réponse partielle ou totale à la requête d'alimentation. Pour réaliser cela notre système doit accéder au catalogue-ETL qui relie chaque attribut d'une table datamart à tous les attributs des autres schémas (Fédéré,exports,pivots) qui lui sont sémantiquement équivalents dans les sources locales. Pour se faire nous suivrons la procédure suivante :

Soient « Ep » l'ensemble des propriétés (attributs) à projeter de la requête d'alimentation et « Es » l'ensemble des propriétés (attributs) des sources..

Si ($Ep \cap Es \neq \emptyset$) (il existe des attributs communs entre les attributs projetés et ceux de la source locale Si) :

Dans ce cas la source locale Si est pertinente.

Sinon (Ep ∩ Es = ø) (il n'existe aucun attribut commun entre les attributs projetés et ceux de la source locale Si)

Dans ce cas la source locale Si n'est pas pertinente.

1.3 Réécriture de la requête

Après la localisation des sources pertinentes on procède à la réécriture de la requête d'alimentation, on utilisation le langage de requête propre à SQL Server « Transact-SQL»

On distingue trois cas dans la réécriture de la requête d'alimentation :

➢ **Cas 1**

Les sources locales contiennent tous les attributs nécessaires pour alimenter une table datamart.

Dans ce cas la réécriture de la requête sur la source concernée comporte les étapes suivantes:

1. Remplacer toutes les propriétés projetées de la requête d'alimentation par les attributs correspondants dans la source locale.

Si la table à alimenter est de type fait on ajoute les fonctions d'agrégats aux colonnes de mesure de fait.

2. Dans la partie **FROM** de la requête d'alimentation on écrit toutes les tables contenant au moins un élément des attributs correspondant à la table de datamart.

Il existe trois façons d'écrire les tables dans la clause **FROM**

Si le type de la source de données est une Base de données relationnelle **alors**

Si le type de SGBD source est SQLServer **alors**

Nom_Serveur.Nom_Source .dbo.Nom_Table *as* a

Sinon

Si le type de SGBD source est PostgreSQL **alors**

OpenQuery ("Nom_Serveur", ' SELECT * From Nom_Schéma."Nom_Table" ') as a

Sinon

Si le type de la source de données est un document XML alors

openXML (@idoc, '/Nom_Source/Nom_Table',2) WITH (Nom_Colonne1 varchar (30), Nom_Colonne2 varchar (30)....etc.)

- Si la table de datamart était alimenté à partir de plusieurs tables de la mêmes source de données, On ajoute le **ON** à la clause **FROM**. On insère dans la partie **ON** les conditions de jointures entre les tables sources.

  Dans cette étape on distingue 2 cas :

  - *Jointure entre deux tables avec clé étrangère*

    Alors dans ce cas on doit ajouter la jointure correspondante dans la partie **ON** de la Requête pour sélectionner les données. Les deux clés étrangères composant cette jointure seront déterminées à partir du Catalogue_ETL du notre système grâce à la classe réflexive «Colonne_Pivot.Référence ».

  - *jointure entre deux tables avec table d'association*

    Alors dans ce cas on doit insérer dans la partie **FROM** de la requête locale le nom de la table d'association concernée par la jointure et dans sa partie **ON** les jointures correspondantes.

3. Si la table a alimenté est de type fait on ajoute une clause **GROUP BY** on insère, les colonnes des clés primaires de la table fait.

Exemple

Dans cet exemple nous verrons les étapes de la réécriture de la requête d'alimentation sur une source comportant toutes les propriétés projetées de la requête d'alimentation (ou bien tous les attributs d'une table datamart).

Soit la source S1 (sur le SGBD SQL Server) avec les tables suivantes :

Etudiant (**num_etud**, nom_etud, prenom_etud, num_faculte)

Livre (**ISBN**, titre_livre, nom_aut, annee, edition)

Auteur (**num_aut**, nom_aut, prenom_aut)

Faculte (**num_fac**, nom_fac)

Emprunter (**num_etud, ISBN**)

- Et soit la table de dimension Étudiant contient les attributs suivants :

  Nom_Etudiant, nom_ Faculté,titre_Livre.

D'âpres le Catalogue_ETL, on obtient les tables qui contiennent les attributs de la table datamart dans la source S1(Étudiant, faculté et livre)

On vérifie dans la table « Colonne_Référence »du notre catalogue_ETL les relations entres ces tables (Étudiant, faculté et livre)

On obtient la jointure entre la table faculté et étudiant

etudiant.num_faculte= faculte.num_fac,

Et on obtient aussi une nouvelle table « emprunter » en relation avec la table Étudiant et faculté.Alors dans ce cas on doit ajouter dans la partie from l'association « emprunter » nous aurons donc les deux jointures suivantes .

emprunter.num_etud = etudiant.num_etud

emprunter.ISBN = faculte.ISBN

Nous aurons la requête d'alimentation suivante.

```
SELECT Distinct a.nom_etud,b.nom_fac,d.titre_livre
FROM Serveur. Source1.dbo.etudiant as a
JOIN Serveur . Source1.dbo.faculte as b
ON a.num_faculte =b.num_fac
JOIN Serveur .Nom_Source.dbo.emprunter as c
ON a.num_etud =c.num_etud
JOIN Serveur . Source1.dbo.livre as d
ON  d.ISBN = c.ISBN
```

- Et pour la table de fait, Si on veut savoir par exemple le nombre de livre emprunter pour chaque étudiant

```
SELECT a.nom_etud ,c.titre_livre ,COUNT(c.ISBN)
FROM Serveur. Source1.dbo.etudiant as a
JOIN Serveur .Source1.dbo.emprunter as b
ON a.num_etud =b.num_etud
JOIN Serveur. Source1.dbo.livre as c
ON  b.ISBN = c.ISBN
GROUP BY a.nom_etud ,c.titre_livre
```

- Cas 2

Les sources locales contiennent juste une partie des attributs nécessaires à la satisfaction de la requête d'alimentation d'une table datamart mais peuvent être complétés par des attributs d'autres sources.

Pour traiter ce cas nous devrons concevoir une procédure qui détermine pour chaque source de données les propriétés manquantes qui peuvent être complétées à partir des autres sources.

Ajouter un attribut « a » à la source Si à partir de Sj n'est possible que si la règle suivante est vérifiée :

Le principe de jointure entre sources :

On peut importer les colonnes de datamart manquantes de S2 vers S1 s'il existe une relation dans le schéma fédéré entre la table FS1 et la table Fédéré FS2

(FS1 Table Fédérée qui contient les colonne de S1et FS2 Table Fédérée contient les colonnes de S2),cette relation est déterminé, à partir du Catalogue_ETL du notre système grâce à la classe réflexive «Colonne_Fédéré.Référence ».

Exemple

Soient les deux sources suivantes :

(S1 sur le SGBD SQL Server et S2 est sur le SGBD PostgreSQL)

S1 : auteur (num_a, nom_a, prenom_a,adresse, num_établissement)

.

.

S2 : author (idf_a, address, telephone, num_e)

Etablissement(num_e,name_e,adresse_e)

.

.

.

Schéma Fédéré :

Auteur(num_a,nom_a,prenom_a, adress, telephone, num_e#)

Etablissement(num_e,name_e,adresse_e)

- Et la table de dimension «Auteur» contient les attributs suivants : Nom_ Auteur ,Adresse, Nom_ Etablissement

On remarque que la source « S1 » ne répond que partiellement aux attributs de la table de dimension car elle ne contient pas d'attribut équivalent à la propriété «Nom_ Etablissement »et à la propriété Adresse.

La première étape consiste à rechercher les attributs de la table datamart puis écrire la requête d'alimentation « Q » avec les propriétés existantes dans « S1 » en suivant exactement les mêmes étapes que pour le 1er Cas.

Puis on doit rechercher les colonnes datamart manquantes et on va vérifier la condition de jointure dans le schéma fédéré, s'il existe une relation entre la table auteur et établissement, alors on peut appliquer les jointures entres les deux sources S1,S2.

Nous aurons une requête d'alimentation entre la source1 et la source2

```
Select Distinct a.nom_a,b.adresse, b. nom_etablissement
FROM Serveur1.Source1.dbo.auteur as a
JOIN OpenQuery (Serveur2, ' SELECT * From public."etablissement" ')as b
ON a.num_établissement=b.num_e
```

- **Cas 3**

Les sources locales contiennent juste une partie des attributs nécessaires à la satisfaction de la table datamart et ne peuvent pas être complétés par d'autres attributs à partir d'autres sources.

La source interrogée ne répond qu'à une partie des propriétés de la requête globale et les propriétés manquantes ne peuvent pas être complétées à partir des autres sources avec des attributs manquants (c.-à-d. les conditions du « principe de jointure entre sources » cité auparavant ne sont pas satisfaites).Dans ce cas on sélectionne seulement à partir de la source qui contient la clé primaire de la table datamart.

Exemple

(S1 : un document XML contient seulement des informations sur les étudiants,

S2 : une base de données implémenter sur le SGBD PostgreSQL).

S1 :

```
<?xml version="1.0" encoding="iso-8859-1"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="table">

        <xsd:complexType>

                <xsd:sequence maxOccurs="unbounded">

                        <xsd:element name="etudiant">

                                <xsd:complexType>

                                        <xsd:sequence>

                                                <xsd:element name="Num " type="xsd: positiveInteger " />

                                                <xsd:element name="Nom " type="xsd: string "/>

                                                <xsd:element name="Prenom " type="xsd: string "/>

                                                </xsd:sequence>

                                </xsd:complexType>

                        </xsd:element>

                </xsd:sequence>

        </xsd:complexType>

</xsd:element>

</xsd:schema>
```

S2 :

etudiant (num_e, nom_e, prenom_e,adresse)

depart (num_depart, nom_depart)

appartient (num_e, num_depart)

.

.

Schema Fédéré

Etudiant (num_e,nom_e,prenom_e,adresse)

Depart(num_depart,nom_depart)

Appartient(num_e, num_depart)

- Et la table de dimension Étudiant contient les attributs suivants Num_Etudiant,Nom_Etudiant,Nom_ Departement.

Dans cet exemple on remarque que la source « S1» répond partiellement à la requête d'alimentation de la table de dimension Étudiant et la propriété manquante (departement.nom) ne peut pas être obtenue d'une autre source en utilisant la jointure car il n'existe pas d'attribut de liaison entre « S1 »et « S2».

Dans ce cas, Nous aurons une requête d'alimentation suivante

```
DECLARE @idoc int
DECLARE @doc varchar(8000)
SET @doc ='
<?xml version="1.0" encoding="UTF-8"?>
<etudiants>
        <document_name>etudiants doc</document_name>
        <etudiant>
      <Num>100</Num>
              <Nom>ALIMAZIGHI</Nom>
              <Prenom>HADJER</Prenom>
        </etudiant>
        <etudiant>
      <Num>101</Num>
              <Nom>ATTAF</Nom>
              <Prenom>SARAH</Prenom>
              </etudiant>
</etudiants>'
EXEC sp_xml_preparedocument @idoc OUTPUT, @doc
SELECT   Distinct a.Nom ,a.Prenom   FROM      OPENXML (@idoc,
'/etudiants/etudiant',2) WITH (Nom nvarchar(30),Prenom nvarchar(30)) as a
```

1.4 Exécution de la requête d'alimentation sur le serveur ETL Data

La réécriture de la requête vise à remplacer les attributs d'une table de datamart par ceux des sources locales, pour qu'elle puisse être exécutée au niveau du serveur ETL Data.

Remarque

Si la requête d'alimentation est alimenté à partir d'un document XML ,

La syntaxe de la requête de sélection est présentée de la manière suivante :

```
DECLARE @idoc int
DECLARE @doc varchar(8000)
SET @doc ='
//Document XML
'

EXEC sp_xml_preparedocument @idoc OUTPUT, @doc
Select.....FROM  OPENXML (.....) .....
```

Cette syntaxe crée une représentation interne de l'image XML à l'aide de sp_xml_preparedocument. Une instruction SELECT est alors exécutée sur la représentation interne du document XML, en utilisant un fournisseur d'ensembles de lignes OPENXML

1.5 Renvois le résultat aux datamart

Âpres l'exécution de la requête sur le serveur ETL Data, le résultat de la requête d'alimentation sera insérée dans les tables du datamart.

Conclusion

Dans ce chapitre on a décrit en détail le processus de traitement de la requête d'alimentation qui comprend l'étape d'analyse, localisation des sources, réécriture, exécution de la requête d'alimentation sur le serveur de l'administrateur afin d'obtenir un résultat homogène qui sera alimenté dans les tables datamart.

Dans ce qui suit on décrira les différents composants du prototype développé pour notre solution ainsi que tous les outils qui sont nécessaires à son fonctionnement et leurs emplacements dans l'architecture générale du prototype.

Chapitre 2 REALISATION

Introduction

Les outils de développement permettent de réduire considérablement la charge de travail: et par voie de conséquence, la vitesse de développement. Parmi ces outils on trouve les langages de programmation.

Dans ce chapitre, nous allons présenter l'environnement de développement de notre systéme et les différentes interfaces de notre application.

1. Présentation des outils de développement

1.1 Choix du langage de programmation(JAVA)

Le langage utilisé pour la réalisation du notre logiciel est le JAVA qui est un langage de programmation orienté objet développer par SUN. Le chois de ce langage du aux avantages suivants : [JER 08]

- Java est un langage orienté objets
- Java est extensible à l'infini
- Java est un langage à haute sécurité

Java est un langage simple à apprendre

Java est portable

De plus se qui concerne notre projet, JAVA fournit tous ce qui est nécessaire à la manipulation des bases de données tels que : établir une connexion vers une base de données , exécuter des requêtes à partir de code JAVA , insérer des données ....etc Et ceci est possible à l'aide de l'API JDBC que nous allons définir par la suite .

1.1.1 Les pilotes JDBC

JDBC est une API Java (ensemble de classes et d'interfaces défini par SUN et les acteurs du domaine des bases de données) permettant d'accéder aux bases

de données à l'aide du langage Java via des requêtes SQL. Cette API permet d'atteindre de manière quasi-transparente des bases de données comme PostgreSQL, SQL Server, MySQL, Oracle, Informix

Pilotes JDBC : L'ensemble des classes qui implémentent les interfaces spécifiées par JDBC pour un gestionnaire de bases de données particulier est fourni par le pilote JDBC.

- Les protocoles d'accès aux bases de données étant propres à chaque SGBD il y a donc plusieurs drivers (pilotes) pour atteindre diverses bases de données.
- Cette API est indépendante du SGBD utilisé de plus, elle bénéficie des avantages du JAVA , dont la portabilité du code, son indépendance de la base de données et de la plate forme sur laquelle elle s'exécute.

Pour qu'une application JAVA puisse interroger une base de données SQL Server ou bien postgreSQL, elle doit être en mesure de se connecter à celle-ci. Cette connexion est assurée par l'interface JDBC. [JER08]

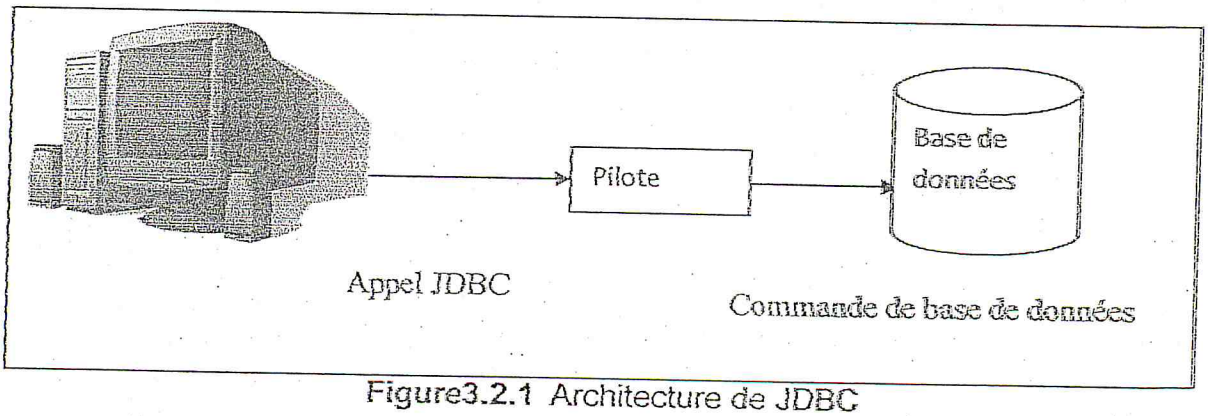


Figure3.2.1 Architecture de JDBC

1.1.2 Manipulation des documents XML

La manipulation d'un document XML se fait par le biais de parseur . Un parseur est une application dont le rôle est de convertir un flux de balisage en une sortie accessible par un programme. Le parseur vérifie la conformité d'un document à la norme XML, à savoir qu'un document XML doit être bien formé, et la validité de est associé à une DTD ou à un schéma

Le *parseur* permet donc de créer une structure hiérarchique contenants les données contenues dans le document XML. Nous distinguons deux types de *parseurs les parseur validant (validating)* permettant de vérifier qu'un document XML est conforme à sa DTD, et *les parseurs non validant (non-validating)* se contentant de vérifier que document XML est bien formé (c'est-à-dire respectant la syntaxe XML de base).

Il existe deux approches pour accéder à un document XML, les parseurs de callback et les parseurs de type arbre :

➢ **Les parseurs de type « callback »**

Les parseurs de type callback se basent sur un modèle événementiel, c'est à dire l'envoi d'événements à chaque élément rencontré dans le document XML. Cette technique est employée par l' API SAX (Simple API for XML). Cette approche ni cependant pas adaptée lorsqu'on désire modifier le document XML.

L'API SAX permet d'analyser un document XML basé sur le traitement séquentiel des données. Le parseur analyse le document et, lorsqu'il rencontre une balise ou un noeud, il renvoie un événement Un parseur SAX envoie le document sous forme de flux en examinant les différents éléments qu'il rencontre tout au long du document. Une application utilisant SAX implémente généralement des gestionnaires d'événements « handler », lui permettant d'effectuer des opérations selon le type d'élément rencontré. A partir de là, le parseur SAX continue l'analyse du document XML sans se préoccuper de cet élément.

Les applications basées sur SAX peuvent gérer uniquement les éléments dont elles ont besoin sans avoir à construire en mémoire une structure contenant l'intégralité du document.

➢ **Les parseurs de type « arbre »**

Les parseurs de type arbre représentent le document sous la forme d'une structure arborescente. Dans ce cas, le parseur charge le document XML en mémoire sous la forme d'un arbre et permet à l'utilisateur de modifier l'arbre à l'aide des interfaces définies par le W3C. Le DOM est le parseur de type arbre par excellence.

DOM *(Document Object Model)* est une spécification du W3C *(World Wide Web Consortium)* permet de définir la structure d'un document sous forme d'une hiérarchie d'objets, afin de simplifier l'accès aux éléments constitutifs du document. Plus exactement DOM est un langage normalisé d'interface (API, *Application Programming Interface)*, indépendant de toute plate-forme et de tout langage, permettant à une application de parcourir la structure du document et d'agir dynamiquement sur celui-ci.

1.1.3 Planification du datamart avec l'API Quartz

L'alimentation du datamart est faite en utilisant l'API Quartz qui va nous servir à rafraichir les données périodiquement dans le datamart .

Il existe plusieurs schedulers (classe Timer) disponibles pour la plateforme java mais les plus intéressants sont Quartz et Flux. Quartz dispose de nombreuses fonctionnalités qui conviendront à la plupart des applications métiers. Quartz fait partie du projet OpenSymphony qui propose des composants orientés entreprise pour la plateforme J2EE. Il est open-source, peut être redistribué avec modification des sources, et libre d'utilisation dans des projets commerciaux.[W7]

1.2. SGBD SQL Server

Pour l'implémentation de la base de données «Catalogue_ETL»et les différents datamart créer par l'administrateur, nous avons choisi le SGBD «SQL Server » version 2005 qui est un SGBDR offrant un certain nombre de fonctionnalités modernes telles que les requêtes complexes, les clés étrangères, l'intégrité des transactions et la gestion des accès simultanés à la base de données.

1.2.1 Le langage des requêtes de SQL Server T-SQL

Transact-SQL est au centre de l'utilisation de SQL Server. Toutes les applications qui communiquent avec une instance de SQL Server le font en envoyant au serveur des instructions Transact-SQL, quelle que soit l'interface utilisateur de l'application, dans la partie suivante on va définir quelques instructions utilisées lors de la création d'un datamart [MIC 00]

- **Création d'une base de données relationnelle**

Cette instruction permette de créer un datamart

```
Create database <nom de la base>
```

- **Définition d'un schéma relationnel**

Pour créer les différentes tables de dimensions et de fait d'un datamart , il faut définir le schéma relationnel de ces dernières .

```
CREATE TABLE <référence_table> (<liste d'elements>)
```

La liste d'éléments contient l'ensemble des attributs d'une table, avec le type et la longueur de chaque colonne.

Un attribut ou bien une liste d'attributs est déclaré comme un PRIMARY KEY

- **Contrainte de la clé étrangère**

La contrainte de la clé étrangère permette de créer une relation entre les différentes tables de dimension (dans le cas du modèle en flocon de neige), et même elle permet de définir les relations entre la table de fait et les différentes tables de dimensions

```
ALTER TABLE référence_table
ADD CONSTRAINT nom de la contrainte
FOREIGN KEY (<liste de colonnes de la clé étrangère>) REFERENCES <Nom de la table de la clé primaire> (<liste de colonnes de la clé primaire>)
```

- **Suppression la contrainte de la clé primaire ou bien la clé étrangère**

L'instruction suivante permette de supprimer une contrainte (clé primaire ou étrangère)

```
ALTER TABLE < référence_table > DROP constraint <Nom de la contrainte>
```

- **Requête de sélection d'une table**

La requête de sélection permette de sélectionner les données pour alimenter les différentes tables de dimension et de fait d'un datamart

```
SELECT [DISTINCT] <colonne1>,<colonne2>
FROM <référence_table1> join <référence_table2> ON [condition_jointure]
[GROUP BY[ liste_groupe]
```

L'opérateur DISTINCT permet de limiter les lignes retournées en supprimant les doublons dans la table de dimension.

La clause FROM permet de préciser la provenance des données (tables ou « sous requête »). cette clause permet aussi d'exprimer les jointures par JOIN...ON.

La clause GROUP BY est utilisée pour grouper les données dans la table de fait afin d'obtenir un résultat (en conjonction avec des fonctions d'agrégat de données comme SUM,AVG,COUNT).

- **Instruction INSERT**

L'instruction de mise à jour insert nous permette d'insérer de nouveaux enregistrements dans la table de fait et la table de dimension selon les colonnes de la table de fait et dimension

```
INSERT <table> [(col [, col ...])] VALUES (<val> [, <val> ...])
```

1.2.2. Liaison des serveurs

Pour que la requête d'alimentation puisse être exécuter sur le serveur d'administrateur, il faut lier les serveurs distants (postgreSQL et sql Server) au serveur d'administrateur (SQL Server).

Les serveurs liés offrent les avantages suivants [MS]

- l'accès aux serveurs distants.

- la possibilité d'émettre des requêtes, des mises à jour, des commandes et des transactions partagées sur des sources de données hétérogènes situées dans les différents services de l'entreprise .
- la possibilité de traiter diverses sources de données de manière identique.

Composants des serveurs liés

Une définition de serveur lié spécifie les objets suivants :

- Un fournisseur OLE DB
- Une source de données OLE DB

Un *fournisseur OLE DB* représente une DLL qui gère une source de données spécifique et interagit avec elle. Une *source de données OLE DB* identifie la base de données spécifique accessible via OLE DB. Bien que les sources de données interrogées au moyen des définitions de serveurs liés soient d'ordinaire des bases de données, des fournisseurs OLE DB existent pour différents fichiers et formats de fichiers

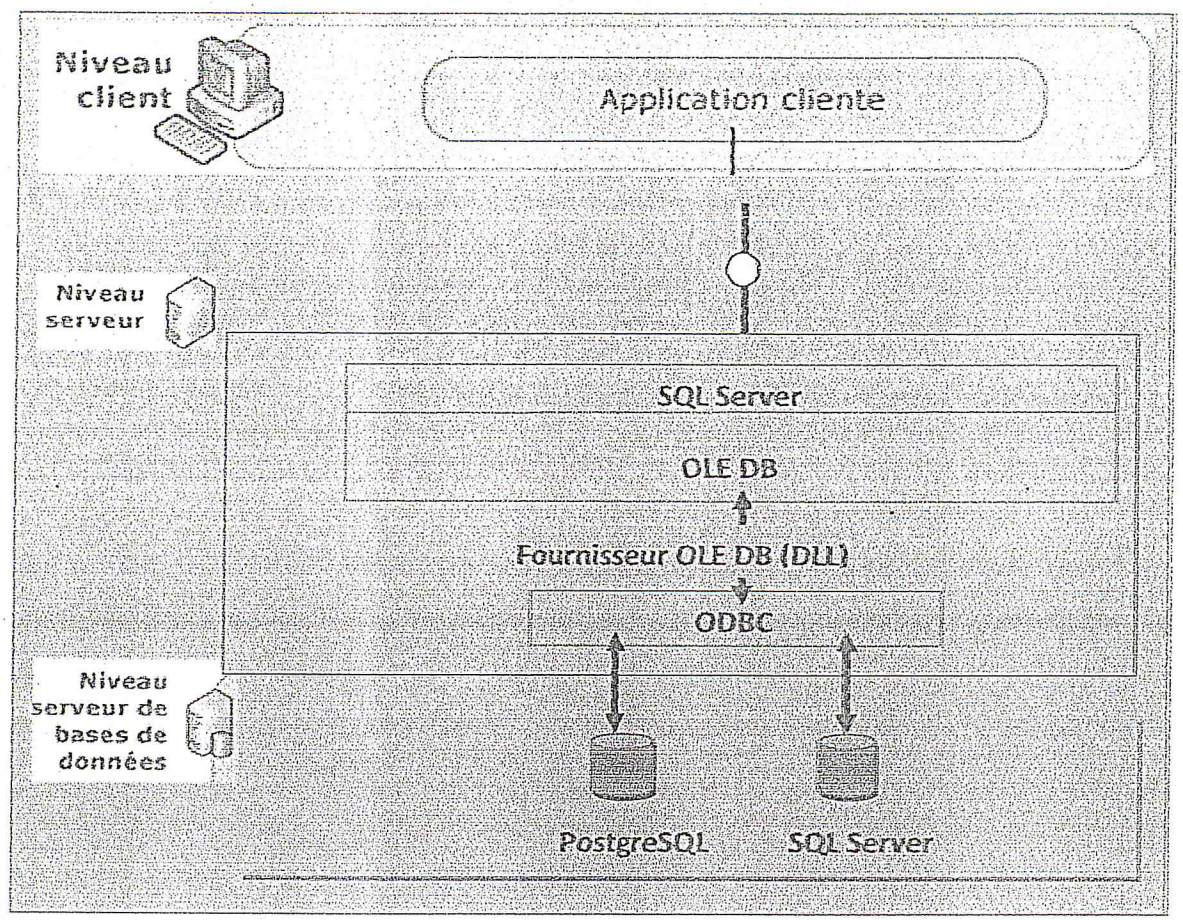


Figure3.2.2 configuration de serveurs liés [WV8]

1.3 Les ODBC

Open Database Connectivity (ODBC) est l'interface de la stratégie de Microsoft pour accéder aux données dans un environnement hétérogène du relationnel et non des bases de données relationnelles de gestion. Basé sur la spécification Call Level Interface du groupe d'accès SQL, ODBC fournit un ouvert, fournisseur neutre d'accès aux données stockées dans une variété d'ordinateurs personnels propriétaires, mini-ordinateur.

ODBC atténue la nécessité pour les vendeurs de logiciels indépendants et aux développeurs en entreprise à apprendre de multiples interfaces de programmation d'applications. ODBC fournit maintenant une interface universelle d'accès aux données.

Avec ODBC, l'administrateur de l'entrepôt de données peut, afficher des données provenant de multiples bases de données sous la plateforme des SGBD traités.[W4]

2. Présentation de l'application

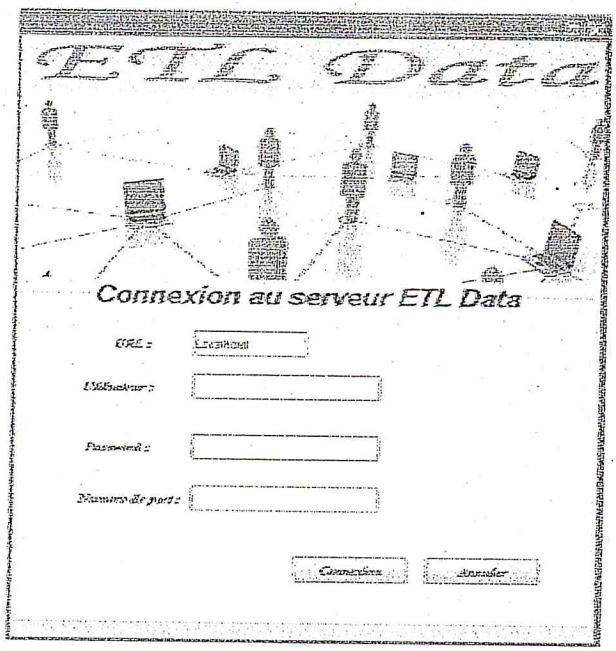


Figure 3.2.3 Connexion au serveur ETL Data

Cette première interface est une fenêtre de connexion au serveur ETL Data de notre système. A chaque connexion, le système va vérifier l'existence ou non d'un catalogue ETL. Si celui-ci existe, l'administrateur sera connecté sur le serveur, si non il aura le choix de la création d'un catalogue ETL ou de quitter l'application.

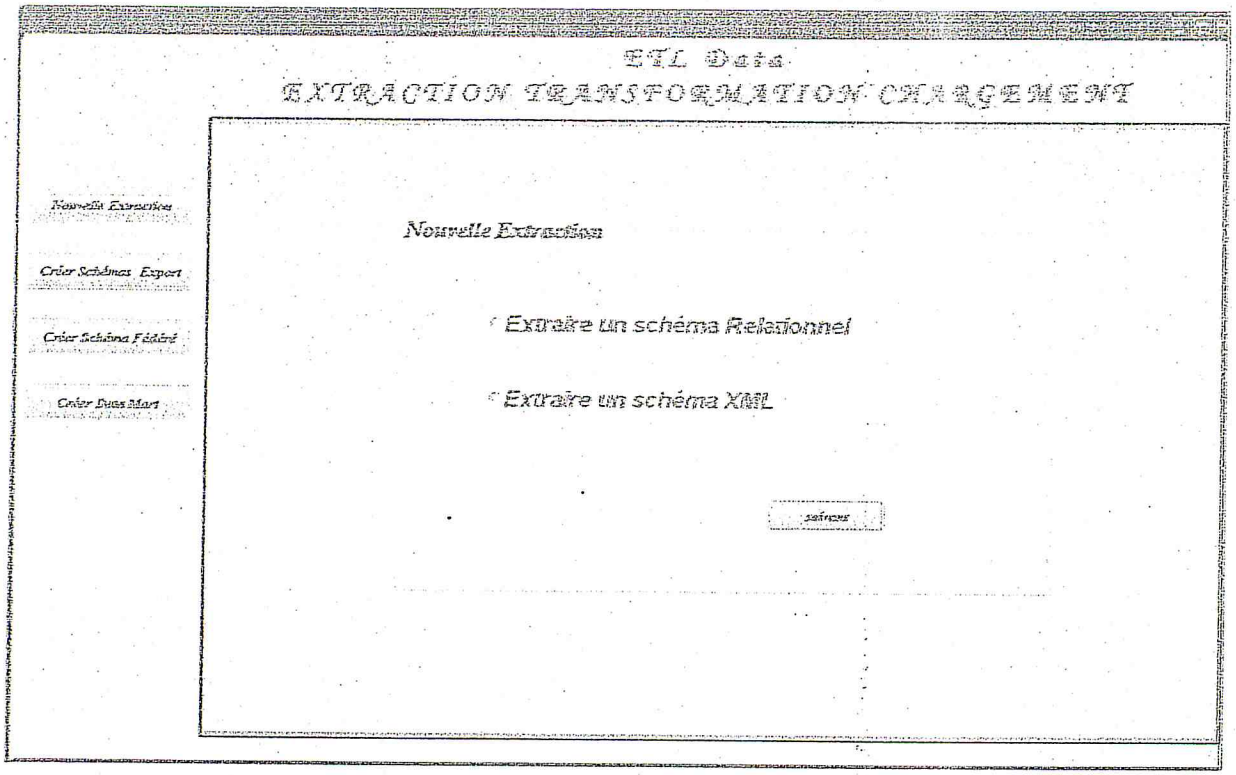


Figure 3.2.4 Choix du schéma

Cette interface donne à l'administrateur le choix d'extraire un schéma relationnel ou un schéma XML

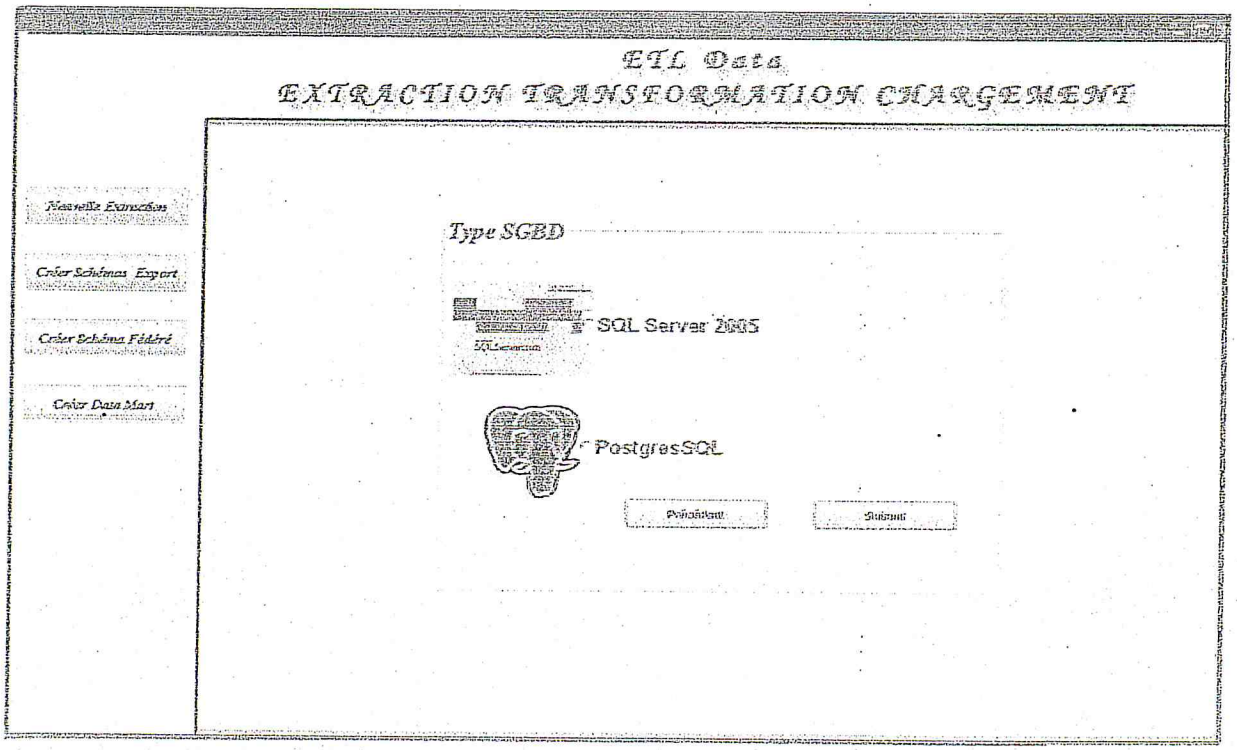


Figure 3.2.5 Choix du SGBD

Si l'administrateur choisit un schéma relationnel, le système lui demande le type de schéma qu'il souhaite : SQL Server 2005 ou PostgreSQL.

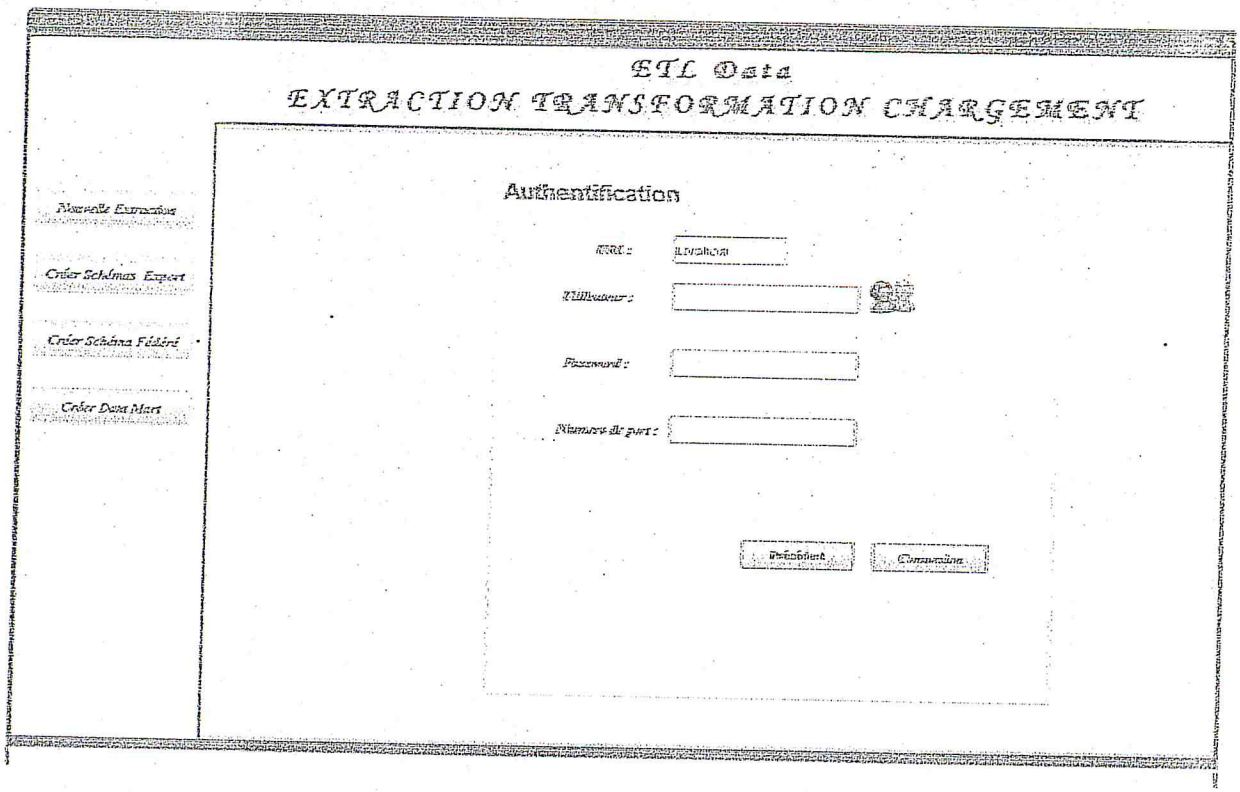


Figure 3.2.6 Connexion au serveur de la source de données

Après avoir choisi le SGBD, l'administrateur doit s'identifier auprès du serveur SQL en fournissant l'adresse du serveur, son nom d'utilisateur, son password ainsi que le port de connexion.

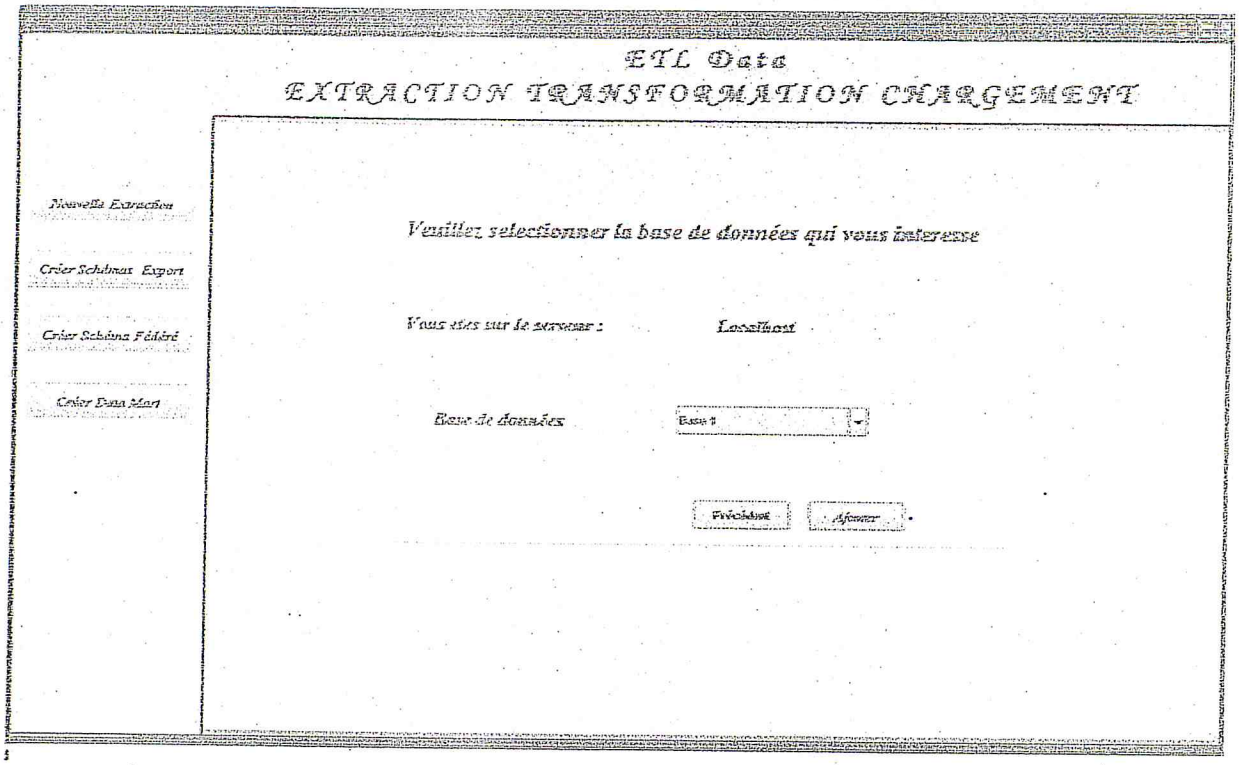


Figure 3.2.7 Extraction du schéma d'une source de données relationnelle

Cette fenêtre permet à 'administrateur de choisir la base de donnée qu'il souhaite extraire

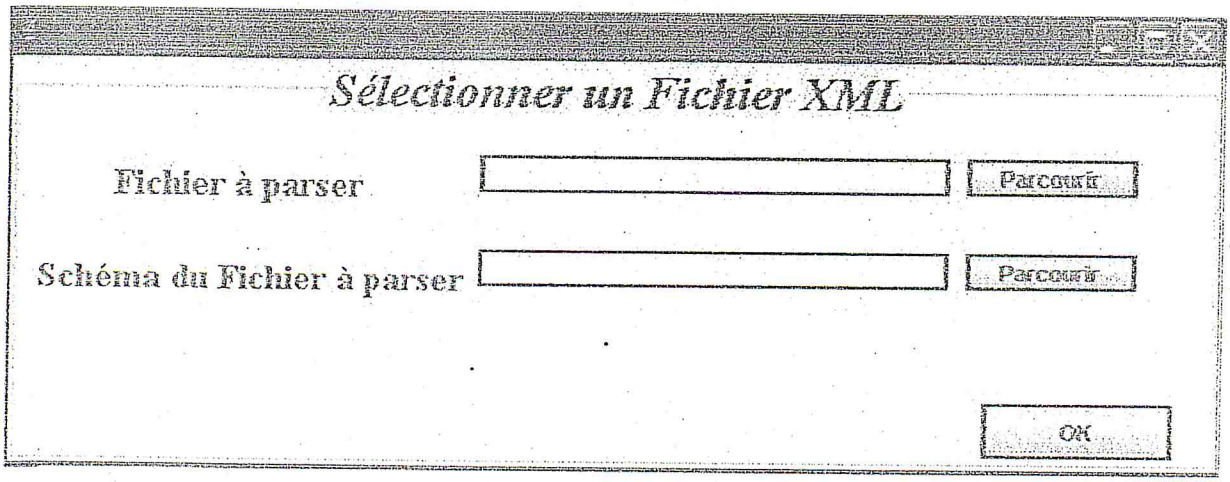


Figure 3.2.8 Extraction du schéma d'un document XML

Si l'administrateur fait le choix d'XML au lieu d'un schéma relationnel, cette fenêtre s'affiche pour lui demander de spécifier le chemin du fichier XML à extraire ainsi que son schéma XSD.

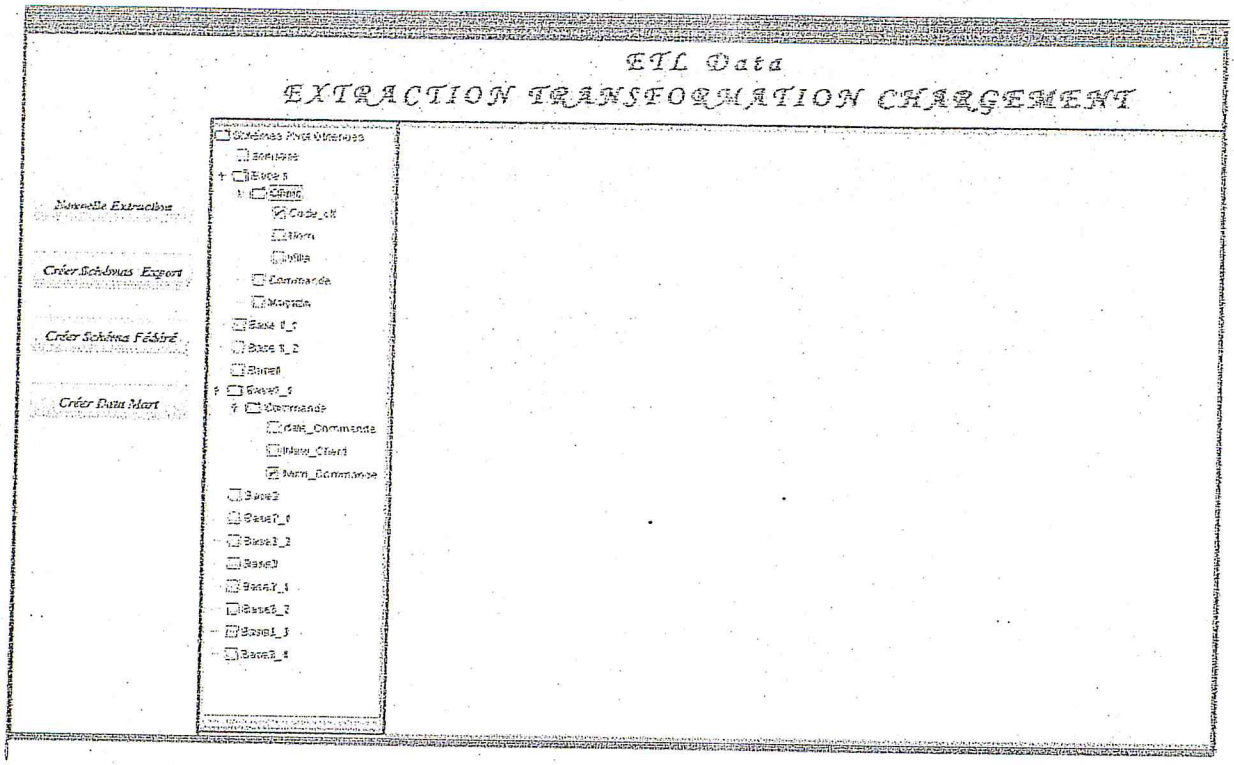


Figure 3.2.9Définir les schémas exports

Cette fenêtre apparaît lorsque l'administrateur souhaite créer un schéma Export.

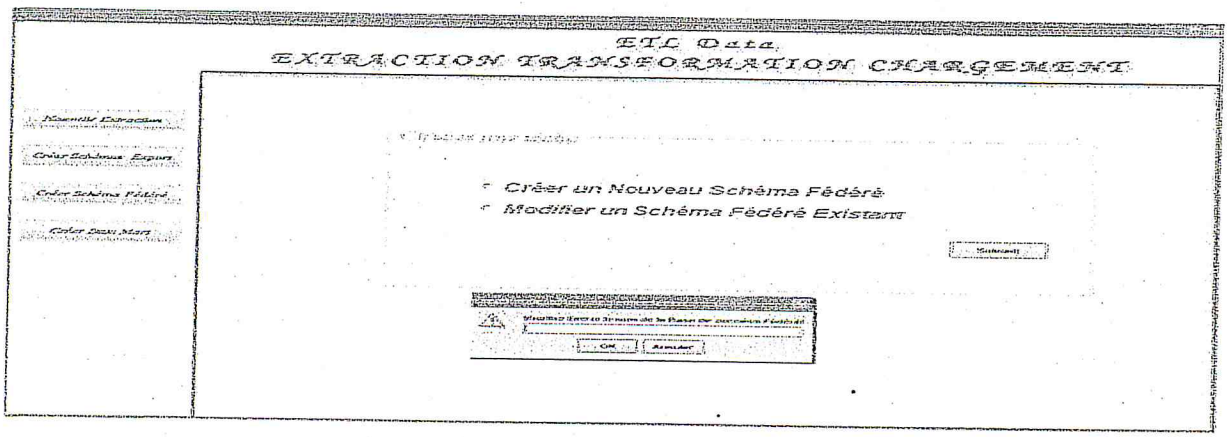


Figure 3.2.10 Générer un schéma fédéré

Si l'administrateur souhaite générer un schéma fédéré, cette fenêtre lui permet de choisir entre la création d'un nouveau schéma ou la modification d'un schéma fédéré existant.

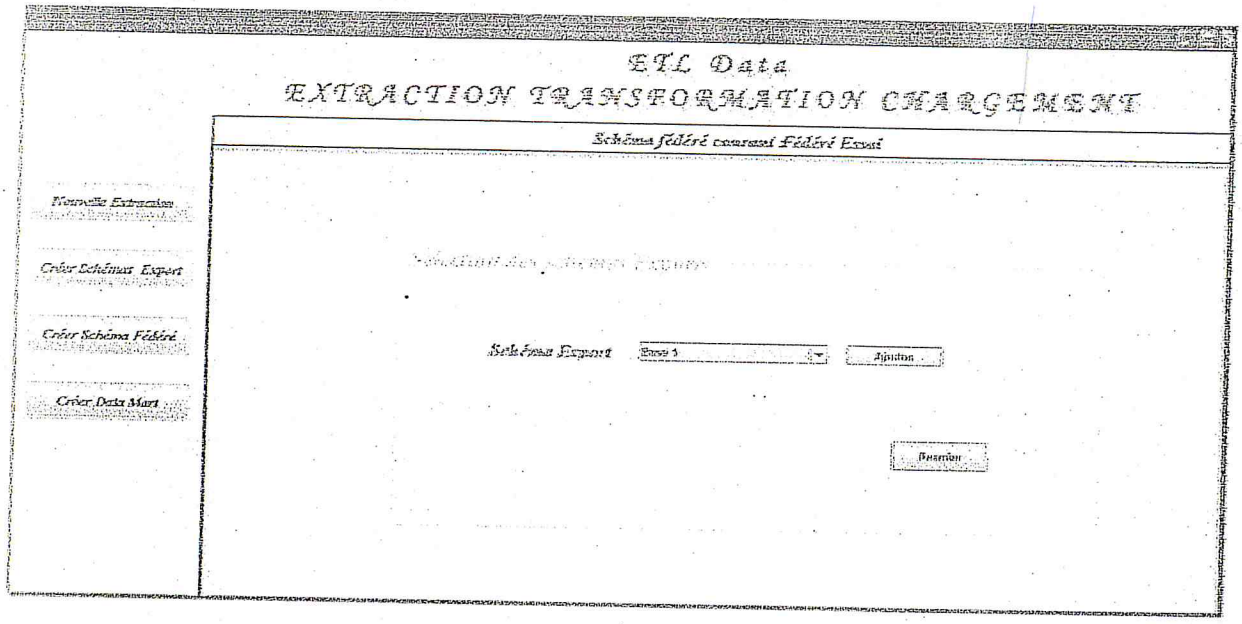


Figure 3.2.11 Sélectionner les schémas exports à intégrer

Après avoir choisi de créer ou de modifier un schéma fédéré, cete interface offre à l'administrateur la possibilité d'ajouter des schémas exports à son schéma fédéré.

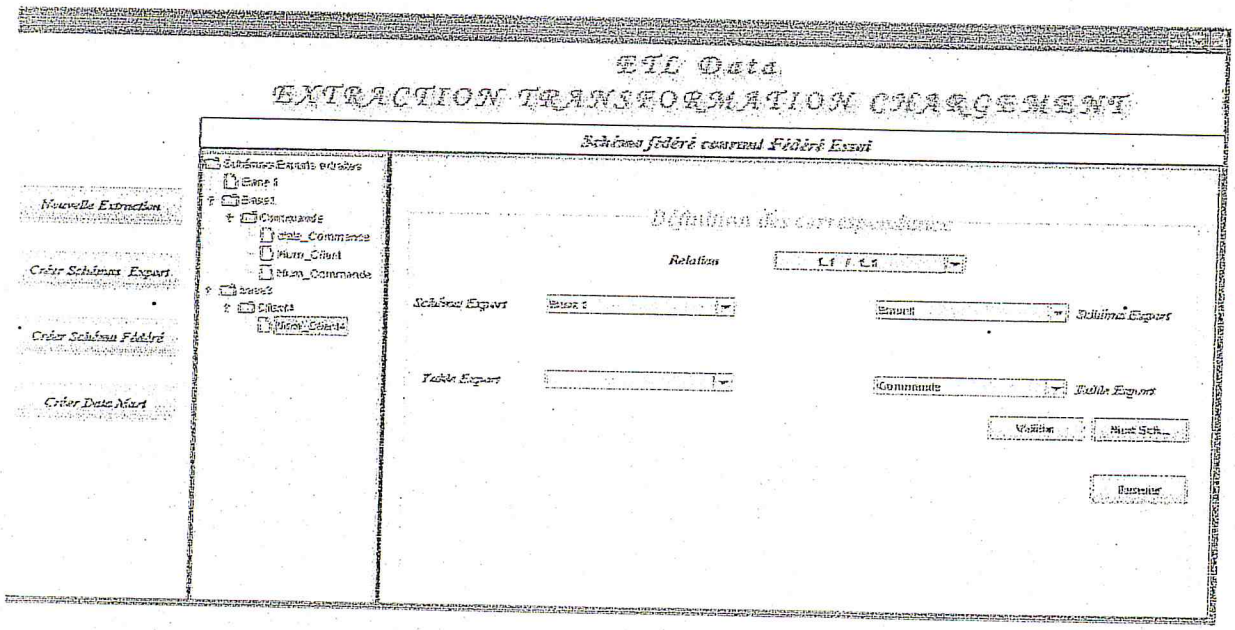


Figure 3.2.12 Définir les relations entre les table exports

Grâce à cette fenêtre, l'administrateur va définir les relations entre les tables des schémas exports déjà choisis pour être intégrés.

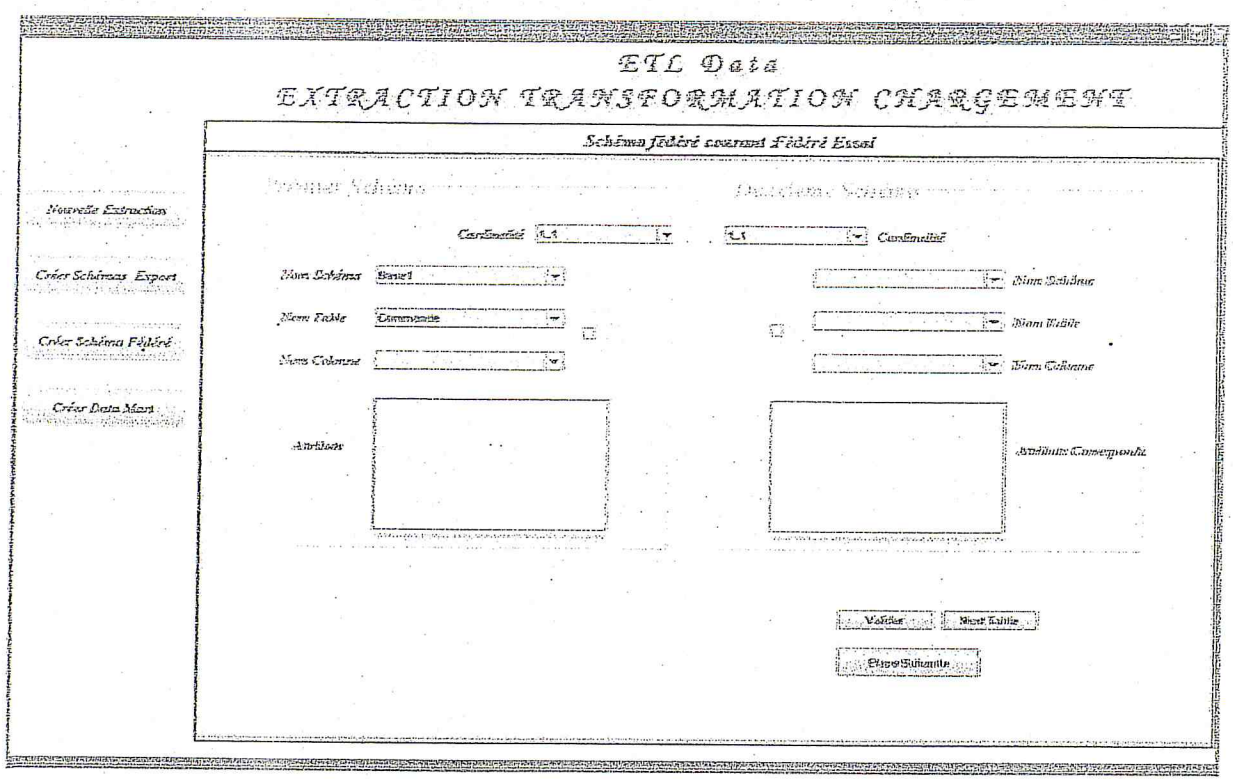


Figure 3.2. Résolution des conflits schématiques

Sur cette interface, l'administrateur doit définir les colonnes communes entre les tables export dont il a déjà défini les relations.

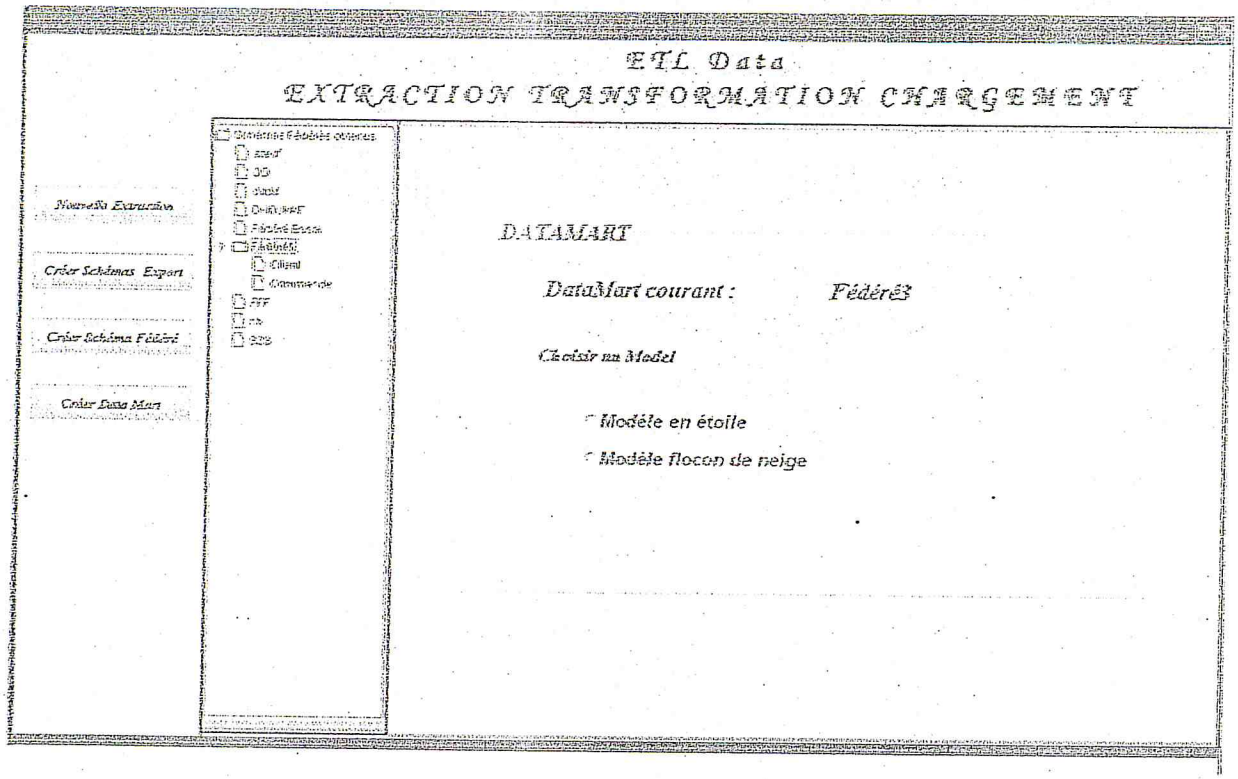


Figure 3.2.14 Modèle du datamart

Cette interface permet à l'administrateur de créer un nouveau datamart en choisissant un modèle pour ce dernier.

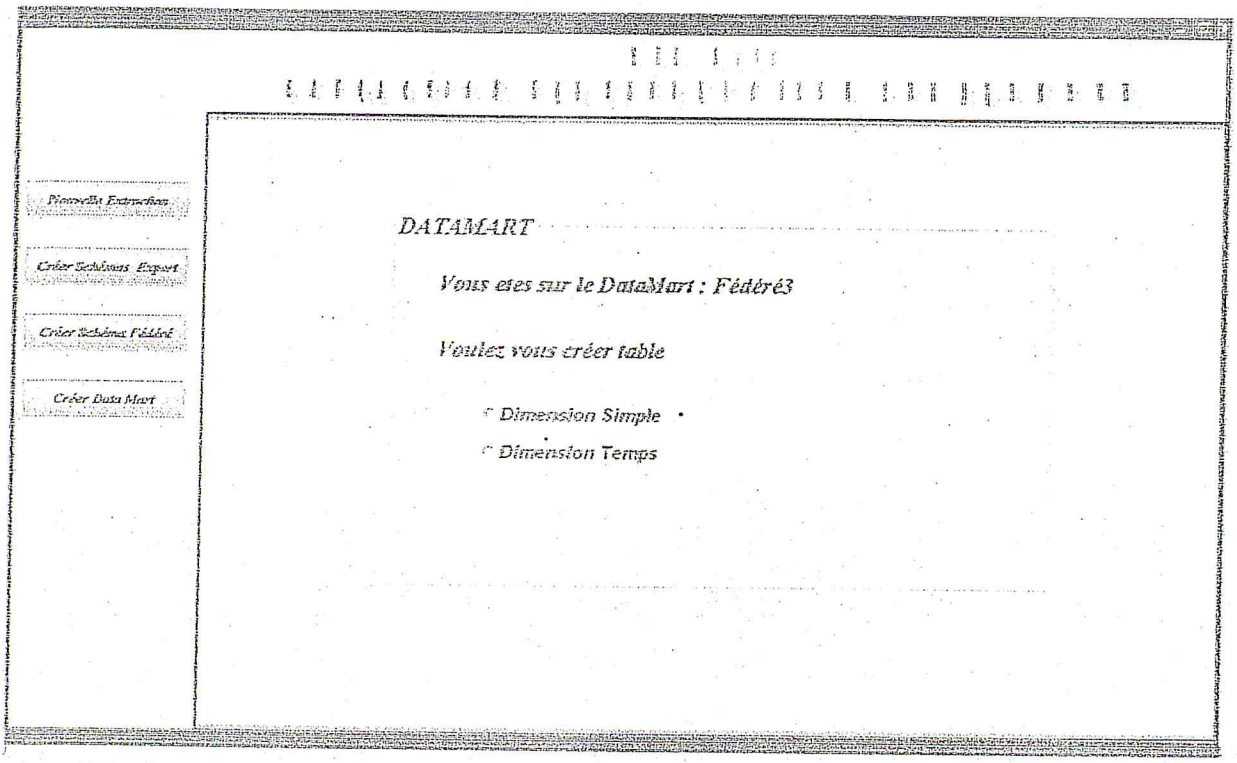


Figure 3.2.15 : Types de la Table dimension

Sur cette interface l'administrateur va choisir le type de la table de dimension, il l'a le choix de créer une table de dimension simple ou une table dimension temps.

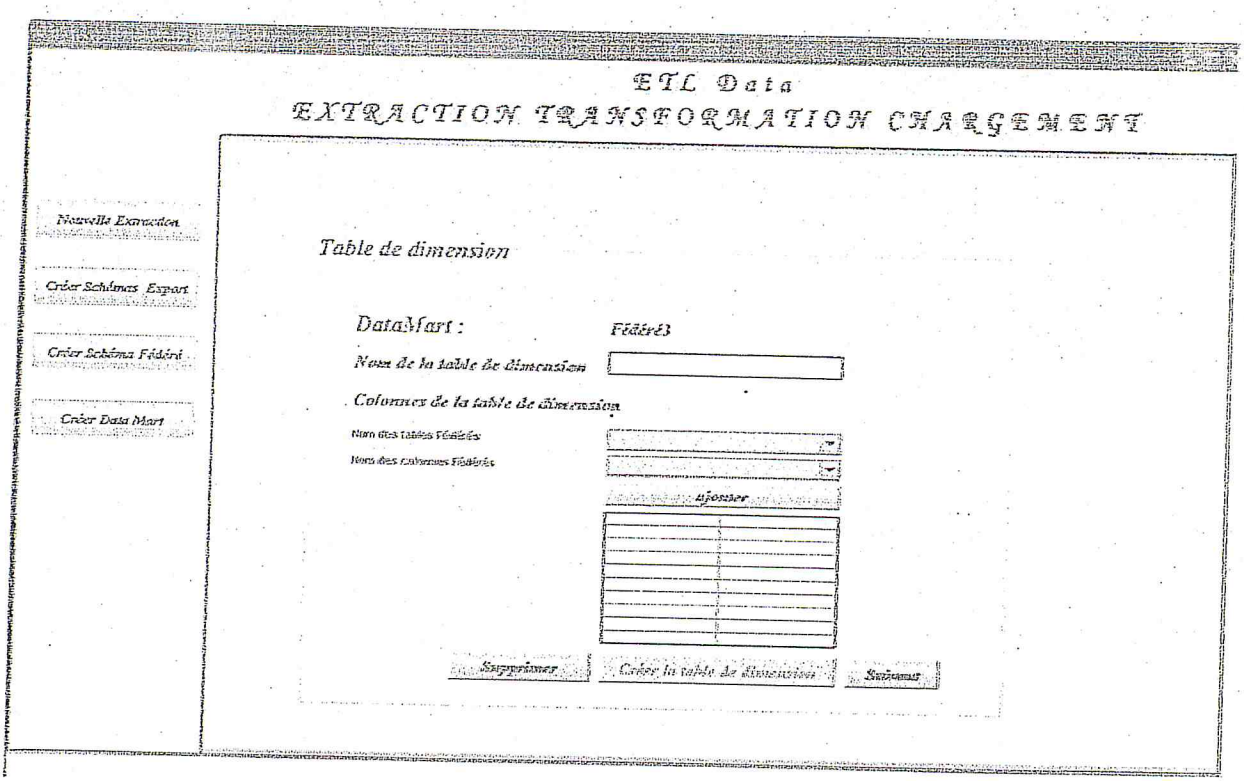


Figure 3.2.16 : Table de dimension Simple

Cette fenêtre permet à l'administrateur de créer une table de dimension

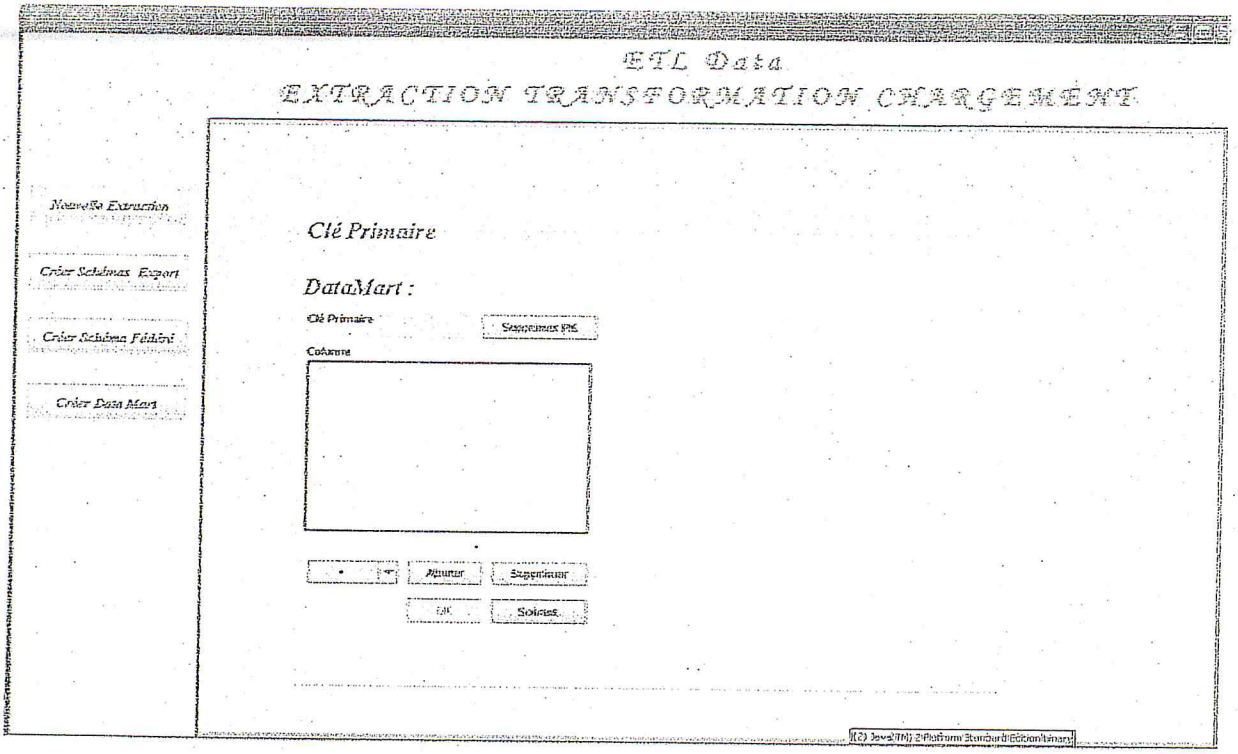


Figure 3.2.17 : Créer Clé primaire

Cette fenêtre permet à l'administrateur de créer la clé primaire, il peut même supprimer la contrainte s'il s'est trompé .

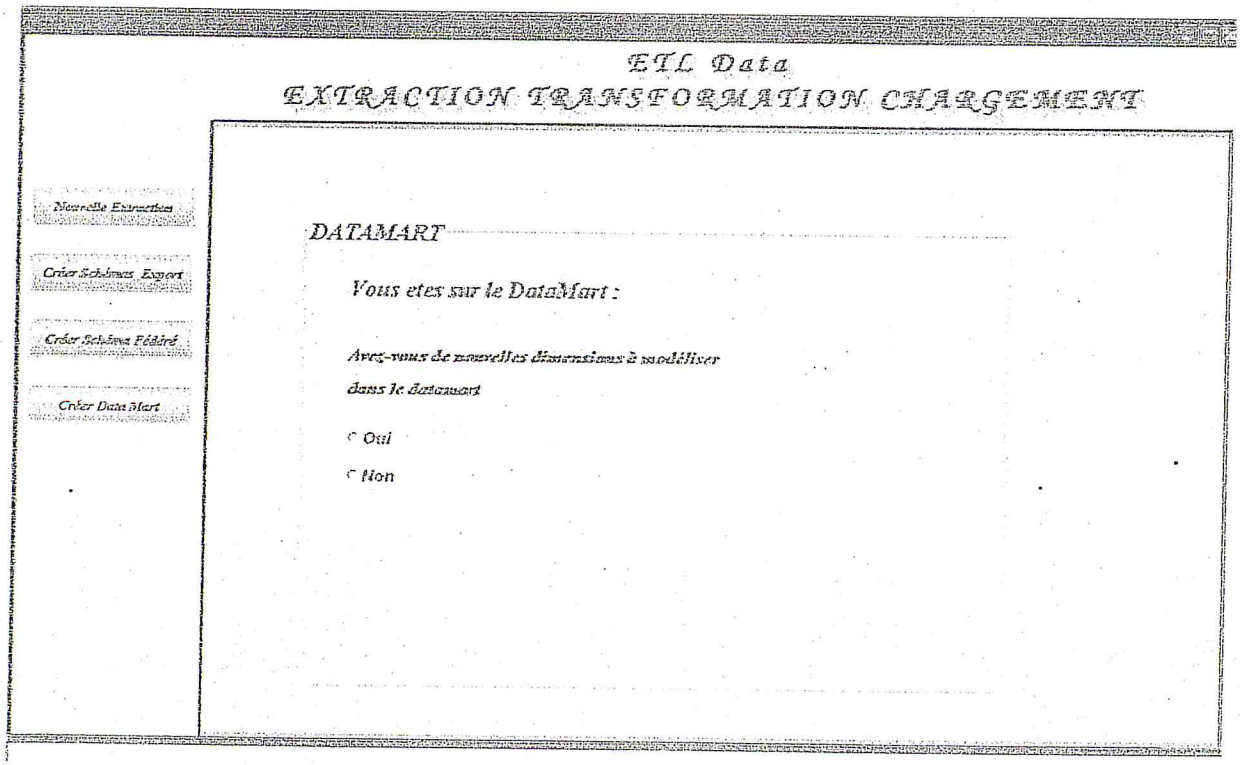


Figure 3.2.18 Ajout d'une nouvelle dimension

Si l'administrateur n'a aucune table de dimension à ajouter dans son datamart , il clique sur non , sinon , il clique sur oui.

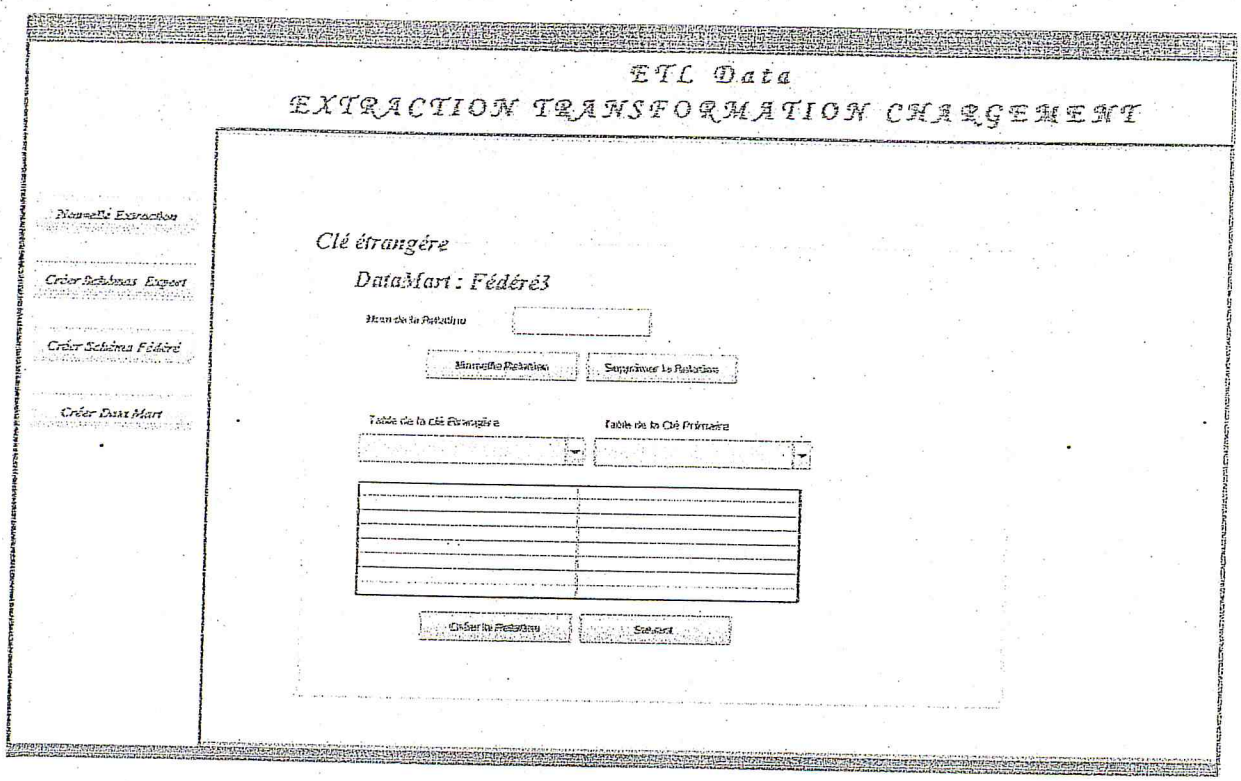


Figure 3.2.19 Créer la Clé Étrangère

Cette fenêtre permet à l'administrateur de créer des relation entre les tables de dimensions (Créer la contrainte de clé étrangère.

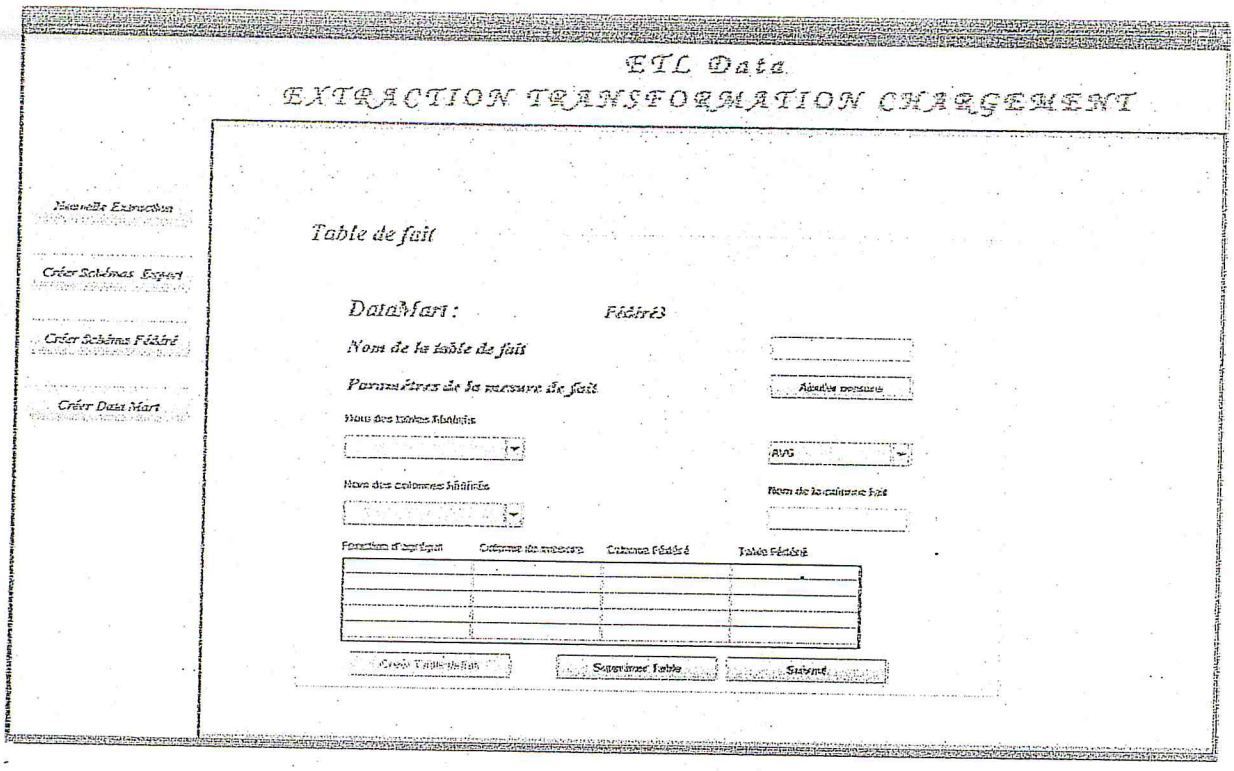


Figure 3.2.20 Table de fait

Cette Fenêtre permet à l'administrateur de créer une table de fait, pour ce fait, il donne un nom à la table de fait .Il choisit les colonnes de mesures à partir des colonnes fédérées. Et la fonction d'agrégat pour chaque colonne mesure, il peut même supprimer la table de fait, si il s'est trompé.

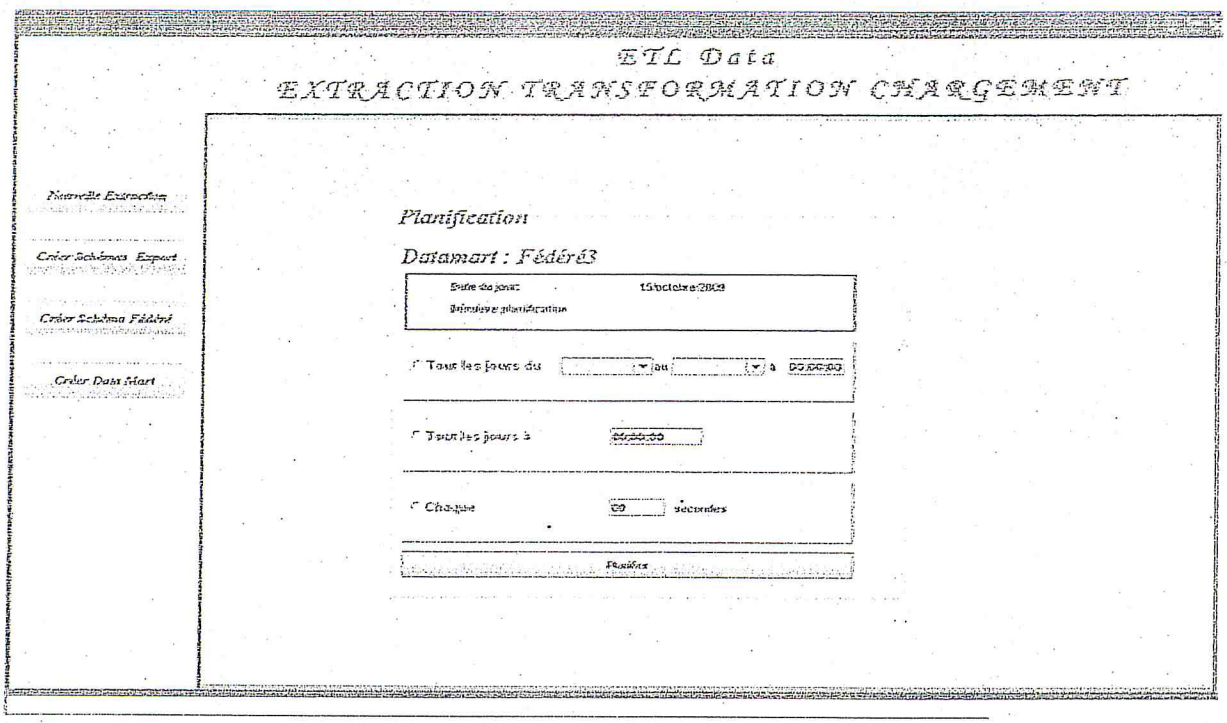


Figure 3.2.21 planifications de l'alimentation du datamart

Cette fenêtre permet à l'administrateur de planifier l'alimentation du datamart

Conclusion

Dans cette étape de notre projet , nous avons présenté les différents outils utilisés pour implémenter notre prototype ainsi les différents interfaces implémentées afin de fournir un système d'intégration de données hétérogènes a base entrepôt de données.

Conclusion générale

Conclusion générale

De nos jours, de larges volumes de données sont disponibles publiquement, les types de données sont divers, et les ressources très nombreuses. Souvent les données provenant de différentes ressources se complètent.

L'architecture d'un système d'intégration d'informations doit permettre leur intégration donnant ainsi l'impression à l'utilisateur qu'il utilise un système homogène, et de lui fournir une vue unifiée de ces données. Dans ce contexte, un problème essentiel de l'intégration est de gérer l'hétérogénéité et la répartition des différentes sources de données. L'intégration de sources de données nécessite des outils qui prennent en charge ces problèmes et qui fournissent un accès efficace à un ensemble de sources de données.

On peut distinguer deux grandes approches pour l'intégration de sources d'information: L'approche *virtuelle et* L'approche *matérialisé*. L'approche *virtuelle* consiste à fondé l'intégration d'informations sur l'exploitation de vues abstraites décrivant le contenu des différentes sources d'information. Les données ne sont pas stockées et ne sont accessibles qu'au niveau des sources d'information L'approche *matérialisé* consiste à voir cette intégration comme la construction de bases de données réelles, regroupant les informations pertinentes pour les applications considérées.

Les systèmes d'intégration utilisent un format commun appelé le *format pivot* pour l'intégration, dans notre contexte, nous avons choisi le modèle relationnel. La raison pour ce choix est basée sur ses nombreux avantages et l'utilisation fréquente de ce modèle dans les systèmes d'informations et il propose un schéma facile à utiliser par rapport aux autres standards, Transact-SQL est le langage de requête utilisé pour interroger les bases de données relationnels est les documents XML , il permet d'exprimer des requête très complexe sur le serveur local ou les serveurs distants.

Dans ce mémoire nous avons présenté une architecture d'intégration de données dans un entrepôt de données, elle repose sur l'approche fédérée fortement couplé, Les niveaux de Sheth sont repris sauf que le premier et deuxième niveau (schéma local et schéma composant) sont combinés dans un seul niveau qui correspond au schéma pivot.

Notre système consiste à définir un schéma fédéré en fonction des schémas sources de données, a partir de se schéma l'administrateur devra cibler les données jugé pertinentes pour construire le schéma externe d'un datamart.

Dans ce travail , nous avons identifié trois points essentielles :

- ✓ L'utilisation des tables systèmes de SGBD SQL Server et PostgreSQL pour construire le schéma pivot, et l'utilisation des schémas XML «XSD» pour définir le schéma des documents XML.
- ✓ La définition d'un schéma fédéré en fonction des différentes sources de données, et la possibilité de modifier le schéma fédéré (ajout d'une source implique un ajout des relations dans le schéma fédéré)
- ✓ Le processus de traitement d'une requête d'alimentation des que le datamart sera planifié par l'administrateur de l'entrepot de données.

Dans ce mémoire nous avons présenté une architecture d'intégration de données dans un entrepôt de données, elle repose sur l'approche fédérée fortement couplé, Les niveaux de Sheth sont repris sauf que le premier et deuxième niveau (schéma local et schéma composant) sont combinés dans un seul niveau qui correspond au schéma pivot.

Notre système consiste à définir un schéma fédéré en fonction des schémas sources de données, a partir de se schéma l'administrateur devra cibler les données jugé pertinentes pour construire le schéma externe d'un datamart.

Dans ce travail , nous avons identifié trois points essentielles :

- ✓ L'utilisation des tables systèmes de SGBD SQL Server et PostgreSQL pour construire le schéma pivot, et l'utilisation des schémas XML «XSD» pour définir le schéma des documents XML.
- ✓ La définition d'un schéma fédéré en fonction des différentes sources de données, et la possibilité de modifier le schéma fédéré (ajout d'une source implique un ajout des relations dans le schéma fédéré)
- ✓ Le processus de traitement d'une requête d'alimentation des que le datamart sera planifié par l'administrateur de l'entrepot de données.

Bibliographies

[BAL 07] M.BALA Interopérabilité Sémantique des Systèmes d'Information Distribués, thèse de magistère,2007

[BUS 97] Busse S., Kutsche R.-D., Federated Information Systems:Concepts, Terminology and Architectures

[CAR 95] M. J. CAREY ET AL., Towards Heterogeneous Multimedia Information Systems:The Garlic Approach March,1995

[DOG 98] DOGAC A et al., An Interoperability Infrastructure for Developing Multidatabase Systems,1998.

[DON 03] Tuyet Dang Dong, Fédérération des données semi-structurées avec XML, thèse de doctorat en informatique , années 2003.

[HES 05] Hes.So, ch.arc-he@Meylan.DBEddy Oracle XML2/05SIG -DEV haute école spécialisées de suisse occidental

[JER 08] JEROME BOUGEAULT , JAVA La maitrise (JAVA 5 et 6) , édition 2008

[JOU 00] Fabrice JOUANOT, Un modèle sémantique pour l'interopérabilité des SI, Article INFORSID 2000 [Catégorie Jeune Chercheur]

[JOU 01] Fabrice JOUANOT, DILEMMA : vers une coopération de systèmes d'informations basée sur la médiation sémantique et la fusion d'objets, Thèse de doctorat, 21 Novembre 2001 .Université de Bourgogne.

[KEZ 07] K.Elaraba Ziane, Données semi-Structurées active , thèse Magistère , Institue Nationnal en Informatique (INI)

[KIM 01] Ralph Kimball , 2001 Entrepôts de données, Guide pratique du concepteur du datawarehouse.

[MAR 06] MARHOUMI Fatima Ezzahra , Entrepôts de données XML : Développement d'un outil Extraction Transformation Load (ETL), thèse du master 2006.

[MIC 00] Programmation d'une base de données Microsoft SQL Server 2000, réalisé par groupe de Microsoft ,

[GRI 07] MATTHIEU GRIMAUD , Comment assurer l'intégration des données structurées dans l'entrepôt de données, L'Institue supérieur du commerce de Paris 2007.

[RIM07] Rim Moussa , Systèmes de Gestion de Bases de Données Réparties & Mécanismes de Répartition avec Oracle (Année Universitaire 2006-2007)

[SAL 08] Nadir Salhi, Benna Amal, Zaia Alimazighi, Bilal Amrouche, Ferhat Makhloufi,"An Ontology and a Description Schema Base for Relational Database Integration",IEEE, pp, 3-10, April 2008.

[SAN01] Sandrine de lignerolles, Piloter l'entreprise grâce au datawarehouse.

[SHE 90] SHETH A., LARSON J. Federated database systems for managing distributed, heterogeneous, and autonomous databases,.3 Sept. 1990

[TES 00]O.Teste, Modélisation et manipulation d'entrepôt de données complexe et historiées, thèse de doctorat .Institue de recherche en informatique de Toulouse 2000.

[UML 05] Benoit CHRROUX , Aomar OSMANI, Yann THIERRY-MIEG , UML2, edition 2005.

Sites Web

[W1] http://grim.developpez.com/articles/concepts/bi- intro/#LV-C-2

[W2] http://ledad.ovh.org/public/datawarehouse.pdf

[W3] http://amkadmi.neuf.fr/liens/ingenierie_archives/sommaire.htm

[W4] http://support.microsoft.com

[W5]http://www.cmi.univ-mrs.fr/~campioni/documents/BD/BD-relationnelles.pdf

[W6] http://ferry.eof.eu.org/support/si017/02_pgAdministration.pdf

[W7] http://grisha.developpez.com/tutoriel/java/scheduling/

[W8] http://msdn.microsoft.com/fr-fr/library/ms188279.aspx