République Algérienne Démocratique et Populaire
Ministère de l' Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Seddik Ben Yahia de Jijel
Faculté des Sciences Exactes et Informatique
Département d'informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option : Système d'Informations et Aide à la Décision

Thème

Génération de requêtes en langage naturel pour le traitement analytique en ligne dans les entrepôts de données

Présenté par :
Rouibah Zineb
Dahouas Selma

Encadré par :
Dr.Boukraa Doulkifli

Promotion : 2019

République Algérienne Démocratique et Populaire
Ministère de l' Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Seddik Ben Yahia de Jijel
Faculté des Sciences Exactes et Informatique
Département d'informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option : Système d'Informations et Aide à la Décision

Thème

Génération de requêtes en langage naturel pour le traitement analytique en ligne dans les entrepôts de données

Présenté par :
Rouibah Zineb
Dahouas Selma

Encadré par :
Dr.Boukraa Doulkifli

Promotion : 2019

Remerciements

Tout d'abord, nous exprimons notre gratitude et nos remerciements à ***Dieu*** *de nous avoir donné la force, la patience et la volonté nécessaires pour mettre fin à ce travail et nous réconcilier tout au long de notre parcours d'étude.*

Notre remerciement s'adressent aussi à Mr " ***Boukraa Doulkifli*** *" pour son soutien, ses orientations, ses précieux conseils et ses encouragements qui nous ont permis d'élaborer ce travail dans des bonnes conditions.*

Nous remercions également les ***membres du jury*** *qui ont accepté de juger notre travail.*

Un grand merci également à nos ***parents*** *qui ont toujours été là pour nous, et qui ont eu la patience de nous enseigner.*

Enfin, nous remercions nos ***amis****, qui nous ont toujours soutenus et pour leurs encouragements qui ont été d'une grande aide.*

Merci pour tous ceux qui nous ont aidés de près ou de loin.

Dédicaces

Je dédie ce travail à :

Premièrement et avant tout à mes chers parents qui m'ont comblée de leur soutien et m'ont vouée un amour et confiance inconditionnels, et qui m'ont fourni tous les moyens de confort dans mes études pour arriver à ce point.

Je dédie aussi ce travail à :

Mes sœurs, mon frère, et les membres de ma famille, ils sont également chers, merci d'être toujours présents pour moi. Vos consolations m'étaient très bénéfiques pour continuer brillamment le chemin que j'ai tracé.

A mon encadreur Boukraa Doulkifli qui nous a dirigées de tous ses efforts pour réaliser ce travail, les conseils, les recommandations, les suggestions qu'il m'a donnés.

Tous mes amies pour tous les meilleurs et les plus agréables moments qu'on a partagés, pour toute l'entente qui nous unit. Elles ont été toujours près de moi dans le mal et le bien, je les souhaite plus de succès, mes chères " Imane, Safa, Nadra, Wafa, Dalal et Houda". Et enfin à mon binôme "Zineb" pour sa compréhension, et pour son soutien moral.

Et enfin à tous ceux qui ont contribué de près ou de loin, à la réalisation de ce travail.

Selma

Dédicaces

Je dédie mon travail à

Ceux qui m'ont donné le tout alors qu'ils n'ont rien, qu'ils m'ont soutenue dans mon chagrin, mes parents le symbole de tendresse je vous aime tant, c'est grâce à vous que je suis ici, tous mes gratitudes ne pourraient jamais rendre vos plaisirs.

A mes frères surtout mon frère " Moussa" qui était prêt de me donner tout courage et soutien dont j'ai besoin, vraiment il m'a aidé tout au long de mes études universitaires, Que Dieu te bénisse. Et à toute ma famille je vous souhaite le bonheur.

A mon encadreur "Boukraa Doulkifli" Qui nous a bien aidé à réaliser ce travail, et même ses efforts et conseils pour arriver à ce point-là.

A mes amies pour les bons moments, ils ont été ma deuxième famille surtout mon groupe "Noussaiba, Wafa, Houda, Dalal", à mon binôme "Selma" pour sa compréhension, et pour son soutien moral.

Et enfin à tous ceux qui ont contribué de près ou de loin, à la réalisation de ce travail.

Zineb

Table des matières

Liste des tableaux

Table des figures

Liste des abréviations

LN Langage Naturel

ED Entrepôt de Données

ETC Extraction Transformation Chargement

BDD Base de Données

IA Intelligence Artificielle

OLTP On-Line Transaction Processing

OLAP On-line Analytical Processing

SGBD Système de Gestion de Base De Données

SGBDR Système de Gestion de Base De Données Relationnelles

ROLAP Relational On-Line Analytical Processing

MOLAP Multidimensional On-Line Analytical Processing

HOLAP Hybrid Online Analytical Processing

3FN Troisième Forme Normale

BI Business Intelligence

SQL Structured Query Language

XML Extensible Markup Language

XSD XML Schema Definition

DTD Document Type Definition

W3C World Wide Web Consortium

Introduction générale

Depuis que l'homme a inventé l'ordinateur, le domaine de l'informatique s'est développé de jour en jour, et avec ce développement, l'homme a toujours été tenté de rendre le travail avec la machine plus simple. Or, comme la machine possède ses propres langages à différents niveaux, l'interaction homme-machine a toujours été au centre d'intérêt de plusieurs domaines à la fois. De nos jours, avec les progrès de l'intelligence artificielle (IA), la machine est de plus en plus rapprochée de l'homme, par sa capacité d'exécuter des tâches humaines. Le traitement de la langue naturelle s'inscrit comme l'une des branches de l'IA les plus prometteuse, notamment pour ce qui relève du traitement de la langue naturelle pour des besoins de traduction, d'assistance en ligne, etc.

Par ailleurs, depuis l'introduction du modèle relationnel dans le monde des entreprises, ce modèle accompagné de son langage d'accès SQL a connu un succès incontestable, comme en témoigne la grande part du marché des SGBD relationnels. Ce succès est du entre autres, à la simplicité du modèle relationnel et aussi du fait que SQL est très proche du langage naturel (langue anglaise) par des clauses de base exprimée naturellement, en l'occurrence SELECT, FROM et WHERE.

Toutefois, le SQL a hérité de l'inconvénient du modèle relationnel dans le sens du «tout relation», qui se traduit par la représentation de la réalité par des relations seulement, et le besoin de *joindre* beaucoup de relations afin d'aboutir aux données voulues.

D'ailleurs, la complexité d'une requête SQL commence dès lors qu'il faut inclure des jointures dans la clause WHERE et dont le nombre est fonction du nombre de tables à joindre.

De ce fait, les utilisateurs finaux n'expriment pas de requêtes SQL directement dans les applications, mais plutôt à travers des interfaces de saisie ou de récupération de données, tout en exécutant le SQL en arrière plan.

Toutefois, de nombreuses situations nécessitent l'accès à une base de données sans passer par une application et par des utilisateurs finaux. Par exemple, dans un contexte de mobilité, un utilisateur peut vouloir accéder à des données sur son cellulaire ou tablette à travers l'expression rapide de sa requête. Ainsi, avec les possibilités de reconnaissance vocale, la requête de l'utilisateur sera traduite en texte qui sera exécuté moyennant une connexion au serveur de base de données de son

entreprise. Dans ce cas, une interface en langage naturel pour exprimer ces requêtes trouve tout son intérêt.

Dans notre travail, nous nous intéressons à un accès particulier aux bases de données qui est l'analyse OLAP (On-Line Analytical Processing). L'accès se fait à des structures analytiques appelées cubes OLAP ou en général à un entrepôt de données, par des analystes. La particularité de ces bases de données est qu'elles sont modélisées d'une manière spécifique sous la forme de faits et de dimensions et que les requêtes sont prévisibles, sous forme de regroupements de données selon les niveaux hiérarchiques des dimensions. L'intérêt d'un accès en langage naturel à des structures OLAP est d'autant plus intéressant qu'il s'agit d'un contexte de prise de décision, laquelle doit être faite dans certains cas le plus vite possible.

L'objectif de ce travail est de proposer des solutions pour faciliter l'accès à des cubes de données OLAP, en remplaçant les requêtes SQL analytiques par des requêtes en langage naturel. Pour aboutir à cet objectif, nous contribuons par ce qui suit :

- Nous proposons une modélisation des méta-données d'un entrepôt de données ou cubes OLAP adaptée pour la génération de requêtes décisionnelles. Ce modèle est générique et permet après instanciation d'obtenir les méta-données d'un entrepôt de données ou cubes spécifiques. Nous choisirons le langage XML pour la représentation de ce modèle.
- Nous proposons un algorithme de génération de requêtes en langage naturel et également en SQL pour permettre leur exécution.
- Nous contribuons par une application informatique dédiée à des développeurs de bases de données ou des analystes qui leur permet de créer leurs schémas de cubes, de générer les requêtes et de visualiser puis exécuter les requêtes. La partie de visualisation et d'exécution peut être déployée séparément, notamment sur des périphériques mobiles (Smartphone ou tablettes) pour un accès mobile.

Nous avons organisé notre mémoire comme suit :

Dans le premier chapitre, nous définissons les concepts de base, où dans une première section, nous présenterons les notions d'analyse ED et OLAP. Nous parlons ensuite de l'interaction en LN avec les bases de données et en particulier avec l'ED dans les deux sens « utilisateur vers ED » et « ED vers utilisateur ».

Dans le deuxième chapitre, nous présentons le modèle générique des métadonnées d'un entrepôt de données sous le format XML.

Dans le troisième chapitre, nous développons notre algorithme de génération des requêtes en langage naturel d'un ED et de génération de requêtes SQL correspondantes.

Enfin, dans le quatrième chapitre, nous présentons la phase de mis en œuvre de notre application en présentant d'abord les outils utilisés pour la réaliser. Par la suite, nous détaillons les structures de bases de données de l'application et l'interface graphique principale pour spécifier l'ED ou un cube OLAP. Enfin, nous montrons la génération des requêtes en langage naturel et leur requêtes SQL équivalentes ainsi que leur exécution.

Enfin, nous concluons et donnons quelques perspectives.

CHAPITRE 1

Concepts de base

1.1 Introduction

Le concept d'entrepôt de données a été formalisé pour la première fois en 1990 par Bill Inmon. Il est exclusivement destiné aux processus d'aide à la décision.[1]

Les premiers utilisateurs des entrepôts de données sont les entreprises de toutes dimensions, où les systèmes d'entreposage de données permettent aux chefs d'entreprise de stocker les informations et les données de l'entreprise à partir des systèmes opérationnels et d'un large éventail d'autres ressources de données. En plus la modélisation dimensionnelle qui représente l'entrepôt de données est simple, acceptable et compréhensible par les utilisateurs.

L'entrepôt de données est capable de rendre les données de l'entreprise facilement accessibles par les utilisateurs, présenter les données d'une manière cohérente bien qu'elles soient assemblées à partir de différentes sources, ces données sont adaptables et résistantes au changement ce qui améliore la prise de décision.

Le processus ETC (Extraction-Transformation-Chargement) est chargé d'extraire les données collectées à partir de différentes sources hétérogènes et de les transformer par nettoyage et normalisation, puis de charger ces données dans l'entrepôt.

Les systèmes d'entrepôt de données servent les utilisateurs ou les travailleurs du savoir dans le rôle d'analyse des données et de prise de décision. De tels systèmes peuvent organiser et présenter des données sous différents formats afin de répondre aux divers besoins des utilisateurs. Ces systèmes sont appelés systèmes de traitement analytique en ligne (OLAP).[2]
Par ailleurs, avec les développements dans le temps actuel, le rapprochement de la machine à l'être

humain est devenu un besoin, et pour cela les spécialistes tentent à chaque fois de développer des outils dans ce contexte, comme un exemple l'interaction en langage naturel avec les bases de données dans les deux sens : du langage naturel (LN) vers les bases de données (BDD) et des BDD vers LN.

Dans le cas de la prise de décision et pour des interactions plus ciblées avec l'utilisateur, un entrepôt de données s'avère être un moyen efficace. L'interaction en langage naturel entre l'entrepôt et les utilisateurs via des requêtes OLAP aide les responsables des entreprises à accéder aux informations plus rapidement et plus facilement. En outre, dans certains contextes, l'accès via des dispositifs mobiles tels que les cellulaires ou tablettes à de petites quantités de données analytiques devient plus qu'une nécessité. Un tel accès étant aujourd'hui démocratisé par les possibilités de reconnaissance du langage naturel par les appareils mobiles.

Dans ce chapitre, nous allons présenter une vue générale sur les entrepôts de données et leurs points principaux, puis nous aborderons les interactions en langage naturel avec BDD, et avec les entrepôts de données en particulier. Nous terminons ce chapitre par l'énoncé des objectifs de notre travail.

1.2 Entrepôts de données et OLAP

1.2.1 Définition d'entrepôt de données

Selon W.H.Inmon, architecte de premier plan dans la construction des systèmes d'entrepôt de données, un entrepôt de données est une collection de données orientée sujets, intégrées, variant dans le temps et non volatiles, à l'appui du processus de prise de décision de la direction. On peut donc dire que l'entrepôt de données est un ensemble de données sémantiquement cohérentes qui servent d'implémentation physique d'un modèle de données d'aide à la décision et stocke les informations dont une entreprise a besoin pour prendre des décisions stratégiques. Ainsi, on peut dire que l'architecture d'un entrepôt de données est construite en intégrant des données provenant de multiples sources hétérogènes pour prendre en charge l'exécution de requêtes ad hoc, l'édition des rapports analytiques et la prise de décision.[2]

1.2.2 Caractéristiques d'un entrepôt de données

Dans cette section, nous allons détailler les quatre caractéristiques d'un entrepôt de données, à savoir : *orienté sujet, intégrés, non volatiles et historisés.*

Les entrepôts de données doivent être **orientés sujet**, ce qui signifie qu'il doit être possible de les définir par leur sujet. Par exemple, un entrepôt peut être déployé spécialement pour analyser les données liées aux ventes de l'entreprise. Cet entrepôt de données servira à répondre à des questions

comme « quels ont été les meilleurs clients pour tel produit au cours de l'année précédente ? ». [3] Dans la même logique, l'entrepôt de données doit être en mesure d'assembler des données en provenance de différentes sources dans un format consistant. Il doit permettre de résoudre les problèmes comme les conflits de noms et les incohérences en termes d'unités de mesure. On parle là **d'intégration**.[3] Troisièmement, les entrepôts de données doivent être **non-volatiles**. Cela signifie qu'une fois qu'une donnée est entrée dans l'entrepôt, elle ne doit plus changer. L'utilisateur est ainsi en mesure d'analyser les données telles qu'elles ont été stockées dans l'entrepôt.[3]
Dernière caractéristique des entrepôts de données, ils doivent être **historisés**. Cela signifie qu'elles permettent de focaliser les analyses sur les changements survenus au fil du temps à partir de larges ensembles de données, afin de découvrir des tendances. C'est ce qui oppose les entrepôts de données aux systèmes OLTP dont les données opérationnelles sont atomiques et ne reflètent que la valeur actuelle de la dernière transaction.[3]

La figure suivante illustre ces caractéristiques :

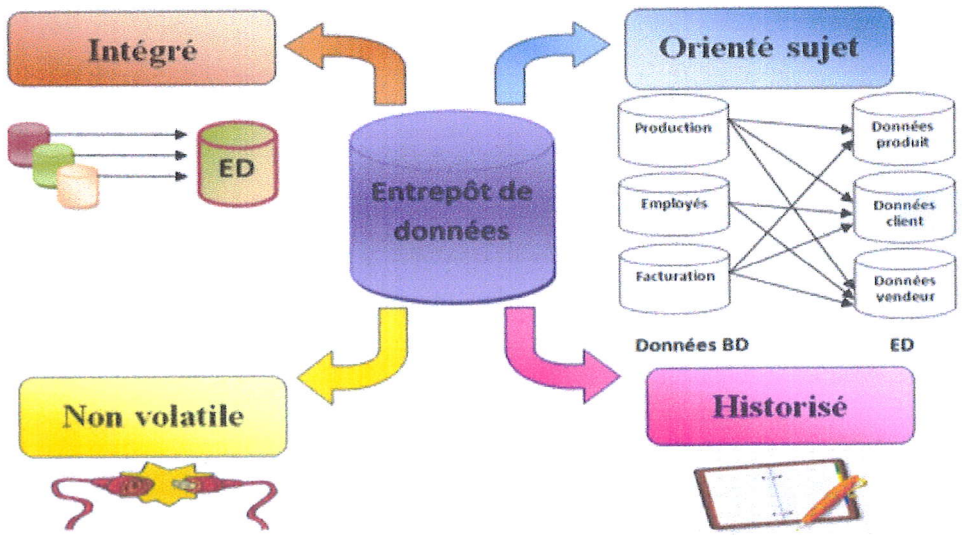

FIGURE 1.1 – Caractéristiques d'un entrepôt de données.

1.2.3 Architecture d'entrepôt de données

Une architecture simple a été proposée par Inmon et illustre les différents composants et étapes pour construire un bon entrepôt de données capable de répondre aux besoins des utilisateurs et permettant une interaction avec ses différents types.
La figure suivante montre ces étapes.[4]

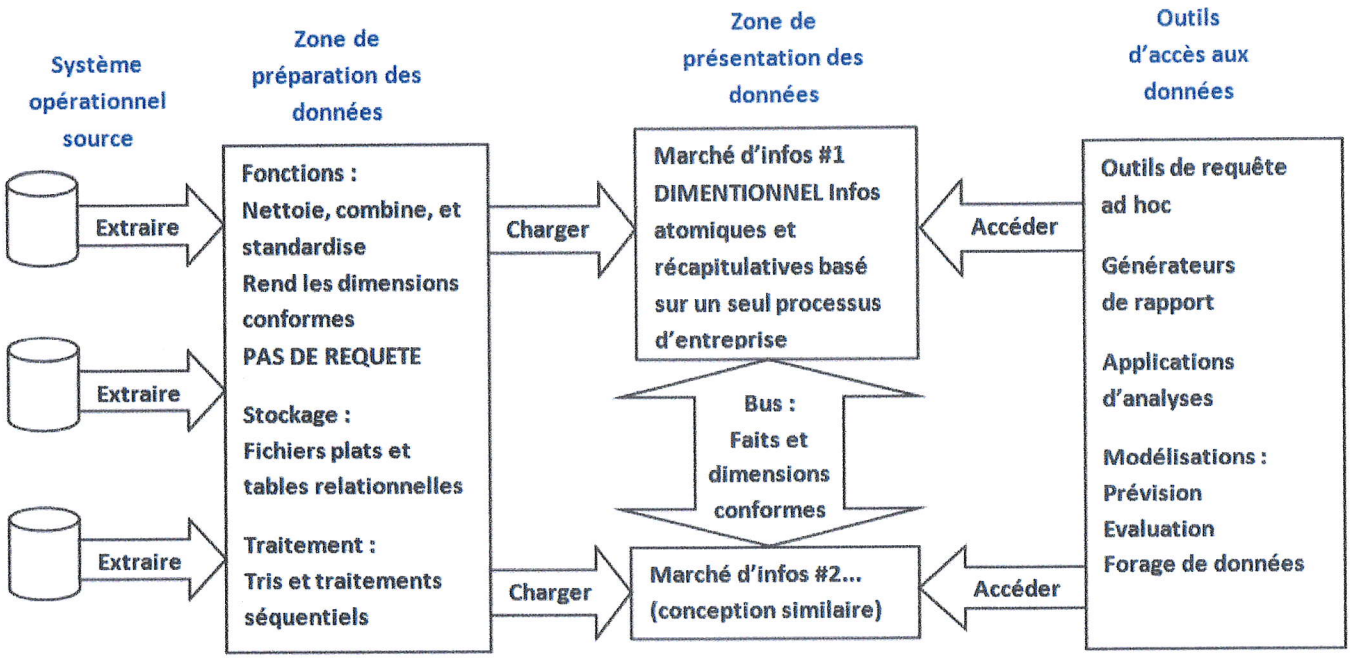

FIGURE 1.2 – Architecture d'entrepôt de données.[4]

1.2.3.1 Préparation des données

La préparation de l'entrepôt de données englobe tout ce qui est entre les applications opérationnelles sources et la présentation des données. Un bon exemple pour mieux comprendre cette partie est l'exemple de la réalité qui compare la zone de préparation des données à la cuisine, où des produits nutritifs crus sont transformés en un repas. Dans l'entrepôt de données, des données opérationnelles crues sont transformées en éléments accessibles aux requêtes des utilisateurs qui les consommeront selon leurs besoins. La zone de préparation des données est en arrière-plan et n'est pas accessible aux utilisateurs finaux, elle est accessible uniquement aux professionnels, comme la cuisine d'un restaurant qui est accessible uniquement au personnel qui est occupé à la préparation et ne peut pas répondre aux demandes d'information des clients, et c'est le cas pour l'entrepôt de données. Donc un point très important dans l'aménagement de l'entrepôt de données, est d'interdire aux utilisateurs l'accès à la zone de préparation des données, qui ne fournit aucun service de requête ou de présentation.[4]

La zone de préparation de l'entrepôt de données est considérée à la fois comme une zone de stockage des différentes données et un ensemble de tâches couramment appelé ETC.

➢ **Processus ETC**

Le processus ETC est un ensemble de tâches divisé en 3 catégories qui sont :

- ✓ Extraction (E), première étape du processus d'apport de données à l'entrepôt de données. Elle permet de se connecter à diverses sources de données, de les lire, de les interpréter et de les copier dans la zone de préparation des données en vue de manipulation ultérieure.
- ✓ Transformation (T), après l'extraction, les données sont transformées de plusieurs manières selon le cas, par exemple :
  - nettoyer :
    - correction orthographique
    - résolution de conflits de domaine
    - traitement du problème des éléments manquants
    - conversion à des formats standards
  - combiner des données provenant de sources multiples.
  - dédoubler les données s'il y a lieu et pourvues de clés propres à l'entrepôt de données.
- ✓ Chargement (C), après nettoyage et ajustements des données, ils seront prêts à être chargés dans la zone de présentation de l'entrepôt de données.

Il est acceptable de créer une base de données normalisée pour supporter les processus de préparation ; toutefois, ce n'est pas l'objectif principal. Les structures normalisées doivent être écartées au niveau des requêtes des utilisateurs car elles sont difficiles à comprendre et entraînent des performances médiocres. Dès lors qu'une base de données supporte des services de requête et de présentation, elle doit être considérée comme faisant partie de la zone de présentation de l'entrepôt de données. Par défaut, les bases de données normalisées sont exclues de la zone de présentation rigoureusement être structurée selon le modèle dimensionnel.[4]

1.2.3.2 Présentation des données

Contrairement à la zone de préparation qui interdit l'accès des utilisateurs, la zone de présentation est le lieu où les données sont offertes aux requêtes des utilisateurs, aux programmes de reporting et autres applications d'analyse. Les données sont organisées et stockées dans cette zone qui est l'entrepôt de données, tel qu'il est perçu par la communauté des utilisateurs. Elle est tout ce que les utilisateurs voient et touchent par le biais des outils d'accès.[4]
Les données doivent être présentées, stockées et consultées sous forme de schémas dimensionnels, car la modélisation dimensionnelle est la seule technique viable pour fournir des données à des utilisateurs d'entrepôt de données. Cette technique permettant de rendre les bases de données simples et compréhensibles. À partir des années 1970, les services informatiques, les consultants, les utilisateurs finaux et les fournisseurs se sont ralliés à une structure dimensionnelle adaptée aux besoins de simplicité fondamentale des humains. Au contraire à la modélisation en 3NF (en troisième forme normale) qui vise à supprimer les redondances dans les données qui sont divisées en entités discrètes, chacune devenant une table de la base de données relationnelle. Ce modèle

normalisé est trop compliqué pour les requêtes d'entrepôt de données. Les utilisateurs ne peuvent pas comprendre, parcourir, ni se rappeler des modèles normalisés dont la complexité s'apparente à l'enchevêtrement des autoroutes d'un centre urbain congestionné. De même, les systèmes de gestion de bases de données relationnelles (SGBDR) ne peuvent pas faire de requêtes efficaces sur un modèle normalisé ; sa complexité dépasse les possibilités de ses fonctions d'optimisation, ce qui conduit à des performances désastreuses. L'utilisation de la modélisation normalisée dans un entrepôt de données va à l'encontre du but recherché, qui est de permettre de trouver rapidement des données par une recherche intuitive.[4]

1.2.3.3 Outils d'accès aux données

Le dernier composant majeur d'un environnement d'entrepôts de données est l'ensemble des outils d'accès aux données. Nous appliquons ce terme général à un ensemble de moyens fournis aux utilisateurs pour exploiter la zone de présentation en vue de prendre des décisions basées sur des analyses. Par définition, tous les outils d'accès aux données font des requêtes sur les données de la zone de présentation. Les requêtes sont évidemment la raison d'être de l'entrepôt de données.[4]

Un outil d'accès aux données peut être une chose aussi simple qu'un outil de requête ad hoc ou aussi complexe qu'une application de forage de données ou de modélisation. Les outils de requêtes ad hoc, quelle qu'en soit la puissance, ne peuvent être compris et utilisés efficacement que par une petite fraction de la population des utilisateurs potentiels d'entrepôts de données. La plupart des utilisateurs accède aux données par l'intermédiaire d'applications d'analyse préfabriquées, pilotées par des paramètres. Environ 80 à 90 des utilisateurs potentiels sont desservis par ces applications qui sont essentiellement des modèles préétablis leur évitant d'avoir à construire eux-mêmes des requêtes relationnelles. Certains outils d'accès plus sophistiqués, les outils de modélisation ou de prévision, sont en mesure de renvoyer leurs résultats vers les applications opérationnelles ou vers les zones de préparation ou de présentation de l'entrepôt de données.[4]

Comme un exemple de ces outils on a la technologie OLAP (traitement analytique en ligne) qui est basée sur la notion des cubes. Les données alors sont stockées dans des cubes. Cette technologie ne s'appelait pas OLAP à l'origine, mais beaucoup de fournisseurs des premiers systèmes d'aide à la décision les ont construits autour du concept de cube, de sorte qu'actuellement les fournisseurs de systèmes OLAP se sont naturellement alignés sur l'approche dimensionnelle des entrepôts de données. Dans ce qui suit, nous allons présenter les détails sur l'OLAP.[4]

1.2.4 Vocabulaire de la modélisation dimensionnelle

Selon le livre "Entrepôt de données : Guide pratique de modélisation dimensionnelle" de Kimball et Ross, les termes dimension et fait sont apparus dans un projet de recherche commun General Mills et de l'Université de Dartmouth au cours des années 1960. Dans les années 1970, aussi bien

AC Nielsen que IRI ont utilisé ces termes systématiquement pour décrire leurs offres d'information, décrites exactement aujourd'hui comme étant des marchés d'infos dimensionnelles destinés à des données de vente. Bien avant que la simplicité ne devienne une tendance générale, ces précurseurs dans la fourniture de bases de données se sont appuyés sur ces concepts pour simplifier la présentation des informations d'analyse. Ils comprenaient qu'une base de données ne serait utilisée qu'à condition d'être présentée de manière simple.

L'approche dimensionnelle n'est vraisemblablement pas l'invention d'une seule personne. Il s'agit d'une force irrésistible dans la conception de bases de données, qui s'impose dès lors que le concepteur fait de l'intelligibilité et de la performance ses objectifs prioritaires. [4]

1.2.4.1 Table de faits

La table de fait dans un modèle dimensionnel est la table principale qui contient comme clé la concaténation des clés des tables de dimensions que nous allons présenter par la suite, ainsi que les mesures qui représentent des indicateurs pour mesurer la performance, comme le montre la figure ci-dessous :

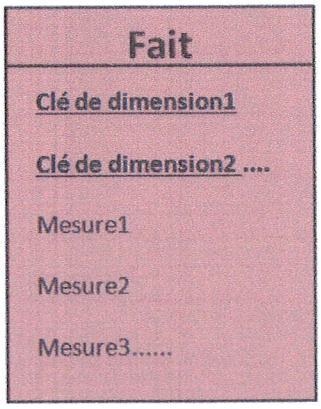

FIGURE 1.3 – Table de fait.

Nous utilisons le terme "fait " pour représenter une mesure économique. Nous pouvons nous imaginer installés sur le marché, observant des produits en train d'être vendus et notant la quantité vendue et le montant de la vente chaque jour pour chaque produit dans chaque magasin. Une mesure est prise à l'intersection de toutes les dimensions (jour, produit et magasin). La liste des dimensions définit le grain de la table et nous dit quelle est la portée de la mesure.
Les faits les plus utiles sont des faits numériques, additifs, tels que les montants des ventes. L'additivité est cruciale parce que les applications d'entrepôt de données ne récupèrent presque jamais une seule ligne de table de fait, elles récupèrent des centaines et des milliers à la fois. C'est pour ça il est très utile de les additionner. Il existe trois types d'additivité : additif qui indique que les

valeurs peuvent être additionnées selon toutes les dimensions à différents niveaux de granularité, semi-additif qui indique que les valeurs ne peuvent être additionnées que pour certaines dimensions. Enfin les faits non-additifs ne peuvent pas du tout être additionnés par rapport à aucune dimension.

1.2.4.2 Tables de dimensions

Les tables de dimensions sont les compagnes obligatoires d'une table de faits. Les tables de dimensions contiennent des descriptions textuelles de l'activité. Dans un modèle dimensionnel bien conçu, les tables de dimensions ont de nombreuses colonnes ou attributs. Ces attributs décrivent les lignes de la table de dimension. Chaque dimension est définie par son unique clé primaire, cette clé primaire sert de base à l'intégrité référentielle de toute table de faits à laquelle elle est jointe.

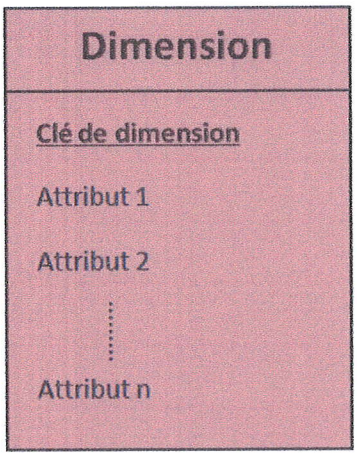

FIGURE 1.4 – La table de dimension.

Les attributs de dimension sont la principale source des contraintes de requête, de groupement et d'intitulés de colonne des états. Dans une requête servant à la préparation d'un état ou à une recherche, les attributs sont identifiés par le mot par (en anglais by). Par exemple, quand un utilisateur indique qu'il veut voir le montant des ventes en euros par marque et semaine, la semaine et la marque doivent être disponible en tant qu'attributs de dimensions.

Les tables de dimension sont les points d'entrée dans la table de faits. Des attributs de dimension nombreux permettent de varier les possibilités d'analyse en tranches et en dés. Les dimensions établissent l'interface homme/entrepôt de données.

1.2.4.3 Relier les faits et les dimensions

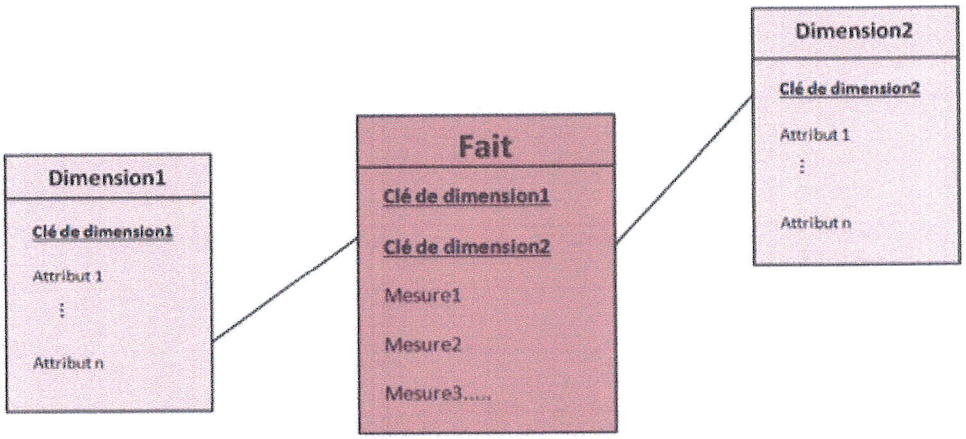

FIGURE 1.5 – Table de fait et tables de dimension dans un modèle dimensionnel.

La table de faits contenant des mesures est jointe à un ensemble de tables de dimensions remplies d'attributs descriptifs. Cette structure est appelée "schéma en étoile". Ce schéma est caractérisé par la simplicité et la symétrie afin que les utilisateurs puissent plus facilement comprendre et parcourir les données.
Une dimension peut être hiérarchisée à plusieurs niveaux. On obtient donc une hiérarchie qui est le détail de la dimension et on le fait en normalisant les tables de dimension dans 3FN. Cette nouvelle structure s'appelle "flocon de neige".

1.2.5 Technologie OLAP (Online Analytical Processing)

Le traitement analytique en ligne (OLAP) est conçu comme une technologie de système d'information permettant d'accéder, de visualiser et d'analyser efficacement les données stockées dans l'entrepôt de données. Il agit comme un pont entre l'entrepôt de données et le module d'exploration de règles d'association.[5]

1.2.5.1 Cube OLAP

Un cube OLAP est une base de données à n dimensions, où n est le nombre de dimensions. Les mesures du cube OLAP proviennent de la table de faits, tandis que les dimensions OLAP proviennent des tables de dimension. Les tables de dimension peuvent contenir des données hiérarchiques, les utilisateurs peuvent afficher les données en explorant le cube OLAP en fonction de leurs besoins.[5]

Exemple : La figure suivante représente un cube de données formé de montants de vente en cellules et de trois arêtes graduées respectivement par des catégories de produits, des villes de magasins et

des trimestres, on peut lire la cellules zoomée comme suite : le montant des ventes des produits de catégorie (HIFI) dans la ville (Toulouse) dans le trimestre (T2.07) est 1500 euro.[6]

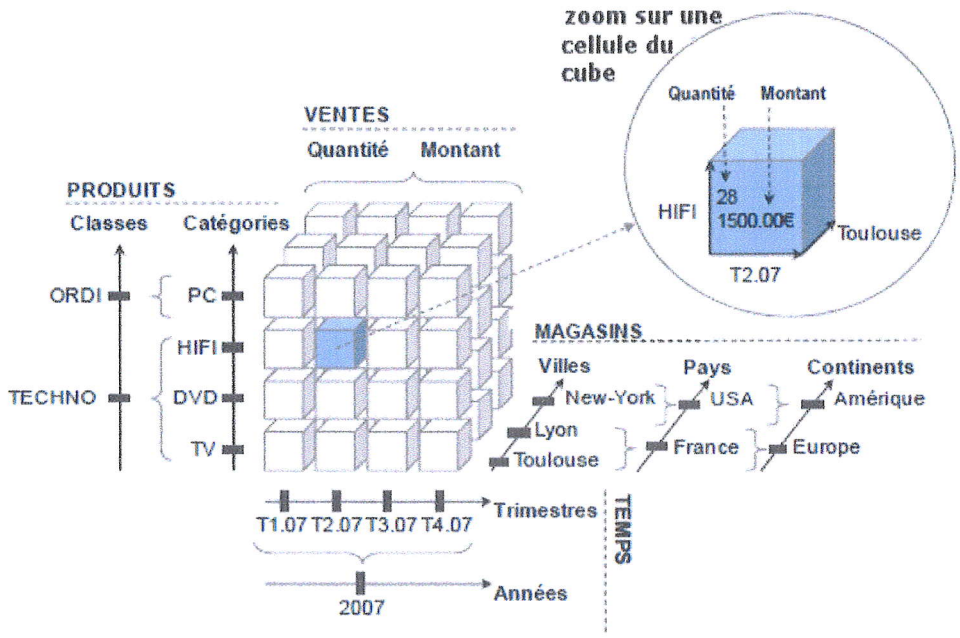

FIGURE 1.6 – Notion de cube.

1.2.5.2 Pourquoi OLAP dans l'entrepôt de données ?

L'entrepôt de données et le processus OLAP sont complémentaire, le premier représente une vue statique de données qui collecte et combine des données provenant de sources multiples, alors que le processus OLAP est venu exploiter ces données pour expliquer le présent et prédire le futur, à travers leurs requêtes qui touchent de grandes quantités de données, Ces requêtes dépendent de différents types des opérateurs OLAP (restriction, transformation, ordonnancement ...).[2]
OLAP transforme les «données» de l'entrepôt de données en «informations stratégiques». Cela va de la navigation de base (souvent appelée slice and dice) aux calculs, en passant par des analyses plus sérieuses telles que la modélisation complexe.[2]

1.2.5.3 Les opérateurs OLAP

Les opérateurs OLAP permettent la navigation dans le cube OLAP. On peut distinguer deux familles d'opérateurs :

- Les opérateurs liés à la structure : ces opérateurs concernent la représentation, et permettent un changement de points de vue selon différentes dimensions.
- Les opérateurs liés aux données : ces opérateurs agissant sur la granularité des données et l'extraction des données.

➢ **Les opérateurs liés à la structure**

Rotate : effectue sur le cube une rotation autour d'un de ses trois axes passant par le centre de deux faces opposées, de façon à présenter un ensemble de faces différent, cette opération consiste à changer l'axe d'analyse en cours d'utilisation.

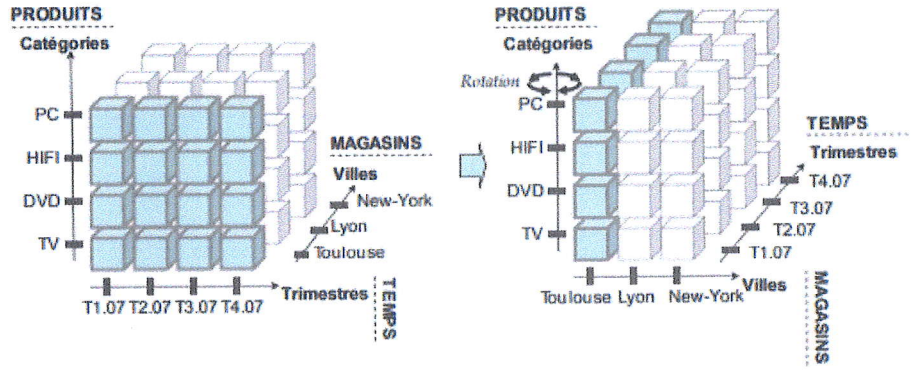

FIGURE 1.7 – Principe de rotation.[6]

Switch(Permutation) : consiste à inter-changer la position des membres d'une dimension.

(Un)nest : permet d'imbriquer des membres à partir du cube. L'intérêt de cet opérateur permet de grouper sur une représentation bidimensionnelle toutes les mesures d'un cube quel que soit le nombre de ses dimensions.

Push : consiste à combiner les membres d'une dimension aux mesures du cube, passer des mesure comme contenu de cellule.

➢ **Les opérateurs liés aux données**

Les opérations de forage (Roll-up, Drill-down) placent la navigation sur la structure hiérarchique des axes d'analyses, afin de permettre l'analyse d'un indicateur avec plus ou moins de précision. [6]

Roll-up : consiste à représenter les données du cube à un niveau de granularité supérieur conformément à la hiérarchie définie sur la dimension.

Drill-down : consiste à représenter les données du cube à un niveau de granularité de niveau inférieur conformément à la hiérarchie définie sur la dimension.

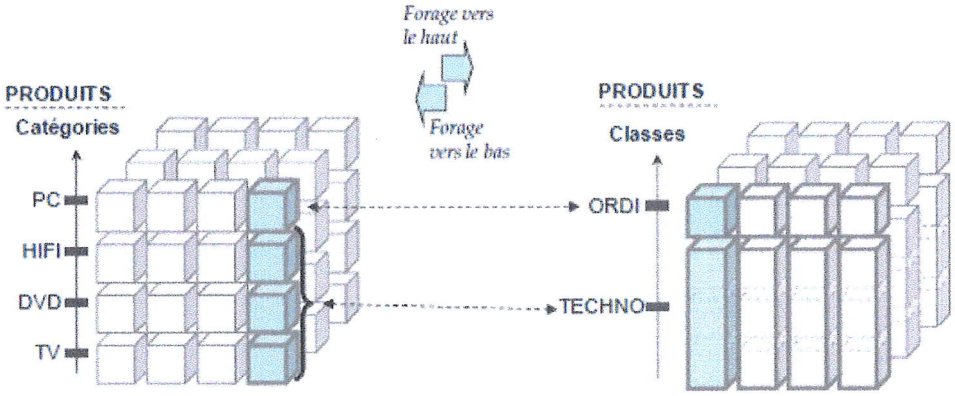

FIGURE 1.8 – Principe du forage.[6]

Les opérations de restriction permettent à un utilisateur de restreindre l'ensemble des données analysées. La spécification d'une tranche de cube (slice) consiste à exprimer une restriction sur une des données de l'un des axes d'analyse. La spécification d'un sous-cube (dice) consiste à exprimer une restriction sur les données d'un indicateur d'analyse.[6]

Slice : correspond à une projection selon une dimension du cube, qui permet de travailler sur une tranche du cube.

Dice : correspond à une sélection du cube, qui permet d'extraire un bloc de données (Sous-cube).

1.2.5.4 Types de systèmes OLAP

On peut découvrir trois différents types d'une implémentation OLAP, nous allons décrire brièvement chaque type.

OLAP multidimensionnelle (MOLAP) : est la version données en grille décrite par E. Codd ; on stocke les données dans un cube qui est en fait une base de données multidimensionnelle. De cette façon, le concept de relationnel disparaît.
Le produit MOLAP doit utiliser uniquement la multiplication et l'addition, et les ordinateurs fonctionnent très rapidement puisque l'architecture MOLAP réalise des pré-agrégations sur tous les niveaux de hiérarchies des dimensions de l'ED. La technologie MOLAP est la meilleure opération pour les tableaux denses, où la plupart des cellules de données d'un cube contiennent une valeur.[7]
OLAP relationnel (ROLAP) : utilise un schéma étoile / flocon de neige traditionnel et des sources de données relationnelles uniquement. Avec ROLAP, les données ne sont ni agrégées ni manipulées. Les données sont stockées dans des tables relationnelles pouvant être interrogées par SQL.[7]

OLAP Hybride (HOLAP) : L'HOLAP est un mélange du ROLAP et du MOLAP, il combine entre les bons côtés des deux systèmes, HOLAP est donc Hybride.
Avec OLAP hybride, les données sont stockées à la fois dans un magasin de données OLAP et dans une base de données relationnelle. Par exemple, vous avez des données de niveau résumé stockées dans le magasin de données OLAP et des données détaillées stockées dans la base de données relationnelle. Vous pouvez ensuite explorer en aval le magasin de données OLAP jusqu'à l'architecture hybride.[7]

1.2.5.5 Les outils OLAP

Dans une vision générale, les outils OLAP sont un moyen efficace pour incarner les bases du processus OLAP. Ils remplacent bien l'utilisateur humain en guidant l'analyse de différents cubes d'OLAP afin d'améliorer les performances de l'entreprise ou de rechercher les raisons de certains problèmes, puis de solutions, ils sont généralement utilisés pour l'analyse historique. Dans la partie ci-dessous, on va présenter certains de ces outils publiés sur le site Software Advice, ils ont été mis à jour le 14 mars 2019.[8]
Dundas BI : de **Dundas Data Visualization** est une plate-forme de veille stratégique et de visualisation de données basée sur un navigateur, qui comprend des tableaux de bord intégrés, des outils de création de rapports et des analyses de données. Il offre aux utilisateurs finaux la possibilité de créer des tableaux de bord interactifs personnalisables, de créer leurs propres rapports, d'exécuter des requêtes ad-hoc, ainsi que d'analyser et d'analyser en profondeur leurs données et leurs mesures de performance. Elle supporte les plateformes : Mac, Win, Linux
Sisense Software : est une solution de Business Intelligence (BI) agile qui fournit des outils avancés pour gérer les données d'entreprise avec des analyses, des éléments visuels et des rapports. La solution permet aux entreprises d'analyser des jeux de données volumineux et disparates et de générer des tendances commerciales pertinentes pour eux. Elle supporte les plateformes : Mac, Win, Linux.
Oracle Business Intelligence Standard Edition One : est une suite de Business Intelligence sur site conçue pour les petites et moyennes entreprises. Les fonctionnalités principales incluent la modélisation de données, ETC, l'analyse ad-hoc, la base de données, les tableaux de bord et les rapports. Oracle BI Standard Edition One prend également en charge les plates-formes : Mac, Win, Linux.

1.3 L'interaction en langage naturel avec les bases de données

Les gens ne comprennent pas le langage de la machine qui est complexe pour eux. Ainsi, pour pouvoir programmer des applications, des langages évolués ont été conçus. D'autre part, l'accès aux données a également été facilité par la proposition du langage SQL avec différentes

implémentations. Ce langage, très proche de l'anglais parlé et malgré son utilité, reste difficile à écrire par des non informaticiens, surtout quand il s'agit de requêtes complexes de jointures, et de regroupement. En particulier, lorsque l'utilisateur a besoin d'interagir avec les données sans passer par des applications spécifiques. Dans ce cas, plus l'intermédiaire entre l'utilisateur et la base de données est proche de l'être humain (e.g. requête en langage naturel), plus l'accès est facilité. Dans cette section, nous présentons l'interaction avec les bases de données en langage naturel qui permet à l'utilisateur de communiquer plus facilement avec la base de données.

1.3.1 L'accès à la base de données en langage naturel

Pour interagir avec une BDD dans le sens d'accès aux données, nous avons besoin de maitriser le langage SQL qui est difficile de comprendre par un utilisateur ordinaire, et pour rendre cet accès facile, il est nécessaire de convertir son langage naturel en SQL (voir figure 1.9).

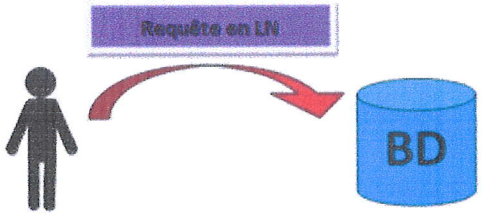

FIGURE 1.9 – Accès à la base de données en langage naturel.

Dans ce qui suit, nous présentons quelques travaux :

Travail de Ikshu Bhalla et Archit Gupta : Les auteurs ont exploré un modèle existant pour générer une requête SQL à partir d'une question en langue naturelle proposée par Xu et al [9]. En générale, ils ont divisé la requête SQL en trois : partie SELECT, opérateur (MAX, MIN, COUNT ...) et WHERE mais ils ont appliqué leur approche uniquement sur la partie SELECT de la requête SQL et ont obtenu de nouveaux résultats et améliorent la précision du modèle de 1,5% pour l'opération d'agrégation et de <2,5% pour la partie de la requête de la colonne SELECT. [10]

Travail de Ruichu Cai et al : Ce travail a combiné des techniques d'apprentissage approfondi et des techniques classiques d'analyse. L'idée est d'essayer d'améliorer le cadre décodeur-décodeur traduisant la langue naturelle en bases de données par des nouvelles techniques incluses dans les phases de codeur et de décodeur respectivement sur le traitement de réseau neuronal qui conscient la grammaire SQL. Les résultats de ce travail montrent des améliorations significatives par rapport aux approches de base pour la traduction automatique standard, en particulier en ce qui concerne l'exactitude des résultats en exécutant les requêtes SQL sur des bases de données réelles.[11]

Travail de Vadim Sheinin et al : Ce travail a présenté une méthodologie semi-supervisée pour traiter les requêtes logiques imbriquées qui nécessitant une succession de select en SQL,

cette méthodologie ne suppose aucune restriction sur la requête d'entrée et sur la structure de la base de données. Elle s'appuyait sur différentes technologies qui effectuent le traitement linguistique d'une requête jusqu'à ce qu'un moteur spécialisé soit capable de produire une série de requêtes SQL renvoyant la réponse dans un processus de calcul ascendant récursif. Les améliorations les plus efficaces dans ce travail est le traitement des opérateurs logiques en langage naturel, et un meilleur traitement linguistique des phrases afin d'identifier correctement les arguments des requêtes SQL.[12]

1.3.2 La génération des résultats des requêtes SQL en langage naturel

Contrairement à l'accès à la base de données en langage naturel par poser des questions par exemple, la génération des résultats des requêtes SQL en langage naturel donne la réponse de la question en langage naturel à l'utilisateur, cette manière de répondre fait une transparence entre la base de données et l'utilisateur et fournit une interface supportée par la langage naturel qui est plus conviviale à l'utilisateur, la figure 1.10 montre le sens de l'interaction.

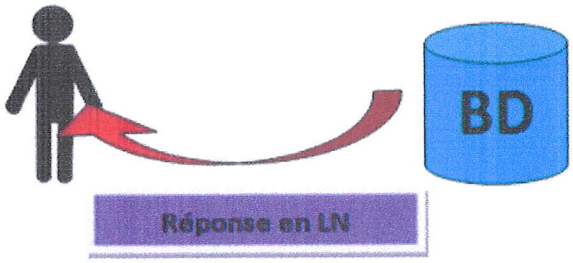

FIGURE 1.10 – La génération des résultats des requêtes SQL en langage naturel.

Dans la littérature, nous n'avons pas trouvé beaucoup de travaux qui s'intéressent à cette direction « BDD → utilisateur » et ce, malgré l'utilité dans différentes situations. Par exemple, la génération des résultats des requêtes SQL en langue naturelle peut exploiter dans des besoins pédagogiques. Le travail de **William J. Holto** a présenté une technique qui traduit les requêtes SQL en anglais, pour les utiliser dans des applications de tutorat intelligent pour le domaine de la construction de requêtes de base de données. Le travail illustre un algorithme de réécriture des graphes (graph-rewriting) basé sur des règles et un ensemble concret de règles permettant de transformer systématiquement les requêtes dans un sous-ensemble de descriptions SQL en anglais sont présentés ; ces règles mise en œuvre dans un système réel et ont montré leur efficacité pour la traduction des requêtes SQL en langage naturel.[13]
Dans le même ordre d'idées, nous mentionnons que certains outils permettent un affichage convivial aux utilisateurs des résultats de requêtes SQL. Par exemple, le SGBD Oracle donne la possibilité à l'utilisateur de formater les résultats SQL par l'ajout des littéraux et d'alias.

Par exemple, pour afficher le plus grand salaire dans l'entreprise, on peut ajouter au résultat la phrase suivante : « le meilleure salaire dans l'entreprise est : ». Malheureusement, ce formatage n'est pas fait automatiquement et l'utilisateur doit écrire les littéraux et alias lui-même.

1.4 L'interaction en langage naturel avec les entrepôts de données

Dans cette section, nous allons présenter l'interaction en LN avec un entrepôt de données qui est un cas particulier de bases de données.

1.4.1 L'accès à l'entrepôt de données en langage naturel

Pour cette partie, nous allons présenter l'accès à l'ED sans utiliser les outils traditionnels comme les outils OLAP mais en utilisant le langage naturel qui facilite l'accès surtout dans les cas où on a besoin d'obtenir les informations rapidement. Un responsable d'entreprise peut facilement exprimer son besoin dans sa propre langue dans le sens de LN vers ED, comme indiqué dans la figure 1.11.

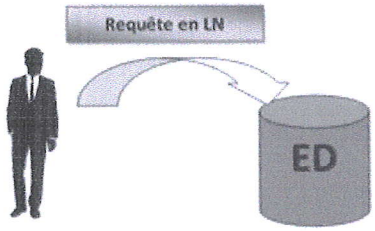

FIGURE 1.11 – Accès en LN à l'ED.

Il existe plusieurs travaux qui se font selon cette direction. Nous en mentionnons deux.
Dans un travail intitulé "Natural Language Interfaces for Data Warehouses" (en français, Interfaces en langage naturel pour les entrepôts de données) de **Nicolas Kuchmann-Beauger** et **Marie-Aude Aufaure**, les auteurs présentent un système de questions-réponses (Q/R) pour des données structurées dans un contexte de BI (business intelligence ou informatique décisionnelle en français). Ce système est un système de réponse aux données stockées en entrepôt, afin de remédier aux lacunes des outils de BI existants et de prendre en compte les nouvelles tendances en matière de partage de contexte et de mobilité dans la recherche d'informations. Le système transforme la requête en LN en une structure de données arborescente, puis l'arbre d'analyse syntaxique est associé à une représentation logique interne qui sera traduite dans la requête cible, qui sera ensuite exécutée. Le résultat de la requête est utilisé pour générer des graphiques et des rapports automatiquement pour les utilisateurs. Les auteurs ont proposé une expérimentation via deux interfaces : un client HTML et une

application iPhone / iPad.[14]

Le travail de **M. Asif Naeem** et al a présenté une approche appelée QueGen (Query Generator) qui génère des requêtes OLAP basées sur la spécification fournie en langue anglaise. Cette approche permet de générer toutes les requêtes OLAP courantes à partir de la spécification LN donnée par les utilisateurs sous forme de quelques déclarations en anglais simple, et après une analyse sémantique et un mappage des informations associées, QueGen génère les requêtes OLAP prévues pouvant être exécutées directement sur des entrepôts de données.[15]

1.4.2 Génération des résultats des requêtes OLAP en LN

L'utilisateur d'un ED veut toujours obtenir les informations avec une manière facile pour faire des analyses, et il n'y a rien de plus facile que de fournir cette information en langage naturel. Donc la direction est de l'ED vers LN comme le montre la figure suivante :

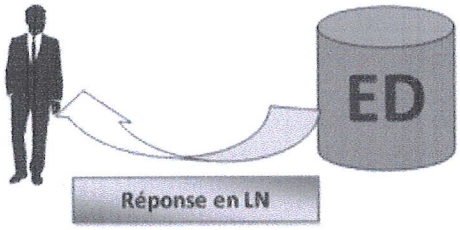

FIGURE 1.12 – Réponse en LN à la requête d'un utilisateur.

Remarque
A notre connaissance et selon la recherche bibliographique effectuée, il n'y a pas de travaux qui sont faits pour l'accès en langage naturel aux ED.

1.5 Génération des requêtes OLAP en LN

D'une manière classique, l'interaction entre l'utilisateur et les BDD se fait à travers des applications qui permettent au utilisateur d'interagir que ce soit dans le sens d'accès aux données ou de recevoir des réponses par le biais des composants graphiques comme les tableaux, les boîtes combinées, les boutons...etc. Cependant, il y a des cas où on ne peut pas utiliser ces applications.

La génération des requêtes en langage naturel a une grande utilité pour les BDD où elle permet à un utilisateur d'obtenir les informations où et quand il le souhaite. Son efficacité apparaît mieux dans le cas d'une prise de décisions en utilisant l'entrepôt de données.

Dans ce travail, nous nous intéressons à la génération des requêtes en langage naturel dans le contexte de l'entrepôt de données en particulier l'OLAP. L'utilité de cela est de permettre à l'utilisateur de faire de l'OLAP dans toutes les situations soit par ordinateur ou par un périphérique mobile (téléphone, tablette, etc.) avec comme interface des requêtes en langage naturel que l'utilisateur pourra sélectionner à partir d'une liste, voire qu'il peut exprimer verbalement et être traduite en texte par un système de reconnaissance vocale.

Avec la démocratisation des petites machines qui permettent l'accès aux données à tout moment, notre objectif est de développer un outil d'analyse OLAP qui soit adaptable à de petites machines (Smartphone, Tablette...). Cet outil est basé sur une liste de requêtes pré-générée en langage naturel et affichées sous forme des liens hypertextes. A partir d'une telle liste, c'est à l'utilisateur de choisir les requêtes dont il a besoin. Ceci est similaire à ce qui se fait dans les assistants qui préparent une liste de suggestions pour aider l'utilisateur de trouver l'information par poser des questions comme l'assistant du site "Software Advice". La figure 1.13 représente une capture d'écran d'une interaction entre l'assistant de "Software Advice" et l'utilisateur qui pose deux questions : 1." What is the most recommended tools of OLAP", 2. "Give me the download link of Dundas".

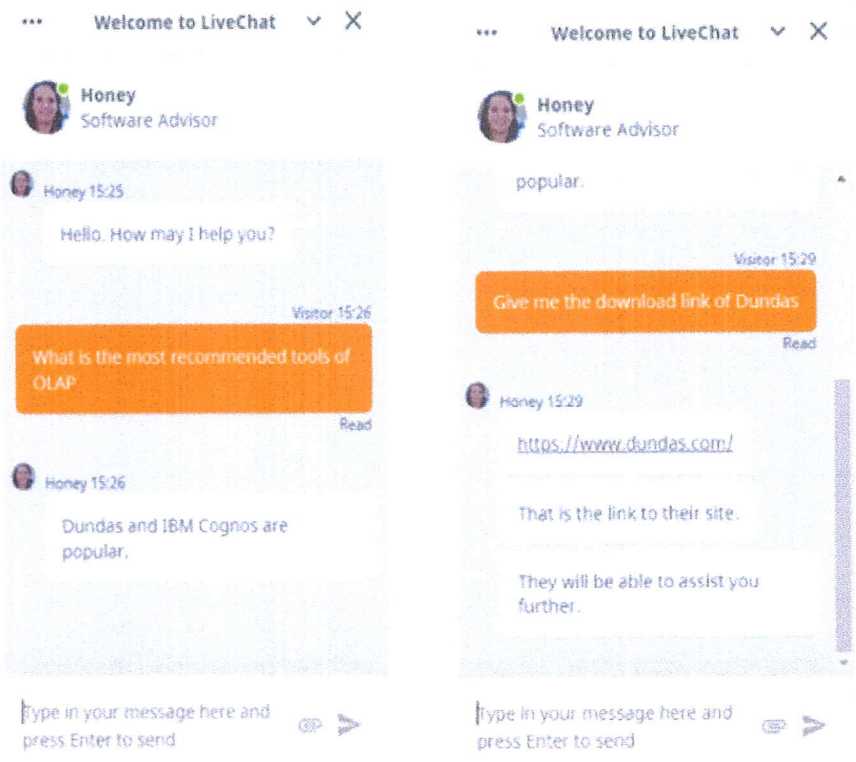

FIGURE 1.13 – Exemple d'un assistant de site.

L'objectif de notre travail est de fournir un accès rapide et facile à l'utilisateur, en particulier dans les cas où il ne peut pas utiliser un ordinateur ou où le temps est insuffisant pour entrer

une requête manuellement, il est préférable de choisir la requête à partir d'une liste prédéfinie, comme lors de conférences. Un chef d'entreprise devient plus à l'aise s'il peut accéder aux informations par un simple clic sur son téléphone puis d'analyser le résultat pour prendre sa décision plus rapidement.

Notons que notre travail n'est pas limité au contexte d'aide à la décision. Il peut également être exploité pour répondre aux besoins académiques en aidant les étudiants à apprendre le langage SQL en exprimant le résultat d'un script SQL en langage naturel.

1.6 Conclusion

Dans ce chapitre, nous avons présenté une entrée dans notre travail en le divisant en quatre sections principales. La première section est un aperçu de l'entrepôt de données avec ses différentes notions, et OLAP qui est un outil pour analyser les données qui relie l'utilisateur et l'ED. Dans la deuxième section, nous avons parlé de l'interaction en langage naturel avec les bases de données dans les deux sens, puis nous avons spécifié l'interaction dans l'ED dans la troisième section. Enfin, nous avons présenté l'utilité de notre travail et ses principaux objectifs.

CHAPITRE 2

Modélisation d'un entrepôt de données générique

2.1 Introduction

Dans le chapitre précédent, nous avons présenté l'objectif de notre travail comme étant la génération de requêtes en langage naturel pour un entrepôt de données. Aussi, nous visons la génération de ces requêtes pour n'importe quel entrepôt. Pour aboutir à cet objectif, nous devons passer par une représentation d'un entrepôt générique qui sera instancié au besoin par des analystes sous la forme d'un entrepôt concret, sur la base duquel les requêtes seront générées.

Dans la littérature des entrepôts de données, il existe principalement deux types de schémas multidimensionnels : le schéma en étoile et le schéma en flocon de neiges. Dans ce travail, nous allons présenter une modélisation d'entrepôt générique selon le schéma en flocon de neige.

Le schéma flocon de neige est, comme mentionné dans le chapitre précédant, caractérisé par la normalisation des tables de dimensions et par l'explicitation des hiérarchies dans le sens où chaque niveau hiérarchique est représenté par une table. Ainsi, ce schéma est normalisé selon la troisième forme normal (3FN). Dans notre travail, nous adoptons ce schéma à cause de l'explicitation des hiérarchies, dans le sens que chaque table correspond à un niveau hiérarchique, contrairement au schéma en étoile dans lequel les hiérarchies ne sont pas explicites. En plus, comme notre objectif est de générer des requêtes en langage naturel, nous avons besoin d'une description du schéma de l'entrepôt en termes de tables, de colonnes, de hiérarchies, etc.

Dans le domaine des bases de données et des entrepôts de données, cette description correspond au dictionnaire de données de la base. Toutefois, comme nous le verrons plus loin, nous avons besoin d'enrichir cette représentation par des désignations en langage naturel des composants de l'entrepôt. C'est pour cela, nous avons besoin d'un schéma unifié qui décrit l'entrepôt en termes de tables et colonnes techniques ainsi qu'en termes de leur équivalent en

langage naturel. Pour cela nous avons fait le choix du langage XML.

XML est le fruit d'une évolution qui s'est déroulée sur une vingtaine d'années sur le langage SGML (Standard Generalized Markup Language) et il hérite aussi le langage qui assurant la création des pages web, nommé HTML (Hyper Text Markup Language) [16]. Un document XML est un fichier texte qui peut stocker les données ayant une structure bien formée et éventuellement respectant un schéma (validité). Les données peuvent être structurées ou semi-structurées. XML peut donc être considéré comme une base de données hiérarchique.

Nous allons présenter un exemple illustratif d'un fichier XML concret qui contient les méta-données d'un entrepôt de donnée, pour tenter de définir un fichier XML générique capable de décrire toutes les instances possibles d'un ED, et nous allons présenter ensuite un schéma XML (.XSD) qui contient l'ensemble des règles qu'un fichier XML générique doit respecter.

Dans ce chapitre, nous allons commencer par un rappel sur les concepts de schéma en flocon de neige, et puis nous allons présenter une vue sur l'XML au cours de laquelle nous allons proposer un fichier XML d'un exemple concret d'ED. Nous terminons ce chapitre par la construction d'un fichier XML générique validé par notre document XSD.

2.2 Exemple illustratif d'un schéma en flocon de neige

Dans le chapitre précédent, nous avons mentionné le schéma en flocon de neige, une structure qui relie la table de faits à ses dimensions où chaque table de dimension est éclatée en un ensemble de hiérarchies. Cette structure facilite l'alimentation pour les utilisateurs.
Dans notre travail et pour construire le modèle XML générique d'un ED, nous utiliserons le schéma en flocon de neige qui nous aidera à mieux montrer les différentes hiérarchies de dimensions.
La figure 2.1. montre un exemple concret de schéma de flocon de neige illustrant les différents types de hiérarchies de dimensions.

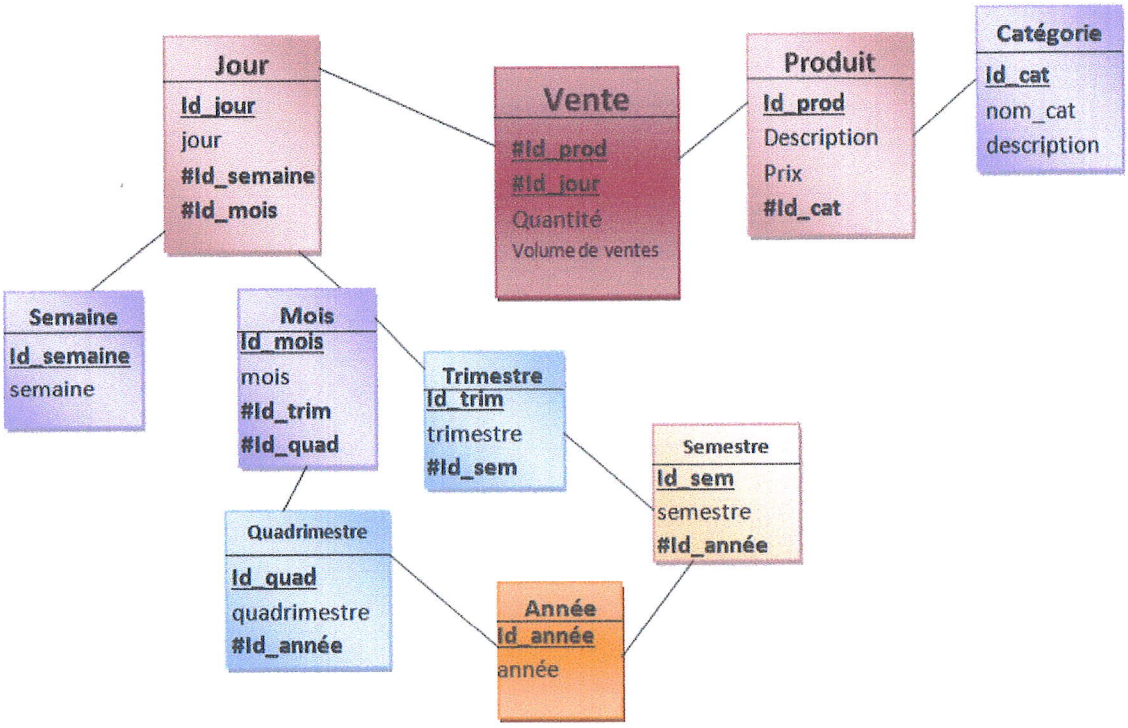

FIGURE 2.1 – Exemple d'un schéma en flocon de neige.

La table de faits dans cet exemple est "Vente" qui contient comme clé une concaténation de deux clés étrangères faisant référence aux tables "Produit" et "Jour". On peut analyser la quantité et le volume des ventes par date et par produit qui sont les deux dimensions de table de fait. La dimension "Produit" est hiérarchisée en deux niveaux : Produit et Catégorie,; par contre, la dimension "Date" est hiérarchisée en plusieurs niveaux. Elle est composée de trois hiérarchies : la première hiérarchie (Jour-Semaine) est parallèle aux deux autres hiérarchies; les deuxième et troisième hiérarchies sont des hiérarchies alternatives (jour-mois-trimestre-semestre-année) ou (jour-mois-quadrimestre-année).

2.3 Conception de schéma XML d'ED

2.3.1 C'est quoi XML ?

XML (Extensible Markup Language) est un langage de balisage générique extensible décrivant une classe d'objets de données appelés documents XML [17]. Il permet de créer des jeux de balises destinés à décrire et qualifier le contenu d'un document, et aussi nous permet de créer nos propres balises. Le rôle de ces balises de description est permettre de définir la structure et la signification de données. [18]

2.3.2 Pourquoi XML ?

La question que l'on peut se poser c'est pourquoi utiliser XML et à quoi peut-il bien servir ? XML possède un certain nombre d'avantage, il est extensible où il permet de définir des balises à votre choix (le nombre de balises n'est pas figé), il ne mélange pas l'information contenus dans le document avec sa représentation ainsi peut extraire les informations de document plus facilement, il a des modèles de données tels que la DTD et Schéma XML qui permettent la vérification automatique d'un document XML pour qu'il soit conforme (valide) par rapport à son schéma.
En plus, XML peut stocker des données et donc des méta-données d'une base de données, et comme dans notre cas, les métadonnées d'un entrepôt de données. Enfin, dans le contexte de notre travail, XML nous permet d'enrichir la description de l'entrepôt par des descriptions en langage naturel qui est utile pour la génération des requêtes en langage naturel. De plus, comme XML est lisible, nous pouvons dire que XML est un langage auto-descriptif, c'est-à-dire qu'il se suffit à lui-même. [19]

2.3.3 Exemple d'un document XML

Dans cette section, nous représentons l'exemple concret de schéma de flocon de neige précédent (figure 2.1) en format XML. En général, dans un fichier XML, nous commençons par le prologue <?xml version= "1.0" encoding="UTF-8" ?> qui définit la version et le type de codage utilisé dans ce fichier. Ensuite, nous définissons une balise racine qui englobe tous les éléments du document. Dans cet exemple, on peut diviser le fichier XML en deux parties la première partie définit la table de fait «*ventes*» et la deuxième partie présente les dimensions *Produit* et *Date*.

- La première partie est illustrée en figure 2.2. Comme nous pouvons l'observer dans cette figure, l'élément "fact" avec ces sous-éléments est une projection de la table de fait "vente". Elle est constituée de deux balises principales : La balise «Key» qui se compose de deux clés : la clé de la première table dans la dimension Produit *"Id_prod"*et la clé de la première table de la dimension Date *"Id_jours"*. Aussi, la balise *"Mesures"* contient deux mesures, la première mesure Quantité possède comme fonctions d'agrégation : le Maximum et le Minimum, et la deuxième mesure *Volume de ventes* possède quatre fonctions : Maximum, Minimum, Moyenne, Somme.

```
<?xml version="1.0" encoding="UTF-8"?>
 <!-- New document created with EditiX at Tue Apr 16 20:08:40 CEST 2019 -->
 <Model>
  <Fact factname="Ventes">
      <Key>
           <Dimension d_name="Produit"    Dkey="Id_prod"/>
           <Dimension d_name="Date"        Dkey="Id_jours"/>
      </Key>
      <Mesures>
        <Mesure  technical_mesurename="Quantite" functional_mesurename="Quantité">
           <Aggregation_func  functional_agg_name="Maximum" technical_agg_name="Max"/>
           <Aggregation_func  functional_agg_name="Minimum" technical_agg_name="Min"/>
        </Mesure>
        <Mesure technical_mesurename="volume_ventes" functional_mesurename="Volume de ventes">
              <Aggregation_func functional_agg_name="Maximum" technical_agg_name="Max"/>
              <Aggregation_func  functional_agg_name="Minimum" technical_agg_name="Min"/>
              <Aggregation_func  functional_agg_name="Average" technical_agg_name="Avg"/>
              <Aggregation_func  functional_agg_name="Sum" technical_agg_name="Sum"/>
         </Mesure>
      </Mesures>
  </Fact>
```

FIGURE 2.2 – La partie de document XML définissant la table de fait "Vente"

- La deuxième partie présente les dimensions Produit et Date. L'élément «*Date*» est composé de sept balises nommées «*Table*» chacune contenant deux sous-éléments : « *Attribute* » et « *Rollup_to* ». En prenant la table « *Jour* » comme exemple, elle contient un attribut appelé jour et deux *Rollup_to*, une est référencée à la table « *Semaine* » par l'attribut *Fkey= "Id_semaine"*, et l'autre référencée à la table « *Mois* » par l'attribut *Fkey= "Id_Mois"*. Les éléments *Rollup_to* des dernières tables dans chaque hiérarchie ne contiennent pas l'attribut Fkey.

```
<Dimensions>
<Dim Dname="Date">
    <Table Tname="Jour"   Ttech_name="J" Tkey="Id_jours">
           <Attribute>jour</Attribute>
           <Rollup_to     NTname="Semaine"  Fkey="Id_semaine" />
           <Rollup_to     NTname="Mois"  Fkey="Id_Mois" />
    </Table>
    <Table Tname="Semaine" Ttech_name="Sem" Tkey="Id_semaine" >
           <Attribute>semaine</Attribute>
           <Rollup_to NTname="all"/>
    </Table>
    <Table Tname="Mois" Ttech_name="M" Tkey="Id_Mois" >
           <Attribute>mois</Attribute>
           <Rollup_to NTname="Trimestre" Fkey="Id_trim"/>
           <Rollup_to NTname="Quadrimestre" Fkey="Id_quad"/>
    </Table>
   <Table Tname="Trimestre"  Ttech_name="T" Tkey="Id_trim">
          <Attribute>trimestre</Attribute>
          <Rollup_to NTname="Semestre" Fkey="Id_sems"/>
  </Table>
  <Table Tname="Quadrimestre" Ttech_name="Q" Tkey="Id_quad">
           <Attribute>quadrimestre</Attribute>
           <Rollup_to NTname="Année" Fkey="Id_Année"/>
  </Table>
  <Table Tname="Semestre" Ttech_name="Semes" Tkey="Id_sems">
        <Attribute>semestre</Attribute>
        <Rollup_to NTname="Année" Fkey="Id_Année"/>
  </Table>
  <Table Tname="Année" Ttech_name="A" Tkey="Id_Année">
        <Attribute>année</Attribute>
        <Rollup_to NTname="all"/>
  </Table>
 </Dim>
```

FIGURE 2.3 – La partie de document XML définissant la dimension "Date"

Pour la dimension « Produit », la première table est « *Produit* » qui possède deux attributs *Prix* et *description* et un élément *Rollup_to* qui référence par l'attribut *Fkey="Id_cat"* à la table Catégorie qui est la deuxième table de cette dimension contenant deux attributs Nom de catégorie et Description de catégorie. L'attribut « *Tname* » est le nom de la table en LN (c'est le nom utilisé dans la requête en LN) et l'attribut « *Ttech_name* » est le nom technique de la table qui est utilisé dans la requête SQL.

```
    <Dim Dname="Produit">
    <Table Tname="Produit" Ttech_name="P" Tkey="Id_prod">
         <Attribute>Description</Attribute>
         <Attribute>Prix</Attribute>
         <Rollup_to     NTname="Catégorie"  Fkey="Id_cat" />
  </Table>
   <Table Tname="Catégorie" Ttech_name="C" Tkey="Id_cat" >
          <Attribute>Nom de catégorie</Attribute>
          <Rollup_to     NTname="all"  Fkey="Id_cat" />
   </Table>
  </Dim>

  </Dimensions>
</Model>
```

FIGURE 2.4 – La partie de document XML définissant la dimension "Produit"

2.3.4 Création de schéma XML

Le schéma XML est un standard qui décrit la structure du document XML. Il définit les types de données représentées dans le document, la syntaxe et la sémantique, il permet aussi de valider un document XML en vérifiant si celui-ci est conforme aux règles décrites dans le document XSD (XML Schema Definition).[20]

Nous allons présenter dans cette partie le schéma XSD de notre document XML qui décrit le modèle d'un ED. Nous, divisons le document en deux parties : la partie qui décrit les faits et la partie qui décrit les dimensions.

La figure suivante illustre la représentation d'élément « Fact ». On commence par la définition de la racine du schéma « *schema* » qui est dans l'espace de noms des schémas XML du W3C (`http://www.w3.org/2001/XMLSchema`),*elementFormDefault="qualified"* veut dire que nos balises n'auront pas de namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<xsd:element name="Model">
<xsd:complexType>
        <xsd:sequence>
    <xsd:element ref="Fact" maxOccurs="unbounded" minOccurs="1"/>
    <xsd:element ref="Dimensions"/>
        </xsd:sequence>
        </xsd:complexType>
</xsd:element>
<xsd:element name="Fact">
</xsd:element>
<xsd:element name="Key">
         <xsd:complexType>
                 <xsd:sequence>
               <xsd:element ref="Dimension" maxOccurs="unbounded" minOccurs="2"/>
      </xsd:sequence>
         </xsd:complexType>
</xsd:element>
<xsd:element name="Dimension">
        <xsd:complexType  id="ID_fact" >
                 <xsd:attribute name="d_name" type="xsd:string" use="required"/>
       <xsd:attribute name="Dkey" type="xsd:string" use="required"/>
          </xsd:complexType>
</xsd:element>
```

FIGURE 2.5 – Partie 01 de "Fact" de document XSD

Le document indique qu'on peut avoir un nombre illimité de tables de faits. L'élément *Fact* contient comme sous-éléments :

- « *Key* » qui représente la clé du fait et elle possède un nombre illimité de la balise « *Dimension* » qui représente avec ses deux attributs *d_name* (le nom de la dimension du fait) et *Dkey*(la clé du dimension), l'ensemble des dimensions qui sont liées à la table de fait et la concaténation de leurs différentes clés représente la clé de la table de Fait.

- « *Mesures* » qui contient plusieurs occurrences de l'élément « *Mesure* », chacune ayant un sous-élément *"Aggregation_func"* qui représente l'ensemble de fonctions d'agrégation associées à une mesure avec un nom technique et un nom fonctionnel. La balise Mesure contient également deux attributs, un nom fonctionnel *"functional_mesurename"* et un nom technique *"technical_mesurename"*, comme indiqué dans la figure ci-dessous (2.6.)

```
<xsd:element name="Mesures">
        <xsd:complexType>
                <xsd:sequence>
              <xsd:element ref="Mesure" maxOccurs="unbounded" minOccurs="1"/>
                 </xsd:sequence>
       </xsd:complexType>
</xsd:element>
<xsd:element name="Mesure">
         <xsd:complexType >
       <xsd:sequence>
                <xsd:element ref="Aggregation_func" minOccurs="1" maxOccurs="unbounded"/>
       </xsd:sequence>
       <xsd:attribute name="technical_mesurename" type="xsd:string" use="required"/>
       <xsd:attribute name="functional_mesurename" type="xsd:string" use="required"/>
        </xsd:complexType>
</xsd:element>
<xsd:element name="Aggregation_func" >
<xsd:complexType>
   <xsd:attribute name="functional_agg_name" type="xsd:string" use="required"/>
   <xsd:attribute name="technical_agg_name" type="xsd:string" use="required"/>
</xsd:complexType>
</xsd:element>
```

FIGURE 2.6 – Partie 02 de "Fact" de document XSD.

La section « Dimensions » (figure 2.7) définit l'ensemble des dimensions de l'ED. Chacune possède comme sous-élément « *table* » et un attribut représentant son nom «*Dname* ». Une dimension peut contenir plusieurs tables et chaque table contient un attribut « *Tname* » qui représente son nom et un autre « *Tkey* » qui est sa clé.« *Attribute* » est une balise fille de la balise « *table* » et exprime les différents attributs d'une table de dimension. L'autre balise est « *Rollup_to* » qui indique la table suivante de la table courante contenant deux attributs :« *NTname* » le nom de la table suivante dans la hiérarchie, « *Fkey* » la clé étrangère, c'est un identificateur qui fait référence à la clé de la table suivante pour l'associer à la table courante.

```
<xsd:element name="Dimensions">
<xsd:complexType>
        <xsd:sequence>
      <xsd:element ref="Dim" maxOccurs="unbounded" minOccurs="1"/>
        </xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="Dim">
        <xsd:complexType>
                <xsd:sequence>
               <xsd:element ref="Table" maxOccurs="unbounded" minOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="Dname" type="xsd:string" use="required"/>
        </xsd:complexType>
</xsd:element>
<xsd:element name="Table">
        <xsd:complexType >
                <xsd:sequence>
              <xsd:element ref="Attribute" minOccurs="1" maxOccurs="unbounded"/>
              <xsd:element ref="Rollup_to" maxOccurs="unbounded" minOccurs="1"/>
       </xsd:sequence>
       <xsd:attribute name="Tname" type="xsd:string" use="required"/>
       <xsd:attribute name="Ttech_name" type="xsd:string" use="required"/>
        <xsd:attribute name="Tkey" type="xsd:ID"   use="required"/>
        </xsd:complexType>
</xsd:element>
<xsd:element name="Attribute" type="xsd:string"/>
<xsd:element name="Rollup_to">
          <xsd:complexType>
              <xsd:attribute name="NTname" type="xsd:string" use="required"/>
               <xsd:attribute name="Fkey"    type="xsd:IDREF" />
           </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

FIGURE 2.7 – Partie de "Dimensions" de document XSD.

2.4 Modèle XML générique d'entrepôt de données

Nous présenterons dans cette partie le schéma XML générique d'un entrepôt de données qui constitue un moule pour toute ED spécifique. Il nous permet plus tard de l'utiliser pour générer les différentes requêtes possibles.

Dans la figure 2.8 qui illustre ce modèle, nous avons commencé par lier le document XML avec le fichier XSD dans la balise« *Model* » qui est la balise racine. Cette dernière, contient deux éléments principaux.

- L'élément « *Fact* » représente la table de fait qui contient un attribut *factname* exprimant son nom et deux sous-éléments :
  - Sa clé *"Key"* est la concaténation des clés de dimensions
  - *"Mesures"* qui contient les différentes mesures du fait avec leurs fonctions d'agrégation.
- L'élément *"Dimensions"* qui contient plusieurs dimensions, chacune ayant différents niveaux (*"Table"*) se rapportant les unes aux autres par la balise *"Rollup_to"* qui contient un attribut de type IDREF « *Fkey* » référençant à l'attribut de type ID de balise « *Table*».

```
<?xml version="1.0" encoding="UTF-8"?>
<Model xsi:noNamespaceSchemaLocation="modéle générique schéma2.xsd"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Fact factname="Fact_name">
    <Key >
      <Dimension d_name="Dimension_name" Dkey="Dimension_key" />
    </Key>
     <Mesures>
       <Mesure technical_mesurename="Mesure_name_tech" functional_mesurename="Mesure_name_func">
         <Aggregation_func functional_agg_name="agg_name_func" technical_agg_name="agg_name_tech"/>
       </Mesure>
     </Mesures>
  </Fact>
  <Dimensions>
     <Dim Dname="Dimension_name">
         <Table Tname="Table_name" Tkey="Table_key" Ttech_name="Table_tech_name">
            <Attribute>Attribute name</Attribute>
            <Rollup_to NTname="Next_table_name" Fkey="Table_key" />
         </Table>
     </Dim>
   </Dimensions>
</Model>
```

FIGURE 2.8 – Modèle générique d'entrepôt de données.

L'arbre XML dans la figure 2.9 est composé de nœuds. Les éléments dans le document XML sont représentés sous forme de rectangles et les attributs sous forme d'ellipses. Le sommet de cet arbre est le nœud racine du document «*Model*», possède deux nœuds fils : *"Fact"* et *"Dimensions"* qui ont également des nœuds fils. Les feuilles d'arbres sont les valeurs des attributs des éléments.

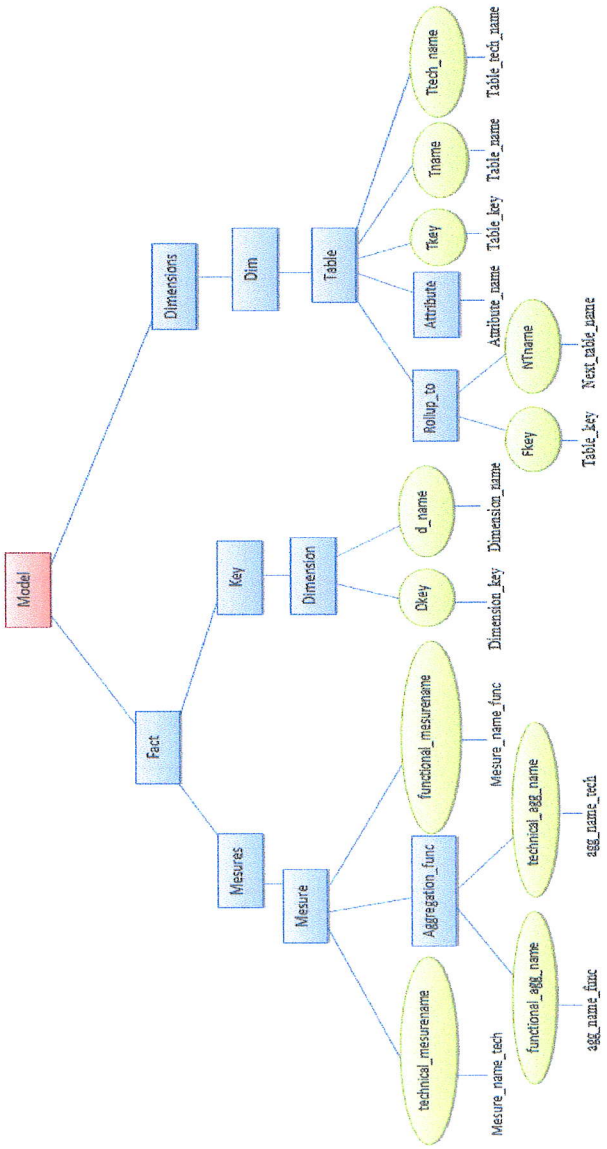

FIGURE 2.9 – L'arbre du document XML.

2.5 Conclusion

Dans ce chapitre, nous avons commencé par parler du schéma de flocon de neige que nous avons choisi pour représenter l'entrepôt de données, puis nous sommes passés à un exemple concret d'ED représenté en XML et nous l'avons validé avec un XSD que nous avons présenté par la suite. Enfin, nous avons suggéré un modèle pour un ED générique au format XML validé avec notre document XSD qui sera la base de la génération de requêtes en langage naturel.

CHAPITRE 3

Génération de requêtes en langage naturel

3.1 Introduction

Dans le chapitre précédent, nous avons présenté un modèle générique d'entrepôt de données qui permet d'instancier tout exemple concret et d'appliquer à celui-ci des règles déjà définies. Nous avons choisi XML comme structure pour représenter le modèle.

Dans ce chapitre, nous examinerons ce modèle pour l'utiliser dans la génération de requêtes en LN en créant un algorithme qui exploite les données représentées en XML et les stocke dans une structure relationnelle, puis stocke les requêtes en langage naturel ainsi que leur équivalent en SQL après les avoir générées. Ce chapitre représente le noyau de notre travail.

Nous rappelons tout d'abord l'exemple du cube de vente, puis nous présentons la partie principale de l'algorithme "génération de requêtes en LN", en montrant quelques exemples. Après cela, nous montrerons comment générer les requêtes SQL et, enfin, présenter l'algorithme.

3.2 Rappel du cube des ventes

Dans le chapitre précédent, nous avons présenté un exemple de schéma en flocon de neige de *"Ventes"*. Ce cube représente le cube de base, autrement dit, le cube qui affiche les données les plus détaillées. A partir de ce cube, il est possible de générer plusieurs cuboïdes en exploitant les hiérarchies de chaque dimension. A partir du cube de base, et en regroupant les données par rapport à un niveau hiérarchique, l'analyste passe à un cuboïde moins détaillé et ainsi de suite, jusqu'à atteindre le cuboïde le moins détaillé qui consiste à appliquer la fonction d'agrégation à l'ensemble des données de la table de faits.

L'ensemble des cuboïdes avec le cube de base forment un treillis. Par exemple, la figure (3.1) représente le treillis du cube des ventes

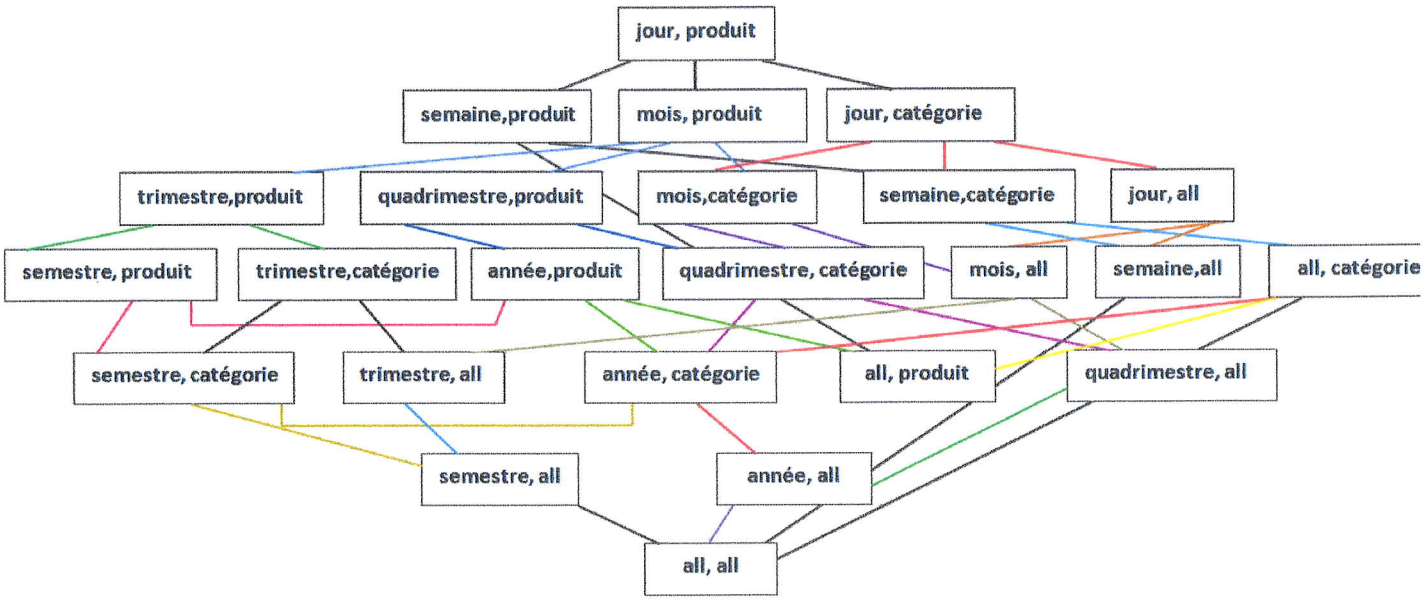

FIGURE 3.1 – Le treillis du cube des ventes.

En outre, comme mentionné dans le schéma générique d'un entrepôt de données (voir chapitre 2), un cube peut contenir plusieurs mesures et à chaque mesure, différentes fonctions d'agrégation peuvent s'appliquer. Par exemple, on peut appliquer la somme à la quantité vendue et le maximum et minimum à la quantité.

Ainsi, à chaque cuboïde, on peut appliquer une combinaison d'une mesure et d'une fonction d'agrégation pour afficher les données. Cet affichage est le résultat d'exécution d'une requête SQL ou MDX (langage multidimensionnel) selon le modèle d'implémentation.

Dans notre cas, nous nous intéressons à la génération des requêtes en SQL. Dans ce cas, comme l'utilisateur peut choisir une combinaison de fonctions et de mesures, nous faisons le choix de générer une requête par cuboïde pour une fonction d'agrégation et pour une mesure donnée. Ce choix est justifié par un besoin de flexibilité : si nous générons une seule requête par cuboïde pour toutes les mesures et fonctions à la fois, il se pourrait que la requête soit inutile et il devient difficile de l'assimiler. Par contre, si chaque requête correspond à une seule mesure et fonction, elle sera simple à comprendre, et il reste possible de la combiner avec les autres requêtes pour un besoin complexe d'analyse.

Notons que les requêtes auxquelles nous sommes intéressées ne font pas usage de sélection de données, cas, les possibilités de sélection de données peuvent s'avérer très nombreuses selon

les différentes valeurs de chaque colonne de sélection. De ce fait, nous pensons qu'il est plus judicieux de permettre cette sélection au niveau de l'interface d'affichage des résultats d'une requête au lieu qu'elle soit exprimée dans la requête.

Dans ce qui suit, nous montrons comment générer les requêtes en langage naturel pour chaque cuboïde à partir du cube de base ainsi que la génération de la requête correspondant en SQL pour permettre d'afficher le résultat.

3.3 Génération des requêtes en langage naturel

Dans cette section, nous présentons comment nous pouvons obtenir des requêtes écrites dans le langage naturel où notre choix dans ce travail est la langue anglaise.

a) **Principe général**

L'algorithme que nous proposons est basé sur le modèle d'ED présenté dans le chapitre précédent en tant qu'entrée pour générer les requêtes en LN. Comme expliqué dans la section précédente, chaque requête correspond à une combinaison de fonction et de mesure et s'applique à un cuboïde (y compris le cube de base).

Ainsi, pour générer une requête, nous choisissons d'abord la mesure utilisée pour analyser les données, puis la fonction d'agrégation pour la mesurer. Le cube de base étant le point d'entrée, correspond à une requête d'une mesure et d'une fonction. Par la suite, pour cibler à chaque fois un cuboïde différent, nous effectuons un déplacement correspondant à une opération de forage vers le haut, par rapport à une seule dimension, donnant lieu à une nouvelle requête. Nous répétons l'opération jusqu'à atteindre le niveau "all" de cette dimension, autrement dit, jusqu'à ce que la dimension soit éliminée du cuboïde.

Nous nous déplaçons ensuite pour fixer une dimension après une autre jusqu'à atteindre la dernière, nous prenons l'ordre du niveau atteint dans chaque dimension sur un tableau des ordres (un exemple de ce tableau dans le tableau 3.1) et nous parcourons avec elles la matrice des requêtes id contenant tous les ordres possibles (voir le tableau 3.2), une fois nous obtenons l'id de requête, on récupère la ligne contenant les noms des niveaux de l'id correspondant à partir de la matrice des noms (l'exemple de cette matrice est dans le tableau 3.4) dans un tableau de chaines de caractères (voir l'exemple dans le tableau 3.3). Ensuite, nous trions les deux tableaux dans un ordre décroissant. Le niveau "all" est un niveau fictif indiquant l'absence de la dimension et nous le représentons par un '0' dans le tableau d'ordre.

Dimension date	Dimension produit
1	1

TABLE 3.1 – Exemple de tableau des ordres des tables des dimensions.

Le 1 dans la dimension date représente l'ordre de la table jour et le 1 dans la dimension produit représente l'ordre de la table produit.

cuboïdes	date	produit
1	1	1
2	1	2
3	1	0
4	3	1
5	3	2
...	...	...
23	7	0
24	0	0

TABLE 3.2 – Exemple de matrice des ordres des tables des dimensions.

Dimension date	Dimension produit
jour	produit

TABLE 3.3 – Exemple de tableau des ordres des noms des tables des dimensions

b) **Forme d'une requête en langage naturel**

Une requête générée en langage naturel possède deux formes :

- ***The*** F_{ij} ***of*** M_i où
  - M_i : la mesure choisie en langage naturel
  - F_{ij} : la fonction d'agrégation choisie en langage naturel

  si toutes les dimensions sont absentes.
- ***The*** F_{ij}***of*** M_i ***by*** $Tablename_1$ ***and*** $Tablename_2$ ***and*** .....où
  - $Tablename_k$: le niveau courant de la dimension k dans le cuboïde, exprimée sous son nom en langage naturel et dans le cas où la dimension existe dans le cuboïde.

c) **Exemple de requêtes en langage naturel**

Dans notre exemple, le cube de ventes contient deux dimensions. Le nombre de cuboïdes sera donc :

- Dimension Date : 1 (jour) + (1 * 2) (semaine et mois) + (1 * 2) (trimestre et quadrimestre) +1 (semestre) +1 (année) +1 (all) = 8.
- Dimension Produit : 1 (produit) +1 (catégorie) +1 (all) = 3.

Cela signifie que le nombre de requêtes est 3 * 8 = 24 requêtes. Avant de présenter les requêtes, supposons que les niveaux des dimensions en langage naturel (ici en anglais) sont les suivants :
quantité → quantity, produit → product, catégorie → category, jour → day, semaine → week, mois → month, trimestre → quarter, quadrimestre → fourmonth, année → year.
Des exemples de requêtes

- *The* Maximum *of* quantity.
- *The* Maximum *of* quantity *by* product.
- *The* Maximum *of* quantity *by* category.
- *The* Maximum *of* quantity *by* month.
- *The* Maximum *of* quantity *by* product *and* day.
- *The* Maximum *of* quantity *by* product *and* week.
- *The* Maximum *of* quantity *by* product *and* month.
- *The* Maximum *of* quantity *by* product *and* year.
- *The* Maximum *of* quantity *by* category *and* day.
- *The* Maximum *of* quantity *by* category *and* week.
- *The* Maximum *of* quantity *by* category *and* quarter
- *The* Maximum *of* quantity *by* category *and* year.

3.4 Génération des requêtes en SQL

Dans la section précédente nous avons présenté comment générer les requêtes en langage naturel d'un ED, et pour montrer l'intérêt de ces requêtes dans un système d'aide à la décision, nous allons clarifier dans cette section comment générer des requêtes SQL équivalentes aux requêtes en langage naturel.

a) **Principe général**
A base de la forme de requête en langage naturel on peut diviser la requête SQL en 4 clauses : *SELECT*, *FROM*, *WHERE* et *GROUP BY*.
Après le parcours dans le schéma en flocon de neige de l'ED (le cube) nous allons obtenir tous les cuboïdes possibles dont nous stockons la description dans une structure de données matricielle (le tableau 3.4) où chaque ligne référence les niveaux de regroupement des dimensions. Pour chaque cuboïde traité, nous générons la requête SQL en même temps que les requêtes en langage naturel.

cuboïdes	date	produit
1	jour	produit
2	jour	catégorie
3	jour	all
4	mois	produit
5	mois	catégorie
...	...	...
23	année	all
24	all	all

TABLE 3.4 – Exemple qui explique la structure matricielle qui stocke les possibilités de cubes.

Nous remplissons d'abord la clause *SELECT* par les attributs des niveaux utilisés dans le regroupement. Ensuite la clause *FROM* contient les tables de regroupement en plus de la table de fait. La clause *GROUP BY* qui est similaire à la clause *SELECT*, où on va regrouper avec les mêmes attributs sélectionnés dans la clause *SELECT*. Enfin, la clause *WHERE* est la plus difficile, car, pour un niveau donné d'un cuboïde, il est nécessaire d'inclure dans la clause *WHERE* toutes les jointures entre ce niveau et le fait. Pour résoudre ce problème, nous utilisons une structure de données appelée «Dim_table» (voir tableau 3.1). Cette structure définit l'Id, le nom, l'id de table précédente, le nom technique et le nom en LN de chaque table. Nous détaillons ci-dessous l'utilisation de cette structure à travers un exemple

b) **Exemple de requête SQL**

Soit la requête suivante :

Select Max(Quantite), Cat.Nom, Cat.Description , Jour.jour
From Prod,Cat, Jour, Vente
Where (Vente.id_p= prod.id_p) AND (prod.id_c=Cat.id_c) AND (Vente.id_j=Jour.id_j)
Group by Cat.Nom, Cat.Description, Jour.jour

Cette requête correspond au cuboïde numéro 2 (dans le tableau 3.2) ; nous remplissons d'abord les attributs des deux tables Jour et Catégorie dans la clause *Select*, puis la clause from reçoit les tables de regroupement jour et catégorie et la table de fait vente. Ensuite, la clause *Where* qui contient trois jointures : deux entre la table de fait vente et les deux tables Jour, Catégorie et une entre la table catégorie et la table produit qui a précédé, et pour obtenir cette table nous récupérons le DDT (drill-down table) de la table catégorie et effectuons une auto-jointure dans la même table dans la colonne '#' pour trouver le nom de la table précédentes à partir de la colonne NT. Enfin, nous remplissons la clause *Group By*, par le nom et la description de catégorie, et le jour.

#	NT	NLTN	DDT	Key
1	Prod	Produit		p_id
2	Cat	Catégorie	1	c_id
3	Jour	Jour		j_id
4	Mois	Mois	3	m_id
5	Trim	Trimestre	4	t_id
6	Quad	Quadrimestre	4	q_id
7	Semes	semestre	5	semes_id
8	Ann	Année	7	a_id
9	Sem	semaine	3	sem_id

TABLE 3.5 – Exemple de la table "Dim_table"

- # : L' Id de la table de de dimension.
- NT : Le nom de table.
- NLTN : Le nom de table en LN.
- DDT (Drill Down Table) : L'Id de la table précédente.
- Key : Le nom de la clé de la table.

3.5 L'algorithme de génération

Remarque

Les deux slashes (//) dans l'algorithme représentent la concaténation entre deux chaînes de caractères.

Algorithme 1 : Génération des requêtes en langage naturel

Entrées :

T_Query_ID /* *Matrice des ordres des tables dans les dimensions* */

Exemple :Ligne01 :1,1

T_TQuery_ID /* *Matrice des noms des tables dans les dimensions* */

Exemple :Ligne01 :jour,prod

T_Query_ID_Table_NL /**Nom des tables en LN de la requête en cours* */

T_Query_ID_Orders /**Ordre des tables de la requête dans le schéma multidimensionnel* */

Dim_Table /* *Table contenant : # : l'ID de table de dimension*
TN :le nom de table
NLTN :le nom de table en LN
DTD :ID de table précédent (Drill-Down)
Key :nom de clé de table */

Clause_Select chaine de caractère /**Contenaire de la clause SELECT* */

Clause_Where chaine de caractère /**Contenaire de la clause WHERE* */

Clause_FROM chaine de caractère ← 'FROM'//cube.fact_table /* *Contenaire de la clause FROM* */

Exemple : 'FROM Ventes'

Clause_GB chaine de caractère /* *Contenaire de la clause GROUP BY* */

#_T_Courante entier /* *ID de la table courante dans Dim_Tables* */

Exemple : le ID de la table Cat dans Dim_tables est 2

#_T_Precedente entier /* *ID de la table précédente de la table courante dans Dim_Tables* */

Exemple : le ID de la table précédente de Cat dans Dim_tables est 1 (table prod)

Sorties :

Querie /* *Table des requêtes* */

1 **Begin**

2 - Choix de la mesure (M_i)

3 - Choix de la fonction d'agrégation (F_{ij})

4 *NL_Query* chaine de caractère ← 'The '//F_{ij}//' of '//M_i//' by '

```
4
5   1 : On sélectionne D_1   /* Première dimension */
6   pour u_1 ← 1 à NND_1 + 1 faire
7       /* NND_1 :nombre de niveaux dans la dimension 1 */
8       2 : On sélectionne D_2
9       pour u_2 ← 1 à NND_2 + 1 faire
10          .
11          .
12          .
13          ND : On sélectionneD_ND   /* ND :le nombre de dimensions dans le schéma*/
14          pour u_ND ← 1 à NND_ND + 1 faire
15              T_Query_ID_orders ← {u_1, u_2, ...., u_ND}
16              #_query_id : entier ← Research(T_Query_ID_orders, T_Query_ID)
17                /* Chercher dans la matrice T_Query_ID la ligne qui correspond à la table
                  T_Query_ID_orders */
18              T_Query_ID_Table_NL ← T_TQuery_ID[#_query_id]
19                /*Récupérer la ligne #_query_id dans la matrice T_TQuery_ID */
20              Arrange(T_Query_ID_orders)   /*Trier dans l'ordre décroissant la table
                  T_Query_ID_orders*/
21              Arrange(T_TQuery_ID_Table_NL)   /*Trier la table
                  T_Query_ID_Table_NL dans le même ordre que T_Query_ID_orders */
22              pour i ← 1 à ND faire
23                  si T_Query_ID_orders[i] ≠ 0 alors
24                      NL_Query ← NL_Query//' and '//T_TQuery_ID_Table_NL[i]
25                      #_T_Courante ← Dim_Table.# où
                          T_Query_ID_Table_NL[i] = Dim_Table.NLTN
26                            Exemple :#_T_Courante ← 2(le # de Cat dans
                          Dim_Tables)
27                      Clause_Select ← Clause_Select//','//liste des attributs avec
                          Attribute.key = #_T_Courante
28                            Exemple :Clause_Select ← cat_name(remarque :on peut
                      omettre les ID ou on peut les ajouter)
```

```
23
24
25
26
27
28
29      tant que Dim_Table[#_T_Courante].DDT ≠ NULL faire
30          #_T_Precedente ← Dim_Table[#_T_Courante].DDT
31          Clause_FROM ←
             Clause_FROM//','//Dim_Table[#_T_Courante].TN
32              Exemple :Clause_FROM ← sales,cat
33          Clause_Where ← Clause_Where//'AND'//
             Dim_Table[#_T_Courante].TN//'.'//Dim_Table[#_T_Courante].key
             //'='//Dim_Table[#_T_Precedente].TN//'.'//
             Dim_Table[#_T_Courante].key
34              Exemple :Clause_Where ← AND cat.pid=prod.pid
35          #_T_Courante ← #_T_Precedente
36            /*changement de pointeurs, passage au niveau inférieur dans la
             hierarchie */
37      si Dim_Table[#_T_Courante].DDT ≠ NULL alors
38          Clause_FROM ←
             Clause_FROM//','//Dim_Table[#_T_Courante].TN
39      Clause_Where ← Clause_Where//'AND'//
         Dim_Table[#_T_Courante].TN//'.'//Dim_Table[#_T_Courante].key
         //'='//cube.fact_table_name//'.'//Dim_Table[#_T_Courante].key
40           Exemple :Clause_Where ← AND cat.pid=prod.pid AND
         prod.pid=sales.pid
41  sinon
42      trim(NL_Query,'by')   /*supprimer le 'by' qui existe dans
         NL_Query*/
43  trim(NL_Query,' AND')   /*supprimer le 'AND' qui existe dans la fin de
     NL_Query*/
44  Querie[#_query_id].NL_query ← NL_Query   /*mettre la requête dans la
     table Querie*/
45
```

```
42
43
44
45
46      si (Clause_Where ≠'')   /*n'est pas une chaine vide*/ alors
47          Clause_Where ← trim(Clause_Where,' AND')   /*supprimer le AND
              du début*/
48          Clause_Where ← 'WHERE '//Clause_Where     /*ajouter le mot
              WHERE au début*/
49          Clause_GB ← 'GROUP BY '//Clause_Select    /*on regroupe par les
              même attributs que la clause select */
50              Exemple :GROUP BY cat_name,month_name,...
51      Clause_Select ← 'SELECT '//F_ij//'('//M_i//')'//Clause_Select
52          Exemple :SELECT max(sale_amount), cat_name,
            month_name,...
53      SQL_Query ← Clause_Select//' '//Clause_FROM//' '//Clause_Where//'
            '//Clause_GB     /*construire la requête SQL globale*/
54      Querie[#_query_id].SQL_query ← SQL_Query    /*mettre la requête dans la
            table Querie*/
```

3.6 Conclusion

Dans ce chapitre, nous avons présenté le principe de la génération des requêtes en langage naturel ainsi qu'en SQL. Nous avons présenté l'algorithme dans lequel nous l'avons écrit, dont l'objectif principal est de générer des requêtes en LN à l'aide d'un modèle générique d'ED. Nous avons d'abord présenté un petit rappel du cube des ventes à titre d'exemple. Nous nous sommes ensuite tournés vers la génération de requêtes en LN et la génération de requêtes en SQL. Enfin, nous présentons le pseudo-algorithme.

CHAPITRE 4

Implémentation et réalisation

4.1 Introduction

Dans ce chapitre, nous présentons la partie pratique qui constitue la dernière étape de notre travail et illustrons une partie de ce que nous avons fait au cours des chapitres précédents. Nous allons essayer de montrer les principaux éléments de notre application, en commençant par les différents outils permettant de la réaliser. Par la suite, nous présentons les structures choisies pour implémenter les méta données de l'ED présentées dans le modèle XML du chapitre précédent, ainsi que les autres structures de base de données nécessaires. Enfin, nous montrons quelques captures d'écran de notre application.

4.2 Environnement de développement

4.2.1 Environnement matériel

Nous avons utilisé pour développer notre application une machine *AMD E1-2500 APU with Radeon(TM) HD Graphics 1.40 GHz* ayant les caractéristiques suivant :

- RAM : 4GO.
- Disque dur : 465 GO.
- Système d'exploitation : Windows 7, 32 bit.

4.2.2 Environnement logiciel

4.2.2.1 NetBeans

L'EDI NetBeans est un environnement de développement et un outil pour les programmeurs pour écrire, compiler, déboguer et déployer des programmes. Il est écrit en Java - mais peut supporter n'importe quel langage de programmation (Java, C / C ++, XML et HTML à

PHP...), nous avons utilisé le langage java pour développer notre application.
L'EDI NetBeans est un produit gratuit, sans aucune restriction quant à son usage et peut être installé sur tous les systèmes d'exploitation prenant en charge Java, comme Windows, Linux, Mac OSX et BSD. Écrire une fois, exécuter n'importe où s'applique également à NetBeans.[21]

4.2.2.2 Oracle Database Express Edition

Database 11g Express Edition (Oracle Database XE) est une base de données d'entrée de gamme, compacte, basée sur la base de code Oracle Database 11g Release 2. Le développement, le déploiement et la distribution sont gratuits, rapides à télécharger ; et simple à administrer. Oracle peut être installé sur une machine hôte de n'importe quelle taille avec un nombre quelconque de processeurs. [22]

4.3 Présentation de l'application

Notre application est développée pour la génération des requêtes en langage naturel en se basant sur la spécification d'un entrepôt de données, puis leur stockage dans une BDD Oracle pour les exploiter ultérieurement.
Nous pouvons dire que notre application est générique et n'est pas dédiée à une entreprise spécifique. Elle est divisée en deux parties :

- La première partie est un processus de tâches successives qui commence par la spécification d'un ou de plusieurs cubes et se termine par la génération de requêtes en langage naturel (LN).
- La deuxième partie est basée sur l'affichage des requêtes en LN déjà générées et stockées dans la base de données, ce qui permet aux décideurs d'analyser les données à travers leur exécution.

4.3.1 Fenêtre de base

C'est la première fenêtre à apparaître au lancement de l'application, elle présente une description de l'application qui dirige l'utilisateur pour choisir entre deux options :

1. La première option permet la création de cubes, la génération de leurs requêtes en LN et l'analyse des données de ces cubes.
2. Le deuxième permet seulement l'analyse de cubes déjà existants.

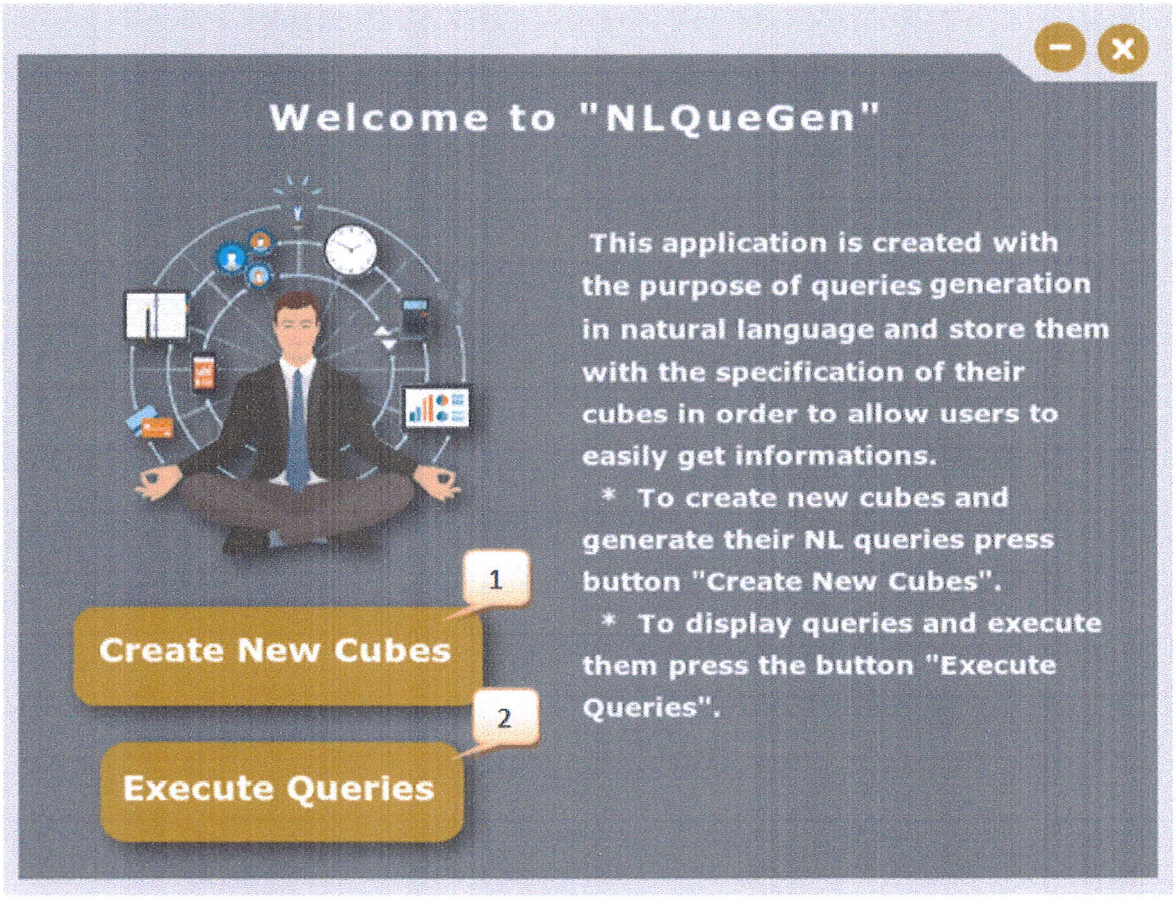

FIGURE 4.1 – La fenêtre de base de l'application.

4.3.2 Création de base de données

Dans la figure suivante (4.2), nous présentons le modèle de BD qui est utilisé pour stocker la description d'ED spécifique ainsi que les requêtes en LN et les requêtes SQL correspondantes. Les tables de la BD sont :

- La table Cube : permet de stocker les cubes avec les noms de leur tables de fait.
- La table Dimension : permet de stocker les dimensions.
- La table Fact_Dim : c'est la table qui relie entre la table Cube et la table Dimension.
- La table Measure : stocke les noms techniques et les noms en langage naturel des mesures.
- La table Function : permet de stocker les noms techniques et les noms en langage naturel des fonctions d'agrégation.
- La table Fact_measure : c'est la table qui relie les tables Cube, Measure et Function.
- La table Queries : permet de stocker les requêtes en LN ainsi qu'en SQL, elle est reliée avec les tables Cube, Measure et Function.

- La table Dim_table : permet de stocker le nom technique de chaque table de dimension, le nom en LN, sa clé, la clé de la table qui la précède et une clé étrangère qui la relie avec la table dimension.
- La table Attribute : permet de stocker les attributs de chaque table dans l'ED.

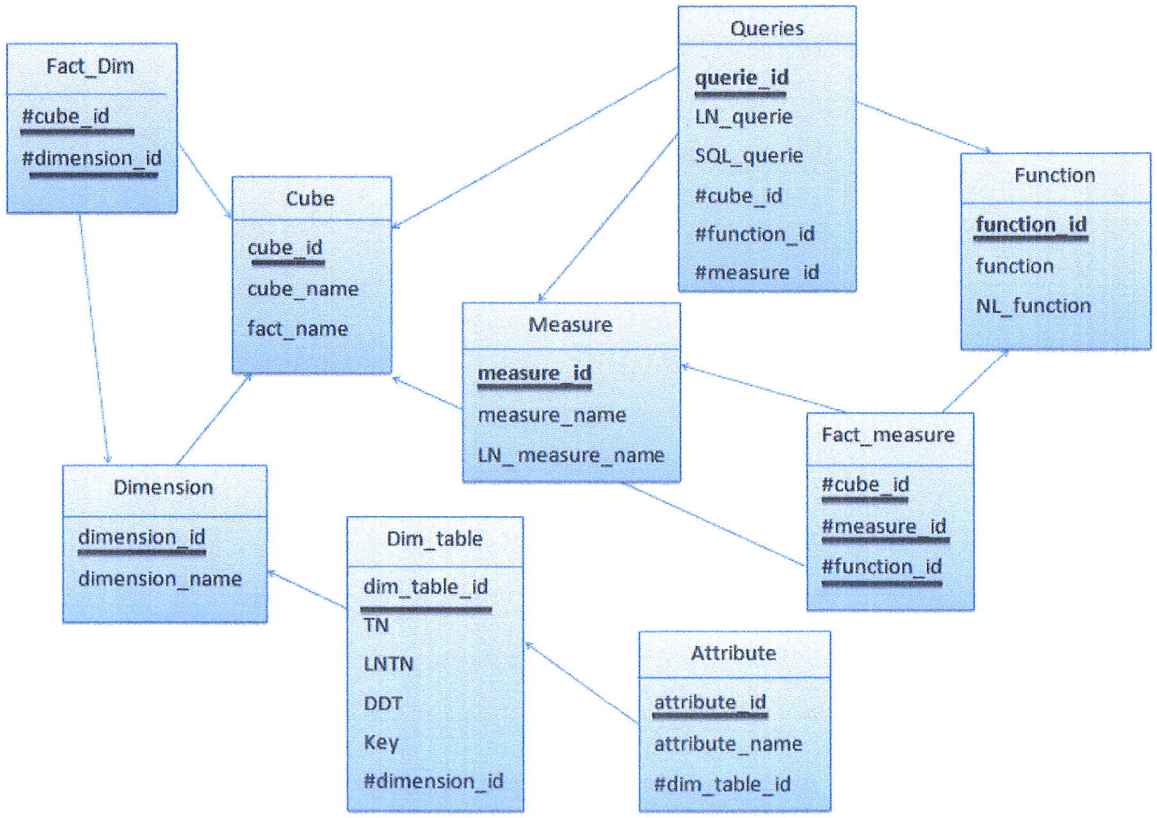

FIGURE 4.2 – Présentation du modèle de la BD.

4.3.3 Les structures java utilisées pour stocker les données

Pour les tables de faits, nous avons créé Arraylist où chaque boîte contenant le nom du fait, l'ensemble des noms des dimensions qui lui sont liées et la liste de ses mesures, qui est également un Arraylist contenant son nom fonctionnel et technique et les noms techniques et fonctionnels des fonctions d'agrégation qui sont associés à cette mesure. La figure suivante illustre cette structure.

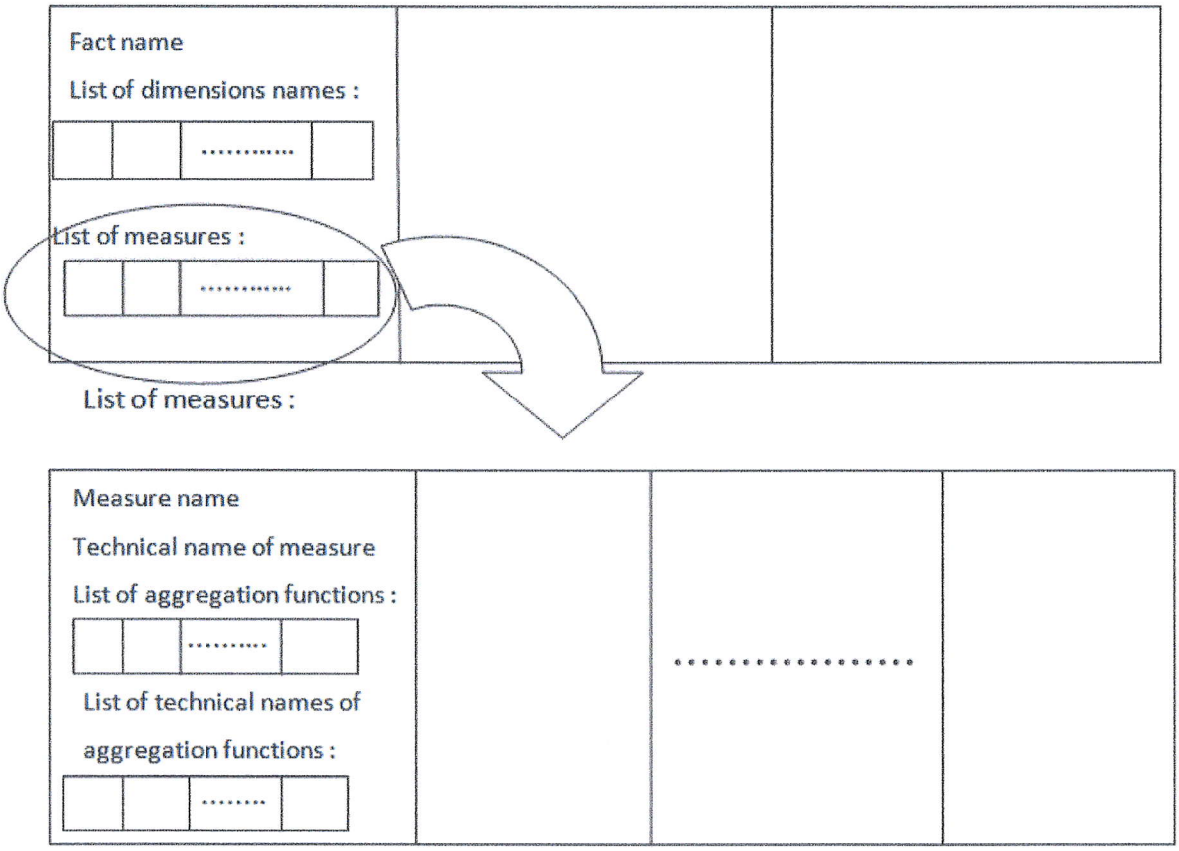

FIGURE 4.3 – La structure utilisée pour stocker les données de la table de faits.

La même structure est utilisée pour stocker la liste des dimensions avec leurs différentes tables, comme indiqué dans la figure ci-dessous.

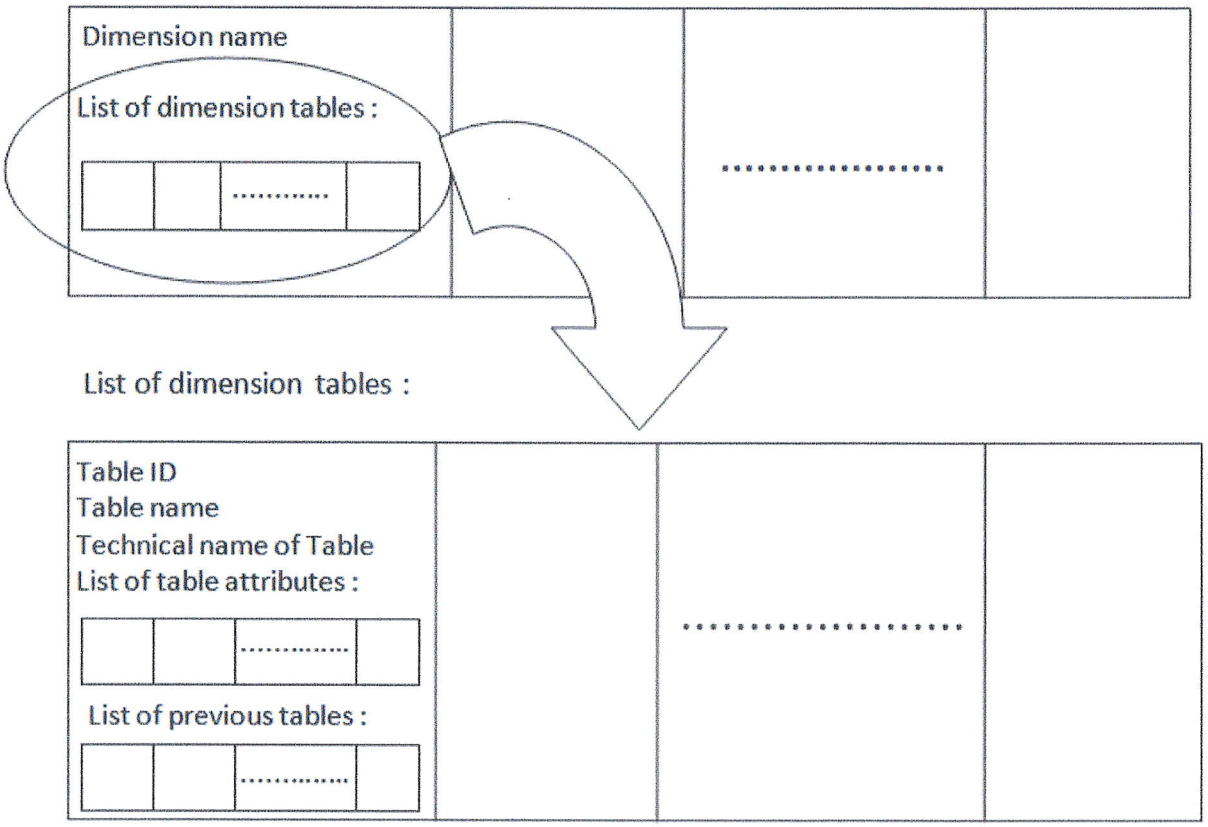

FIGURE 4.4 – La structure utilisée pour stocker les données de dimensions.

Dans ce qui suit, nous détaillons chaque partie de l'application.

4.3.4 Première partie de l'application

Comme expliqué auparavant, la première partie permet d'instancier le modèle générique d'un entrepôt de données et ensuite, de générer les requêtes en langage naturel et en SQL.

4.3.4.1 Fenêtre de Spécification de modèle

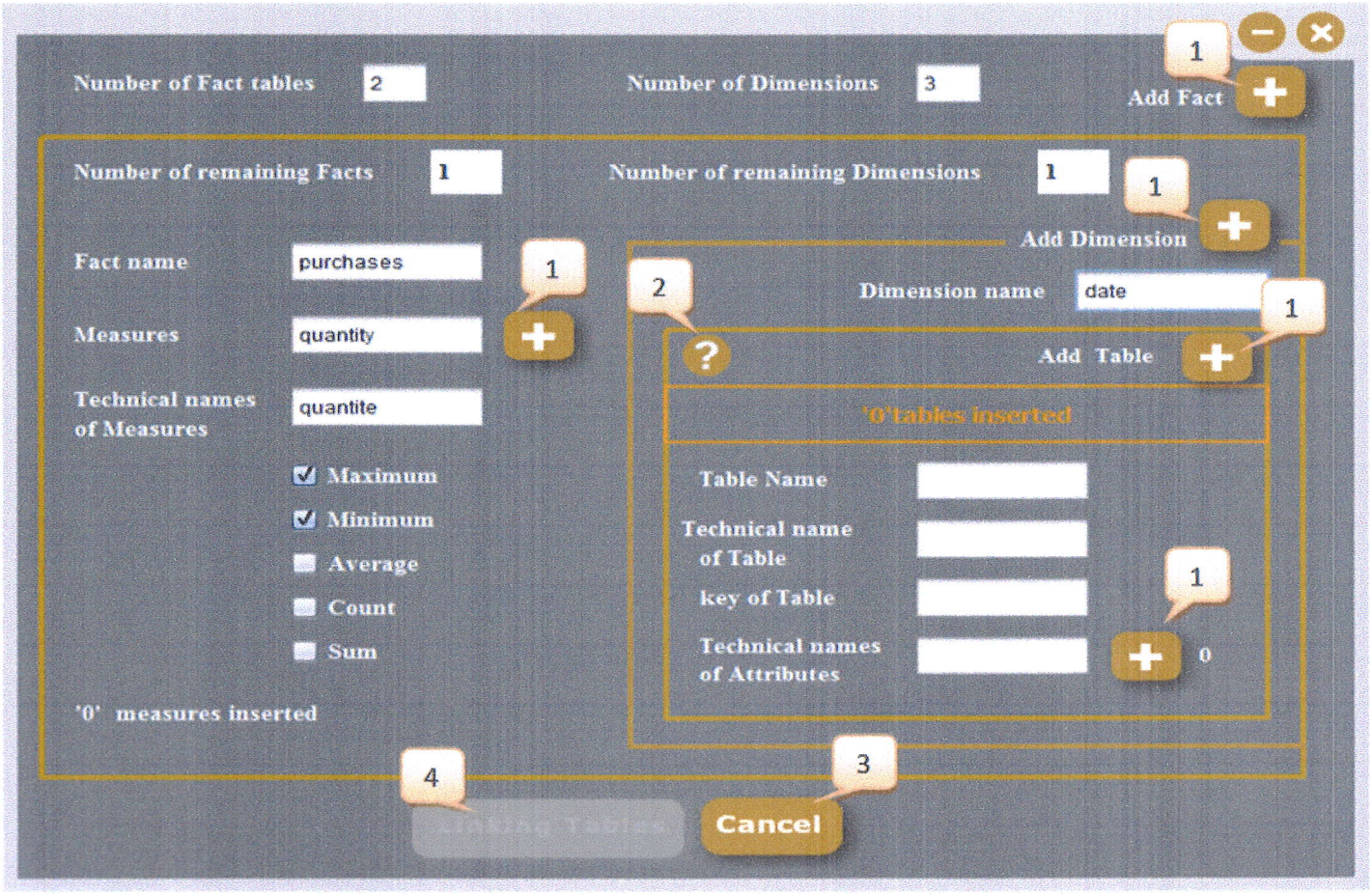

FIGURE 4.5 – Fenêtre principale de l'application.

Le rôle de cette fenêtre est de faire la spécification de l'entrepôt de données, où l'utilisateur commence par déterminer le nombre de cubes, dont il veut générer les requêtes en LN ainsi que le nombre de dimensions. Cette spécification se déroule comme suit :

1. Saisir le nom de la table de fait et ses mesures avec les fonctions d'agrégation.
2. Déterminer les dimensions reliées à ce fait, où pour chaque dimension l'utilisateur doit spécifier le nom de la dimension et ses tables avec pour chaque table : le nom de table en LN, le nom technique, la clé et ses attributs. Les tables doivent être insérées dans leur ordre dans la dimension.
3. Refaire l'étape 1 pour continuer la saisie des autres cubes avec leurs dimensions, et dans le cas où le fait est relié avec une dimension qui est déjà saisie, on n'a pas besoin de la ré-entrer, elle s'ajoute automatiquement au cube.

Rôles des boutons Dans la figure 4.5, les différentes options numérotées, sont :

1. Ajouter et sauvegarder (mesure, table, dimension, fait, attribut).

2. Afficher l'aide.
3. Annuler la spécification.
4. Ouvrir la deuxième fenêtre pour lier les tables.

4.3.4.2 Fenêtre de liaison des tables et de validation de modèle

Dans cette fenêtre, l'utilisateur termine la phase de spécification en reliant les tables entre elles pour chaque dimension (lier les niveaux hiérarchiques) et puis, passe à la phase de génération des requêtes.

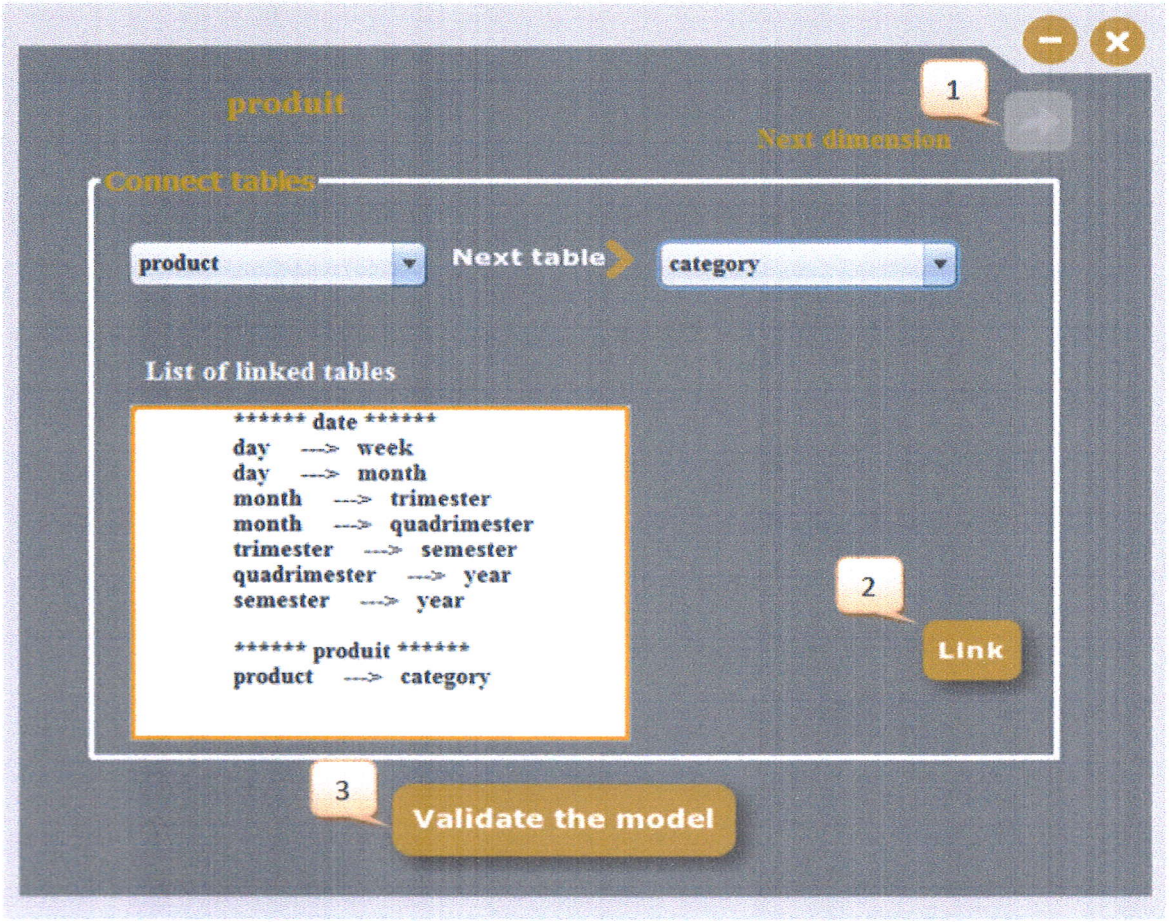

FIGURE 4.6 – La fenêtre de liaison des tables.

Rôles des boutons Dans la figure 4.6, les différentes options numérotées, sont :

1. Passer à la dimension suivante.
2. Relier les deux tables.
3. Valider et sauvegarder le modèle dans la BD.

Lorsque l'utilisateur clique sur le bouton « validate the model », une fenêtre d'information apparaît indiquant que le modèle est validé et sauvegardé avec succès et propose de cliquer sur le bouton « Generate » pour la génération des requêtes en LN.

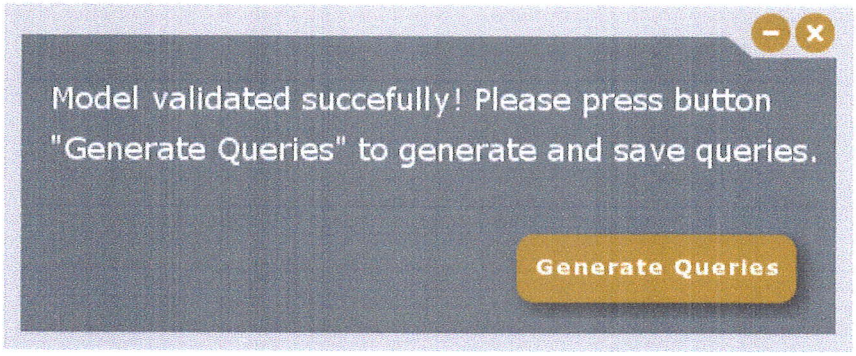

FIGURE 4.7 – La fenêtre de génération des requêtes.

Le clic sur le bouton « Generate Queries » permet de générer les requêtes en LN avec les requêtes SQL et les stocker dans la BD.
Voici, en figure 4.8, un exemple de contenu de la table "Queries" après la génération et la sauvegarde des requêtes dans la BD.

QUERY_ID	NL_QUERY	SQL_QUERY
1	The maximum of quantity by day and product	SELECT max(quantite), jour,descrip_prod,prix FROM sales, jour, produit WHERE jour.id_jour=sales.id
2	The maximum of quantity by category and day	SELECT max(quantite), description,jour FROM sales, categorie, produit, jour WHERE categorie.id_cat
3	The maximum of quantity by day	SELECT max(quantite), jour FROM sales, jour WHERE jour.id_jour=sales.id_jour GROUP BY jour
4	The maximum of quantity by week and product	SELECT max(quantite), semaine,descrip_prod,prix FROM sales, semaine, jour, produit WHERE semain
5	The maximum of quantity by week and category	SELECT max(quantite), semaine,description FROM sales, semaine, jour, categorie, produit WHERE ser
6	The maximum of quantity by week	SELECT max(quantite), semaine FROM sales, semaine, jour WHERE semaine.id_semaine=jour.id_sema
7	The maximum of quantity by month and product	SELECT max(quantite), mois,descrip_prod,prix FROM sales, mois, jour, produit WHERE mois.id_mois=j
8	The maximum of quantity by month and category	SELECT max(quantite), mois,description FROM sales, mois, jour, categorie, produit WHERE mois.id_mc

FIGURE 4.8 – Exemple de table « Queries » de BD.

4.3.5 Deuxième partie de l'application

4.3.5.1 Fenêtre d'affichage des requêtes en LN

C'est la fenêtre qui prouve qu'on a atteint l'objectif de ce travail à travers l'affichage des requêtes en langage naturel de tous les cubes stockées dans la base de données.

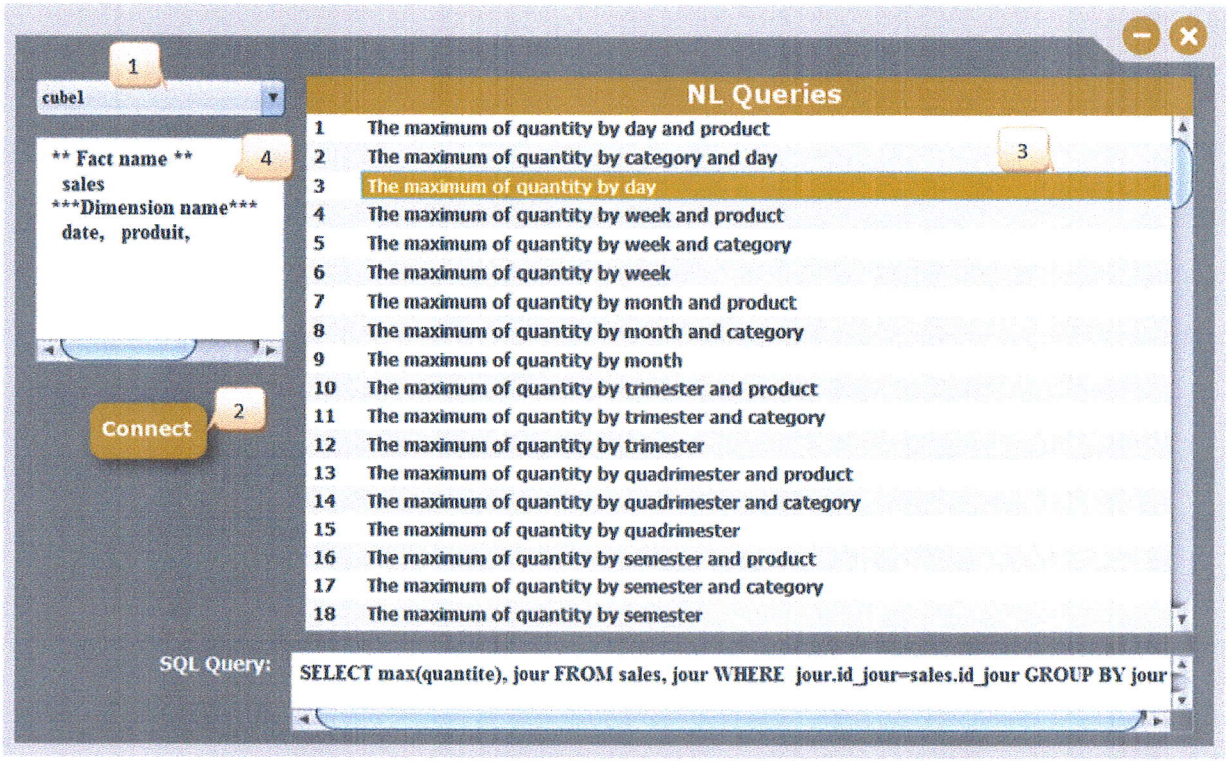

FIGURE 4.9 – La fenêtre d'affichage des requêtes.

Dans la figure 4.9, les différentes options numérotées, sont

1. La modification du cube sélectionné permet le changement d'affichage du tableau "NL Queries".
2. Permet la connexion au cube sélectionné.
3. Permet l'affichage de requête en SQL et l'exécuter dans le cas où le cube est connecté.
4. Les détails de cube sélectionné.

Pour l'exécution des requêtes, il faut se connecter au cube ciblé à travers la fenêtre suivante :

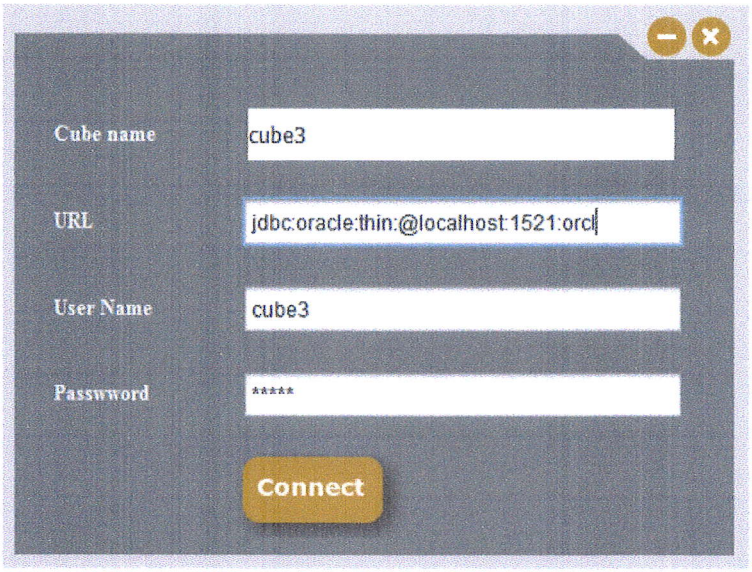

FIGURE 4.10 – Fenêtre de connexion au cube.

4.3.5.2 Fenêtre d'affichage des résultats des requêtes

Cette fenêtre permet d'afficher les résultats de la requête exécutée, ce qui permet à l'analyste d'analyser les données.

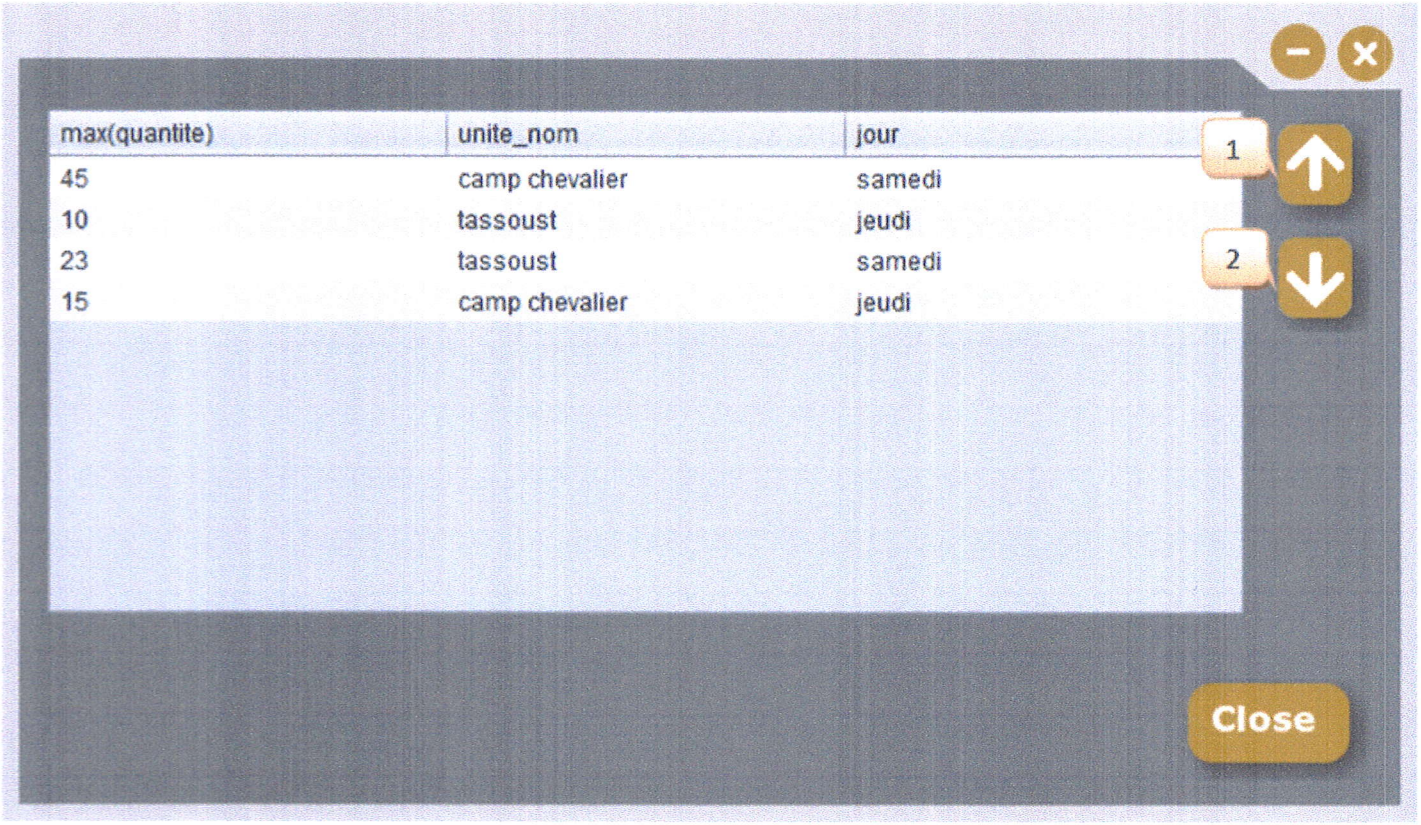

FIGURE 4.11 – Fenêtre d'affichage des résultats.

Dans la figure 4.11, les différentes options numérotées, sont :

1. Trier les données dans l'ordre croissant.
2. Trier les données dans l'ordre décroissant.

4.4 Conclusion

Dans ce chapitre, nous avons présenté notre "application "dont l'objectif principal est de générer des requêtes en LN à partir des cubes de données. Nous avons commencé par présenter l'environnement de création de l'application, ainsi que de nos choix en matière de stockage de données. Enfin, nous avons montré le fonctionnement de l'application en termes de :

- Instanciation du modèle d'entrepôt
- Génération des requêtes en langage naturel
- Génération des requêtes en SQL
- Exécution des requêtes

Conclusion générale

Dans ce mémoire, nous avons présenté un travail de génération d'une charge de requêtes d'exploration et de navigation en LN en l'associant aux requêtes SQL correspondantes à partir des données décisionnelles d'un entrepôt de données et ce afin d'aider un décideur à prendre sa décision.

Les requêtes d'analyse OLAP sont difficiles à écrire pour un non technicien, notamment dans un contexte décisionnel et dans des situations qui nécessitent l'accès aux données sans passer par des applications.

Pour ce faire, nous avons proposé un modèle générique des méta-données d'un ED qui décrit un schéma de flocon de neige, avec un exemple concret de modèle en guise d'illustration.

Par la suite, nous avons présenté un algorithme de génération de requêtes en LN dédié à des entrepôts ou cubes de données que nous avons spécifiés à partir du modèle générique. Chaque requête en LN a sa requête SQL correspondante.

Enfin, nous avons présenté notre application, en commençant par l'environnement dans lequel nous l'avons développée, puis quelques explications de l'application avec des captures d'écran.

Notre application a montré la faisabilité de ce que nous avons fait dans notre travail. Elle peut être exécutée dans le domaine BI où un utilisateur peut spécifier des cubes et générer ses requêtes en LN et effectuer une analyse OLAP rapide sur les données, ou dans le domaine académique qui est un bon moyen permettant aux étudiants d'apprendre le langage SQL en affichant la requête SQL à partir de la requête LN et en l'exécutant pour afficher son résultat.

Finalement, la logique de notre application peut s'appliquer dans un contexte mobile afin d'être plus efficace dans des situations où un directeur d'entreprise est en dehors de son entreprise et il ne peut pas utiliser son outil OLAP qui est habituellement utilisé sur un

ordinateur. Donc, plus tard on peut transformer notre application en une application mobile.

Un autre volet serait d'appliquer ce travail à des schémas en étoile avec l'explicitation des hiérarchies.

Bibliographie

[1] Jean-François Desnos. Projet" entrepôt de données. 2002. http://imss-www.upmf-grenoble.fr/prevert/SpecialiteIHS/ED/IntroductionED.pdf.

[2] G Satyanarayana Reddy, Rallabandi Srinivasu, M Poorna Rao, and Srikanth Reddy Rikkula. Data warehousing, data mining, olap and oltp technologies are essential elements to support decision-making process in industries. *International Journal on Computer Science and Engineering*, 2(9) :2865–2873, 2010.

[3] Bastien L. Data warehouse(entrepôts de données)définition :qu'est-ce que c'est, 14 Février 2018. https://www.lebigdata.fr/data-warehouse-entrepot-donnees-definition.

[4] Ralph Kimball and Margy Ross. *Entrepôts de données : guide pratique de modélisation dimensionnelle.* Vuibert informatique, 2003.

[5] CKH Lee, King Lun Choy, George TS Ho, Kwai-Sang Chin, KMY Law, and Ying Kei Tse. A hybrid olap-association rule mining based quality management system for extracting defect patterns in the garment industry. *Expert Systems with Applications*, 40(7) :2435–2446, 2013. https://www.sciencedirect.com/science/article/pii/S0957417412011815.

[6] Olivier Teste. *Modélisation et manipulation des systèmes OLAP : de l'intégration des documents à l'usager.* PhD thesis, Université Paul Sabatier-Toulouse III, 2009. https://tel.archives-ouvertes.fr/tel-01456620/document.

[7] Michael Schrader and Dan Vlamis. *Oracle Essbase & Oracle OLAP.* Peter Gbolagade Akintunde, 2009. https://books.google.dz/books?id=qVSweqz71CwC&pg=PA10&dq=MOLAP&hl=fr&sa=X&ved=0ahUKEwiUgeXuioDhAhW-BWMBHf0vCPUQ6AEIcTAJ#v=onepage&q=MOLAP&f=false.

[8] software advice. Olap tools, consulté le 14 mars 2019. https://www.lebigdata.fr/data-warehouse-entrepot-donnees-definition.

[9] Xiaojun Xu, Chang Liu, and Dawn Song. Sqlnet : Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv :1711.04436*, 2017. http://arxiv.org/abs/1711.04436.

[10] Archit Gupta Ikshu Bhalla. Genrating sql queries from natural language. *université de Stanford*, 2017.

[11] Ruichu Cai, Boyan Xu, Xiaoyan Yang, Zhenjie Zhang, Zijian Li, and Zhihao Liang. An encoder-decoder framework translating natural language to database queries. *arXiv preprint arXiv :1711.06061*, 2017. https://www.ijcai.org/proceedings/2018/0553.pdf.

[12] Vadim Sheinin, Elahe Khorasani, Hangu Yeo, Kun Xu, Ngoc Phuoc An Vo, and Octavian Popescu. Quest : A natural language interface to relational databases. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018. https://aclweb.org/anthology/L18-1469.

[13] William Jordan Holton. *Translating database queries to English for enhancing database education.* PhD thesis, Georgia Institute of Technology, 2015. https://smartech.gatech.edu/bitstream/handle/1853/54339/HOLTON-THESIS-2015.pdf?sequence=1&isAllowed=y.

[14] Nicolas Kuchmann-Beauger and Marie-Aude Aufaure. Natural language interfaces for datawarehouses. In *8èmes Journées francophones sur les Entrepôts de Données et l'Analyse en ligne*, page unknown, 2012. https://hal.archives-ouvertes.fr/hal-00704293.

[15] M Asif Naeem, Saif Ullah, and Imran Sarwar Bajwa. Interacting with data warehouse by using a natural language interface. In *International Conference on Application of Natural Language to Information Systems*, pages 372–377. Springer, 2012. https://www.researchgate.net/publication/230680103_Interacting_with_Data_Warehouse_by_Using_a_Natural_Language_Interface.

[16] G. Chagnon and F. Nolot. *XML : Synthèse de cours & exercices corrigés.* Collection Synthex : Informatique. Pearson Education, 2007. https://books.google.dz/books?id=Nf2vwefUSOwC&pg=PA1&dq=historique+XML&hl=fr&sa=X&ved=0ahUKEwiGos_Zt5biAhWaAWMBHSUEBmAQ6AEIMjAC#v=onepage&q=historique%20XML&f=false.

[17] C. M. Sperberg-McQueen Eve Maler François Yergeau John Cowan Tim Bray, Jean Paoli. Le langage de balisage extensible (xml) 1.1. 4 février 2004. http://www.yoyodesign.org/doc/w3c/xml11/.

[18] V. Mesguich. *Rechercher l'information stratégique sur le web : Sourcing, veille et analyse à l'heure de la révolution numériqu.* INFORMATION ET STRATÉGIE. De Boeck supérieur, 2018. https://books.google.dz/books?id=pXFhDwAAQBAJ.

[19] C. Vincent. *XML et XSL : les feuilles de styles XML.* Ressources informatiques. Editions ENI, 2003.

[20] Dico du net. Xsd-définition de schéma xml- xml schema definition, consulté le 10 avril 2019. http://www.dicodunet.com/definitions/normes/xsd.htm.

[21] Netbeans. Qu'est ce que netbeans, consulté le 01 juin 2019. https://netbeans.org/index_fr.html.

[22] Oracle. Oracle database express edition, consulté le 01 juin 2019. https://www.oracle.com/technetwork/products/expressedition/overview/index-100989.html.

RESUME

L'idée de ce travail est basée sur la préparation d'un ensemble de requêtes analytiques en langue naturelle permettant aux décideurs d'analyser des données décisionnelles organisées dans un entrepôt de données, au lieu d'écrire des requêtes OLAP considéré difficiles pour un non-technicien. Le travail est centré autour de la création d'un méta-modèle d'entrepôts de données adaptés et adoptés pour la génération de ces requêtes décisionnelles, puis proposer un algorithme qui génère un ensemble de requêtes en LN et SQL, et développer enfin une application permettant aux décideurs d'analyser des données à partir d'ensemble de requêtes en langue naturelle.

Mots clés : Entrepôt de données, Langue naturelle, OLAP, Génération des requêtes.

ABSTRACT

The idea of this work is based on the preparation of a set of natural language analytic queries to allow decision makers to analyze decisional data organized in a data warehouse, instead of writing OLAP queries which can turn to be difficult for a non technician. The work is centered around creating a meta-model of data warehouses adapted and adopted for the generation of these decisional queries, then propose an algorithm that generates a set of queries in LN and SQL, and finally develop an application allowing decision makers to analyze data from natural language query sets.

Key words : Data Warehouse, Natural language, OLAP, Queries generation.

مُلخص

تستند فكرة هذا العمل على إعداد مجموعة من الاستعلامات التحليلية باللّغة الطبيعية التي تسمح لصانعي القرار بتحليل البيانات المنظمة في مستودع البيانات، بدلاً من كتابة استعلامات OLAP التي تُعتبر صعبة بالنسبة لغير التقنيّين. يتمحور هذا العمل حول إنشاء نموذج تعريف لمستودعات البيانات المكيفة والمعتمدة لتوليد طلبات القرارات هذه ، ثم اقتراح خوارزمية تُنشئ مجموعة من الاستعلامات باللّغة الطّبيعية و بـ SQL ، وأخيراً تطوير تطبيق يسمح لصانعي القرار بتحليل البيانات من خلال مجموعة من الاستعلامات باللّغة الطبيعية.