Académie de Montpellier
Université Montpellier II
Sciences et Techniques du Languedoc

MÉMOIRE DE STAGE DE MASTER M2

effectué à Montréal

Spécialité : **Professionelle et Recherche unifiée en Informatique**

BiinTheCloud

par **Olivier BENDAVID**

Date de soutenance : **18/06/10**

Sous la direction de **Alain APRIL**

Résumé

Au vu de l'importance des données accumulées par les entreprises et de l'augmentation de leurs besoins en analyse. Nous voyons apparaître de nombreuses solutions de business intelligence pour y répondre. Ces solutions se basent pour la plupart sur des systèmes d'informations orientés relationnel, système dont les limites commencent à être éprouvée. Les temps d'accès aux ressources deviennent trop long et certaines analyses sont trop complexes pour être réalisée par ces SGBD. C'est dans ce contexte que "Google" a mis au point un nouveau type de SGBD dédié à administrer de grandes quantités de données (BigTable).

Par ailleurs la distribution de services commencent à prendre un nouveau visage avec le développement du concept de cloud computing. Concept ayant pour finalité de dissocier les ressources permettant de fournir un service du consommateur. Nous avons travaillé sur la conception d'une solution de Business Intelligence basée sur le concept de cloud computing. Ainsi nous avons réalisé un prototype dont le système d'information est basé sur un SGBD NoSQL (HBase).

Abstract

Given the importance of stored data by companies and their increasing needs in analysis. We can see many business intelligence solutions coming out to address them. These solutions are based mostly on information systems oriented relational, system whose boundaries are beginning to be felt. The time for accessing resources become too long and some analysis are too complex to be performed by those DBMS. It is in this context that "Google" has developed a new kind of database for the purpose of administer large amounts of data (BigTable).

Moreover, the distribution services are beginning to take a new look with the emerging concept of cloud computing. Concept whose purpose is to separate the resources and consumers to provide service. We have worked on the conception of a business intelligence solution based on the concept of cloud computing. So we made a prototype which information system is based on a NoSQL DBMS (HBase).

Table des matières

1 Introduction

L'utilisation de SGBD relationnel est très majoritairement répandu pour la gestion des données. Si son usage avait été initialement conçu pour organiser, stocker et exploiter de manière optimale des données. Il n'avait pas été pensé pour les grandes quantités de données accumulées actuellement. L'exemple de youtube en est très révélateur, chaque minute l'équivalent de vingt-quatre heures de vidéos est ajouté, c'est dans le contexte d'une incapacité à gérer leurs données qu'ils se sont fait acheter par Google en 2006. Google ayant développé un nouveau type de SGBD dédié a de l'entreposage de grandes quantités de données dont ils se servent déjà pour un grand nombre de leurs produits tels que Google Reader, Google Maps, Google Earth, Google Code hosting, YouTube, Gmail ... D'autres sociétés telle qu' Amazon avec Dynamo se tourne vers ces nouveaux SGBD. Le stockage de très grandes quantités de données pose donc certains problèmes, ces difficultés pouvant aller du simple ralentissement du système, jusqu'à la panne complète. Ces nouveaux types de SGBD permettent de répondre d'une part a ce problème et de par leur conception peuvent servir de système d'information pour une architecture cloud computing.

Par ailleurs l'utilisation de logiciel se reposant sur des SGBD relationnel est en constante évolution. La plupart des entreprises se reposent sur ces logiciels afin de gérer leurs différents processus. Il existe des outils de gestion du personnel, de comptabilité, d'organisation et gestion de projet, de gestion de documents ... L'ensemble des informations recueillies par ces logiciels constituent une base solide permettant de représenter les différents processus d'une ou plusieurs entreprises. C'est sur les informations récupérées à partir de ces outils que la business intelligence opère, l'ensemble de ces informations permettent d'obtenir une vue d'ensemble du fonctionnement d'une entreprise. Une solution de business intelligence se compose d'un ensemble d'outils permettant de collecter, stocker et analyser ces données. Le système d'information d'une solution BI se repose sur un entrepôt de données afin de collecter les données issus des différentes sources. Le but est de fournir une vue d'ensemble des activités d'une entreprise ainsi que des mécanismes d'aide à la décision.

D'autres part avec le développement du concept de cloud computing, nous voyons de nombreux fournisseurs de solutions BI s'y diriger posant sur la table de nouvelles problématiques et manière de concevoir ces logiciels.

L'entreprise Intégratik m'ayant accueilli en stage est spécialisée dans les

ERP, elle se caractérise par l'utilisation d'un framework afin de développer des solutions spécifiques à leurs commanditaires. En vue d'accroitre ses possibilités elle s'intéresse à la possibilité d'intégrer des services de Business Intelligence à son offre actuelle.

Nous avons réalisé une étude sur la conception d'une solution de Business Intelligence basée sur le concept de cloud computing. Ainsi nous avons réalisé un prototype dont le système d'information est basé sur un SGBD NoSQL.

Notre démarche a été d'étudier ce qu'est la business intelligence, le cloud computing, les solutions technologiques et les concepts nous permettant de concevoir un système BI. Notre proposition s'appuie sur l'utilisation d'un SGBD NOSql (HBase) pour système d'information et la mise en place d'une procédure s'appuyant sur l'ingénierie dirigée par les modèles visant à faciliter la migration d'une base de données orientée relationnel vers une base de données orientée colonne.

Ce document s'organise de la manière suivante :
- Un état de l'art :
  - Sur le cloud computing
  - Sur la business intelligence
  - Sur les SGBD NoSQL
- Les propositions réalisées au cours du stage.
- La description du prototype réalisé

2 État de l'art

2.1 L'informatique en nuage "CLoud Computing"

On peut considérer de manière générale, lorsque l'on parle de cloud computing, qu'il s'agit de rendre accessible et exploitable des données et des applications à travers un réseau. Ce terme désigne à la fois les applications en tant que services sur Internet et le matériel et logiciels qui permettent de fournir ces services [AFG+09].

2.1.1 Les différentes couches de services d'un cloud

Le concept de cloud computing est utilisé pour désigner des services, on distingue trois "catégories" de services fournis :
- Infrastructure as a service : IaaS
- Platform as a service : PaaS
- Software as a service : Saas

La figure 1 propose une représentation générale du cloud computing.

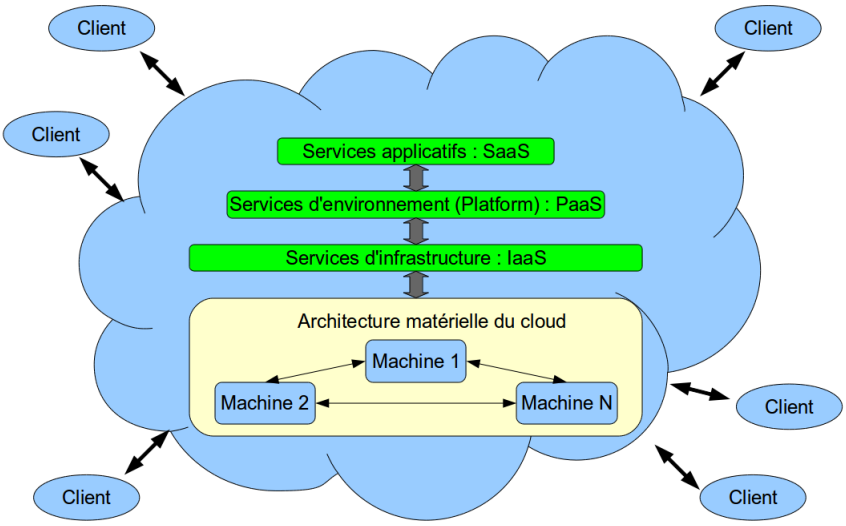

FIGURE 1 – Services Cloud Computing

IaaS

IaaS désigne les services relatif a l'utilisation de l'infrastructure de cloud computing. L'architecture matérielle composant le cloud devient accessible par des logiciels de virtualisation matérielle. Les ressources du cloud sont alors proposés sous forme de service tel que : du stockage et de la virtualisation d'OS. Il s'agit des services les plus proches des ressources matérielles du cloud.

Voici quelques exemples de fournisseurs IaaS :

- Amazon : Web Services Elastic Compute Cloud (EC2) et Secure Storage Service (S3).
- CloudSigma
- RackSpace :Cloud Server et Cloud Files

PaaS

PaaS désigne les services visant à proposer un environnement complet permettant de développer et déployer des applications. Cette solution se compose des services de l'IaaS, infrastructure, solution de stockage et par-dessus lequel s'ajoute un système de gestion de base de données, un environnement de développement d'application et de distribution.

Certaines offres PaaS ont un langage de programmation spécifique. Par exemple :

- Google AppEngine est une offre PaaS où les développeurs peuvent réaliser leurs programme en Python ou Java.
- EngineYard avec Ruby on Rails.
- force.com de Salesforce.com langage propriétaire.
- Coghead (SAP) langage propriétaire.

SaaS

SaaS désigne les services permettant de distribuer des applications, il se repose sur les services présentés précédemment. Certains fournisseurs de SaaS s'exécutent sur un autre fournisseur de cloud PaaS ou offres de services IaaS.

Voici quelques exemples de fournisseurs connus SaaS :

- Oracle CRM On Demand
- Salesforce.com
- NetSuite

2.1.2 Les différents niveaux de déploiement

Le cloud computing peut être exploité à différents niveaux de déploiement suivant le réseau dans lequel les services sont disponible. On parle alors de :

- private cloud : quand les services sont accessible via un réseau privé.
- hybrid cloud : quand certains services sont accessible via un réseau privé et d'autres via internet.
- community cloud : quand les services sont accessible via l'interconnexion de réseaux appartenant a plusieurs organisations.
- public cloud : quand les services sont accessible via internet.

2.1.3 Les principaux avantages

Le cloud computing présente plusieurs avantages dont les principaux sont :

- Le séparation du service et de l'infrastructure nécessaire pour l'exécuter (Virtualisation).
- Nature élastique de l'infrastructure permettant un meilleur contrôle des performances.
- Ressources pouvant être allouée à la demande.

2.2 Business Intelligence

La business intelligence (informatique décisionnelle en français) désigne les moyens, les outils et les méthodes qui permettent de collecter, consolider,

modéliser et restituer les données d'une entreprise en vue d'offrir une aide à la décision et de permettre aux responsables de la stratégie d'entreprise d'avoir une vue d'ensemble de l'activité traitée.

Ce type d'application utilise en règle générale un entrepôt de données (ou datawarehouse en anglais) pour stocker des données provenant de plusieurs sources hétérogènes et fait appel à des traitements par lots pour la collecte de ces informations.

Un système BI est composé de quatre grandes catégories d'outils : l'ETL, data warehousing, data mining et de reporting. La partie sur laquelle notre étude a porté se base sur les deux premiers outils, nous présenterons donc l'outil ETL, data warehousing et la modélisation multidimensionnelle employée pour l'organisation des données dans un entrepôt de données. Nous avons utilisé pour présenter certains points des informations tirées de cours de l'ÉTS (école technique supérieure) et des sites "www.piloter.org" et "wikipedia".

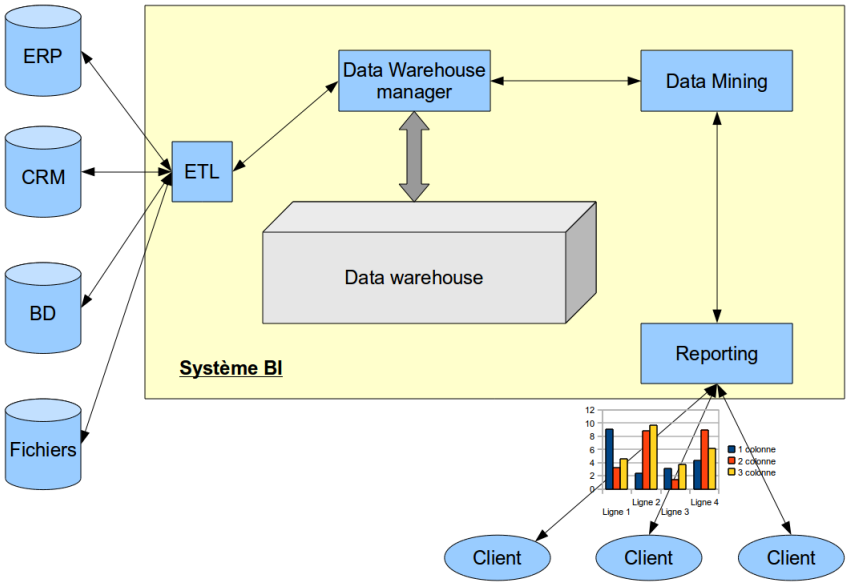

FIGURE 2 – Schéma général Business Intelligence

2.2.1 ETL

L'ETL (extract transform load), assure la connexion du système à une ou plusieurs sources de données, l'extraction, la transformation et le chargement dans l'entrepôt. Il s'agit d'un outil fondamental de l'entrepôt de données, base du système d'information d'une solution BI [KC04]. Les sources de données à

extraire sont hétérogènes tant sur le plan technique que logique. Les données sont stockées dans des systèmes et sous des formats variés. Par ailleurs la plupart des données à collecter doivent être vérifiées afin d'assurer la fiabilité des informations recueillies. Les principales fonctionnalités d'un ETL :

- Extract (Extraire) : Assurer la connexion à la majorité des systèmes de stockage de données afin de pouvoir identifier, sélectionner et récupérer les données. Il doit aussi pouvoir assurer un mécanisme de synchronisation pour la ré-actualisation.
- Transform (Transformer) : Les données à récupérer ne sont pas forcément dans l'état dans lequel elles seront stockées. Elles doivent être vérifiées, re-formatées, nettoyées afin d'éliminer les valeurs incohérentes ainsi que les doublons. Il s'agit donc des opérations permettant de consolider et d'assurer la validité des données.
- Load (Charger) : Stocker les données dans le data warehouse afin qu'elles puissent être utilisable par les autres outils du système BI.

L'ensemble des solutions BI actuelles disposent de ces outils présentant tous des fonctionnalités similaires. Voici quelques exemples des principaux outils ETL utilisés actuellement :

- solution propriétaire
  - IBM Information Server, InfoSphere DataStage
  - SAS Data Integration Studio
  - Oracle Warehouse Builder (OWB)
  - Sap BusinessObjects Data Integration
- solution open-source :
  - Talend Open Studio
  - Pentaho Data Integration (Kettle)
  - Clover ETL

2.2.2 Data Warehouse

Un data warehouse (Entrepôt de données en français) est une base de données regroupant l'ensemble des données fonctionnelles d'une entreprise. Son but est de fournir un ensemble de données servant de référence unique, utilisée pour la prise de décisions dans l'entreprise par le biais de statistiques et de rapports réalisés via des outils de reporting.

D'un point de vue architectural, il existe deux manières de l'appréhender :

- L'architecture "de haut en bas" : selon Bill Inmon, l'entrepôt de données est une base de données au niveau détail, consistant en un référentiel global et centralisé de l'entreprise. En cela, il se distingue du datamart, qui regroupe, agrège et cible fonctionnellement les données.
- L'architecture "de bas en haut" : selon Ralph Kimball, l'entrepôt de

données est constitué peu à peu par les datamarts de l'entreprise, regroupant ainsi différents niveau d'agrégation et d'historisation de données au sein d'une même base.

Un datamart (magasin de données en français) est un sous ensemble des données d'un data warehouse.

2.2.3 Modélisation multidimensionnelle

Le concept OLAP (Online Analytical Processes) a été introduit par E.F Codd, pour rendre des analyses sur des données d'entreprise complexes suivant plusieurs dimensions[CCS93]. Il a établi un ensemble de règles que doit posséder une base de données pour intégrer ce concept.

La modélisation multidimensionnelle a été introduite pour faciliter les traitements OLAP afin de rendre le modèle conceptuel d'organisation des données plus en adéquation avec leur représentation. "The emphasys of OLAP systems is on flexible data grouping and efficient agregation evaluation obtained groups" [LW96].

Il s'agit donc d'un modèle conceptuel de représentation de données. Ce modèle possède plusieurs concepts :

Les concepts : Fait et Dimensions

Le fait modélise le sujet de l'analyse. Un fait est formé de mesures correspondant aux informations de l'activité analysée. Les mesures d'un fait sont numériques et généralement valorisées de manière continue. Les mesures sont numériques pour permettre de résumer un grand nombre d'enregistrements en quelques enregistrements. Les mesures sont valorisées de façon continue, car il est important de ne pas valoriser le fait avec des valeurs nulles.

Une dimension se compose de paramètres correspondant aux informations faisant varier les mesures du fait. Les dimensions servent à enregistrer les valeurs pour lesquelles sont analysées les mesures de l'activité. Une dimension est généralement formée de paramètres (ou attributs) textuels et discrets. Les paramètres textuels sont utilisés pour restreindre la portée des requêtes afin de limiter la taille des réponses.

Lors du processus OLAP, les données sont généralement analysées en partant d'un faible niveau de détail vers des données plus détaillées pour "forer vers le bas". Pour définir ces différents niveaux de détail, chaque dimension est munie d'une (ou plusieurs) hiérarchie(s) des paramètres, on parle de granularité. La hiérarchie sert lors des analyses pour restreindre ou accroître les niveaux de détail de l'analyse.

Le cube

Il s'agit d'une représentation de modélisation multidimensionnelle des données facilitant l'analyse d'une quantité selon différentes dimensions. Cette vision correspond à une structuration des données selon plusieurs axes d'analyses (ou dimensions) pouvant représenter des notions variées telles que le temps, la localisation géographique, le code identifiant des produits. On parle alors de :

- M-OLAP : Multidimensional OLAP
- R-OLAP : Relational OLAP
- H-OLAP : Hybrid OLAP (mix entre MOLAP et ROLAP)
- S-OLAP : Spatial OLAP

Les schémas : Modèles en étoile, en flocon et en constellation

À partir du fait et des dimensions, il est possible d'établir une structure de données simple qui correspond au besoin de la modélisation multidimensionnelle. Cette structure est constituée du fait central et des dimensions. Ce modèle représente visuellement une étoile, on parle de modèle en étoile.

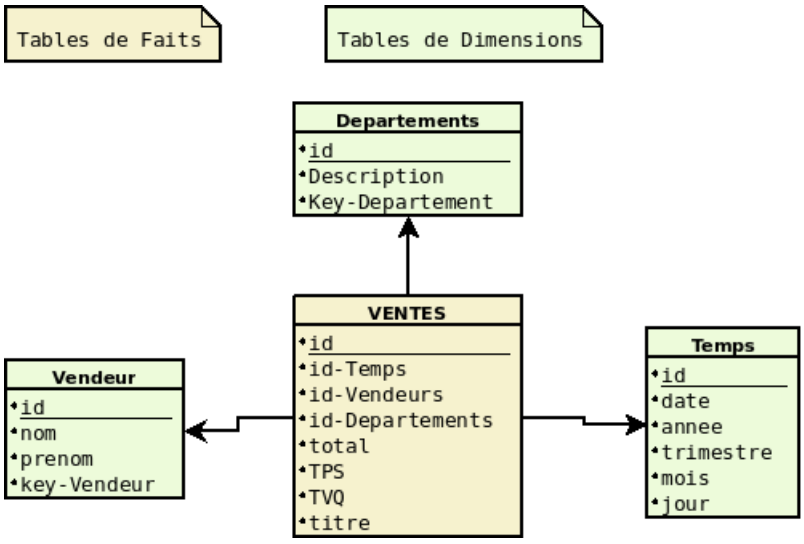

FIGURE 3 – Schéma étoile

Le modèle en flocon (snowFlake) consiste à éclater les dimensions de manière à obtenir des sous hiérarchies. Cela permet d'aller plus loin dans la granularité de hiérarchisation pour chaque dimension mais, rend le modèle plus complexe à utiliser.

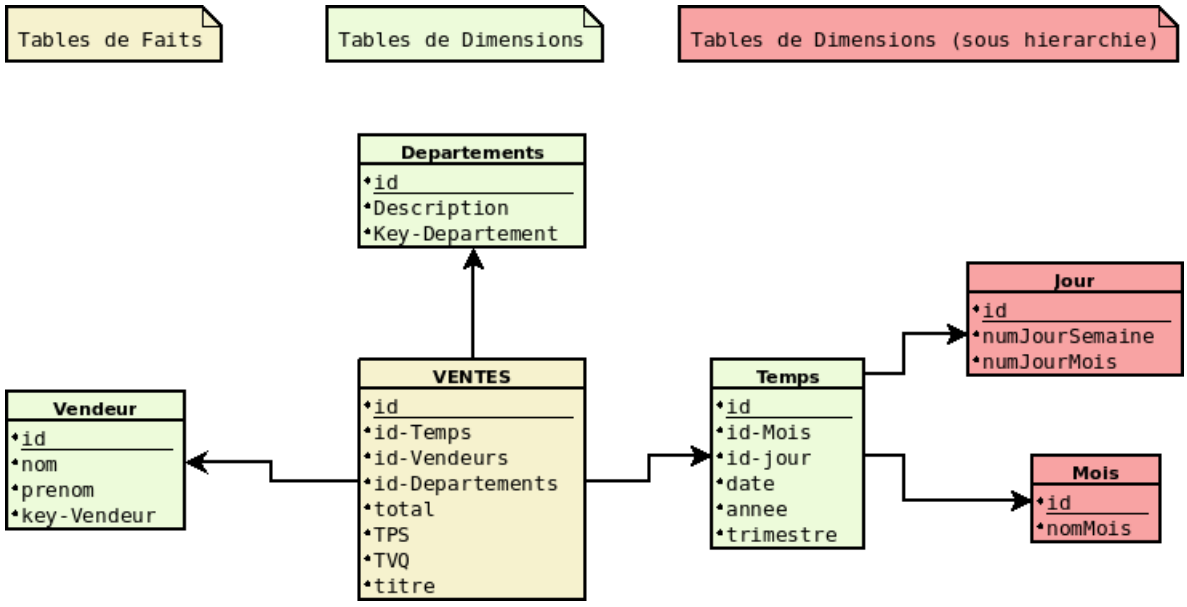

FIGURE 4 – Schéma en flocon

La modélisation en constellation est la fusion de plusieurs modèles en étoile qui utilisent des dimensions communes. Un modèle en constellation se compose de plusieurs faits et de dimensions partagées ou non.

2.3 SGBD NoSQL

Beaucoup d'organisations doivent stocker de grandes quantités de données, la plupart des SGBD relationnel actuels ne permettent plus de répondre aux besoins de stockage et de traitement de ces grandes quantités [Lea10]. C'est dans ce contexte qu'on a vu apparaître de nouveaux SGBD. On regroupe derrière le NoSQL l'ensemble des technologies de persistance qui se distinguent par une absence de requête et par un relâchement des caractéristiques ACID propres aux RDBMS (relational database management software) [Mic09]. La famille des SGBD NoSQL se compose de plusieurs catégories, les SGBD orienté colonne, orienté document, orienté graphe et clé/valeur. Ce sont les SGBD orienté colonne qui vont nous servir pour notre étude.

2.3.1 Qu'est-ce qui caractérise un SGBD NoSQL ?

Les propriétés ACID sont propres aux bases de données relationnelles. Elles sont :

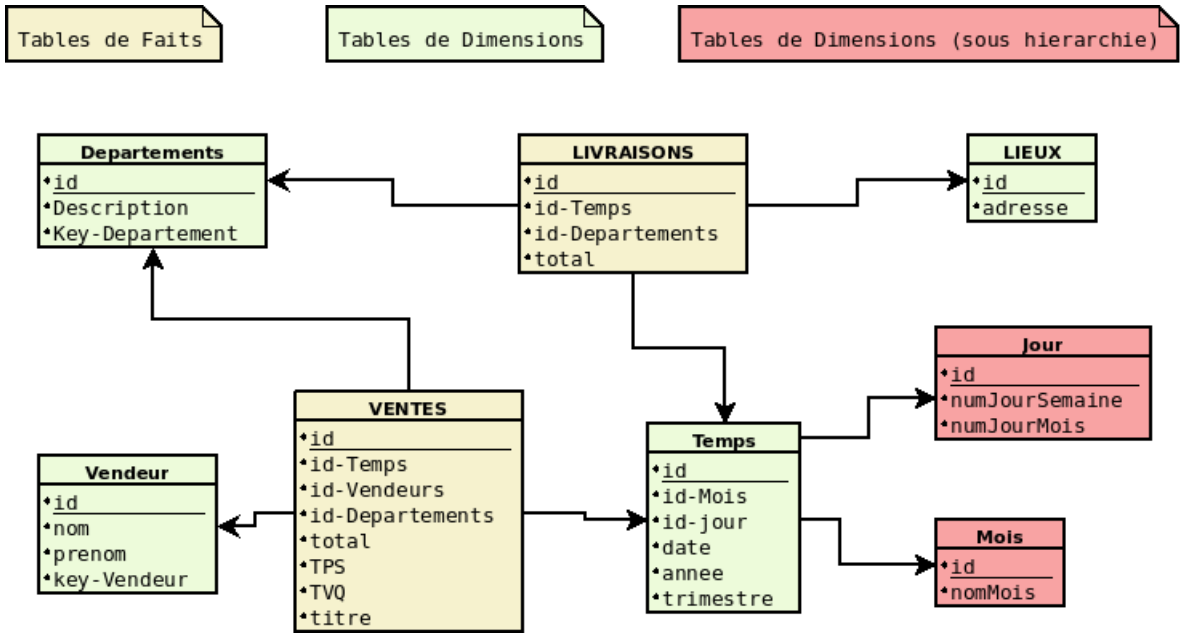

FIGURE 5 – Schéma en constellation

- Atomicité : l'ensemble des opérations d'une transaction sont réalisées ou toutes sont abandonnées pour revenir à l'état initial.
- Consistance : la base de données conserve un état consistant avant comme après l'exécution d'une transaction.
- Isolation : les autres opérations ne peuvent pas voir l'état intermédiaire des données modifiées par une transaction tant que celle-ci n'est pas terminée.
- Durabilité : assure qu'une fois que l'utilisateur a été notifié de la conclusion correcte d'une transaction, l'ensemble des modifications est persistante et ne sera pas annulé.

L'univers du NoSQL introduit l'acronyme BASE correspondant aux propriétés suivantes :

- Basically Available : essentiellement disponible.
- Soft state : état variable dans le temps.
- Eventually consistant : éventuellement consistant, une donnée répliquée peut, à un instant donné, ne pas avoir la même version sur les différents noeuds.

Le théorème CAP permet de faire le lien entre ces deux notions. En effet, il spécifie qu'une base de données ne peut offrir que deux des trois propriétés suivantes :

- Consistance

– Disponibilité
– Tolérance au partitionnement

Les bases de données relationnelles classiques adhèrent aux deux premières propriétés, tandis que les bases de données NoSQL se concentrent sur les deux dernières [Bro09].

2.3.2 Google : BigTable

BigTable est un SGBD dédié à l'entreposage de données. La structure de données d'une big-table est une map triée multidimensionnelle qui est persistante, distribuée et réparti. La map est indexée par une clé de ligne, une clé de colonne et un timestamp (moment). Chaque valeur de la map est un tableau non interprété de bytes[CDG+06].

Les points principaux pris en compte pour son développement ont été :
– capacité à manager de grandes quantités de données.
– performance flexible.
– architecture matérielle basée sur des ordinateurs de commodité.

BigTable se repose sur un système de fichier distribué (GFS google file system) et utilise le langage de programmation concurrente map/reduce lui aussi développé par Google.

2.3.3 Apache(HADOOP) : HBase

HBase fait partie du projet HADOOP de la fondation Apache, il s'agit d'un SGBD orienté colonne. Il a été calqué sur les travaux de BigTable de Google. La solution HBase se repose sur le système de fichier distribué (HDFS : hadoop distributed file system) qui se caractérise par une haute tolérance aux pannes et pouvant être déployée sur du matériel a faible coût.

Il permet un accès très rapide aux données et est adapté aux applications gérant de grands ensembles de données. D'autres part il propose aussi la possibilité de gérer les performances dans l'optique d'une utilisation plus flexible. Orienté colonne désigne la manière dont le SGBD sérialise les données. Contrairement au SGBD orienté ligne (SGBD orienté relationnel), les données sont sérialisées sous forme de colonne plutôt que de ligne.

La figure 6 présente le fonctionnement général d'HBase. Les tables sont stockées en orientation par colonne. Chaque table est constituée d'un ensemble de familles de colonnes (regroupements logiques de colonnes). Sur chaque famille de colonnes il est possible d'ajouter des colonnes. Les lignes des tables sont partitionnées en plusieurs régions. Lors de la création d'une table, une seule région est créée, elle est ensuite automatiquement divisée en sous-parties lorsque sa taille atteint un seuil limite. C'est le master qui

aura pour charge la gestion des régions. Zookeeper est un outil permettant de coordonner des services distribués, c'est lui qui permettra de faire le lien entre le Master et le système de fichier distribué HDFS, il permettra donc d'établir la manière dont seront exploitées les machines constituant le server [Mic10].

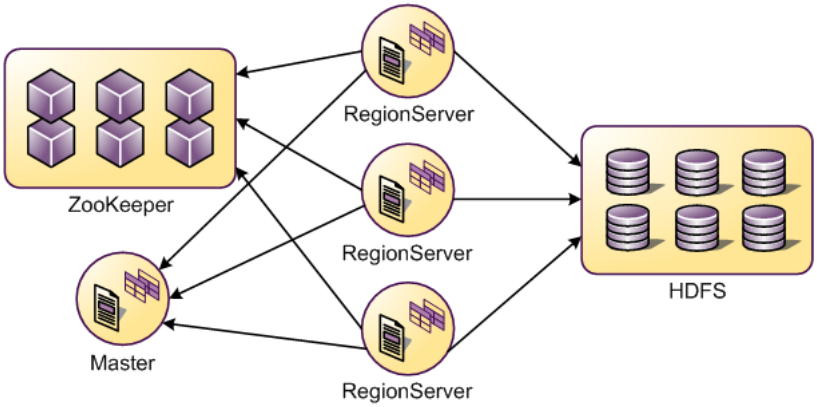

FIGURE 6 – hbase-schema

La figure 7 est une représentation d'un entrepôt pouvant être mis en place à l'aide d'HBase. La position d'une cellule est déterminée par une colonne, une ligne et un instant t.

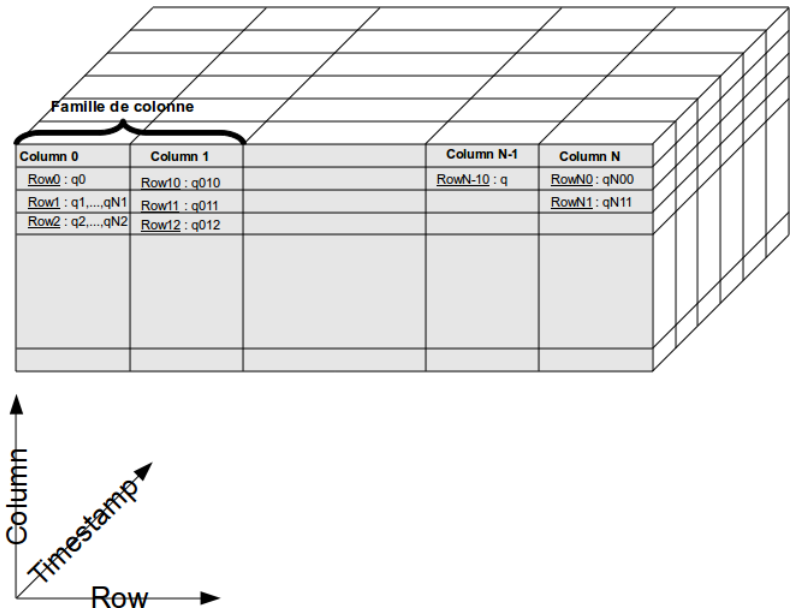

FIGURE 7 – cube

3 Les propositions

Description du contexte

La société Intégratik m'ayant accueilli en stage s'intéresse à proposer des services de Business Intelligence. Étant spécialisée dans les ERP, elle souhaite étudier la possibilité de connecter le système d'information de son ERP a une solution BI. Dans un soucis de performance et afin de rester compétitive elle s'intéresse fortement au concept de cloud computing. Ses objectifs sont multiples, a moyen terme elle souhaiterait proposer une offre similaire à celle proposée par "SalesForce", c'est à dire distribuer ces services ERP à partir d'une architecture cloud computing et à plus long terme, proposer des services de business intelligence.

Rafik Ouanouki est doctorant à l'ÉTS (école technique supérieure), il travaille sur ce même sujet, une solution BI basée sur une architecture cloud computing avec un système d'information orienté colonne. Son assertion est qu'il n'existe pas de tel système. Ses recherches visent à mettre en évidence ce point et par ailleurs tente de démontrer, l'apport d'un point de vue performance d'une telle solution.

Nos recherches se sont donc dirigée vers l'étude d'une solution permettant de connecter le système d'information de l'ERP de l'entreprise à une solution BI basée sur une architecture cloud computing.

Cette recherche nous a conduit aux deux points suivants :

- la conception de l'architecture d'une solution BI basée sur le concept de cloud computing
- la migration de données d'un SGBD SQL vers un SGBD NoSQL

Dans un premier temps nous présenterons les propositions pour concevoir une architecture basée sur le concept de cloud computing. Nous verrons ensuite les propositions relative a la conception d'une solution BI. Et enfin nous présenterons une procédure dédiée à réaliser une migration de données à partir d'un SGBD orienté relationnel vers un SGBD NoSQL.

3.1 Architecture de la solution BI cloud computing

Afin de proposer une solution BI basée sur le concept de cloud computing, nous nous sommes intéressés à la manière de mettre en place l'architecture matérielle et les différentes catégories de services (IaaS, PaaS et SaaS).

La solution proposée est d'utiliser des machines de commodité, il s'agit de machines quelconques. Cela présente plusieurs avantages, le premier étant qu'elles sont peu couteuses, le second avantage est qu'elles sont suffisantes pour assurer l'architecture et qu'il n'est pas nécessaire d'investir dans des

équipements spécialisés, main frame, serveur de stockage ... Cette configuration matérielle est rendue possible car le SGBD que nous utilisons se base sur ce type de configuration matérielle.

Afin d'administrer les machines nous proposons d'installer une distribution Linux (Ubuntu Server), présentant l'avantage d'être peu gourmande en ressource et étant reconnu pour sa stabilité.

Le réseau sera basé sur le protocole Ethernet présentant des débits allant de 10MBit/s a 10GBit/s. De plus dans un réseau Ethernet, le câble diffuse les données à toutes les machines connectées, de la même façon que les ondes radio, ce qui est adapté pour la communication entre les machines d'un cloud.

Afin de compléter la configuration des machines, nous utiliserons pour système de fichier, le système de fichier distribué HDFS (Hadoop Distributed File System), basé sur une implémentation en java nous installerons donc aussi l'environnement Java EE.

Cette partie sur la mise en place de l'architecture est présentée par la figure 8.

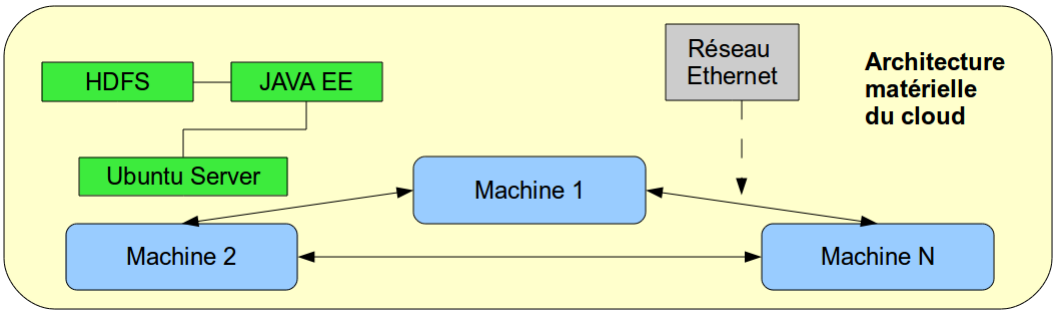

FIGURE 8 – architecture matérielle et configuration basique d'un cloud

Une fois la configuration du matériel réalisée, il nous faudra mettre en place les différentes couches de services permettant de l'exploiter.

La première couche de service est celle liée à l'exploitation de l'infrastructure. Les services liés au stockage seront basés sur la mise en place du SGBD HBase qui s'appuiera sur le système de fichier distribué HDFS. HBase dispose d'une interface nommé "stargate" basée sur une architecture REST (representational state transfer) permettant de distribuer ses fonctionnalités sous forme de web services en se basant sur le protocole HTTP, ces services étant déployé par Jetty, un serveur HTTP java.

L'avantage principal de cette architecture est l'interopérabilité. Roy T. Fielding a défini l'architecture REST [FT02] comme un ensemble de contraintes architecturale visant à minimiser les latences et les communications réseau tout en maximisant l'indépendance et l'extensibilité de l'implémentation

des composants. REST permet la mise en cache et la réutilisation des interactions, substituer dynamiquement des composants et le traitement d'actions par des intermédiaires répondant ainsi aux besoins d'un internet hypermedia extensible et distribué.

Cette partie est présentée par la figure 9.

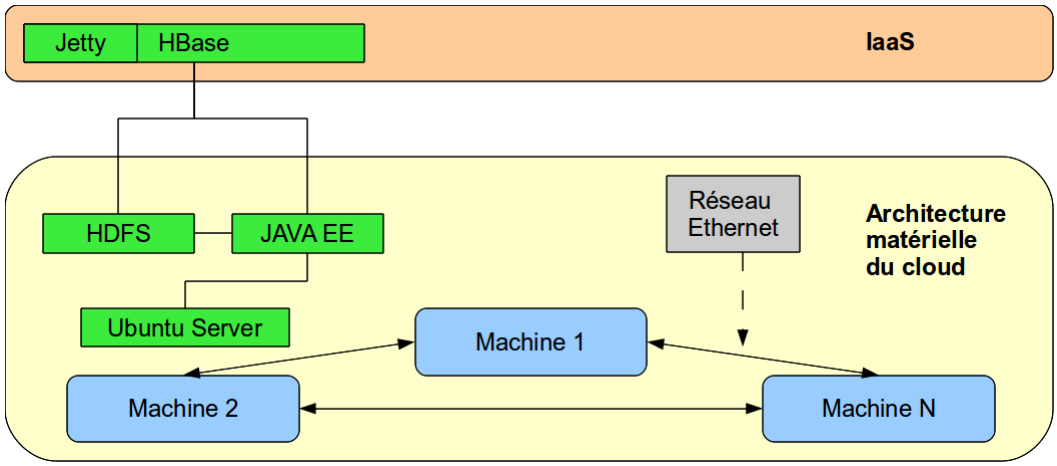

FIGURE 9 – IaaS cloud

La seconde couche de services est lié à l'environnement de développement et de distribution des applications nous utiliserons l'environnement Java EE et la distribution sera réalisée par l'utilisation conjointe de tomcat6 et axis2. Cela permettra de développer facilement des applications en java puis de les convertir sous forme de web services à l'aide d'axis2 et enfin de les déployer avec le serveur tomcat6.

Cette solution présente l'avantage que ces web services pourront être déployés sous forme de WSDL ce qui permettra qu'il soit consommé et exploiter facilement par des applications écrites dans un autre langage que Java.

Le framework de l'entreprise étant implémenté en C#, nous pourrons utiliser les outils mis a disposition dans l'environnement .NET permettant de faciliter leur consommation.

Cette partie est présentée par la figure 10.

Enfin la partie SaaS relative aux services distribués sera composé des services développés de manière à proposer les fonctionnalités des différents outils constituant une solution BI. La figure 11 est une vue de l'ensemble des points présentés précédemment.

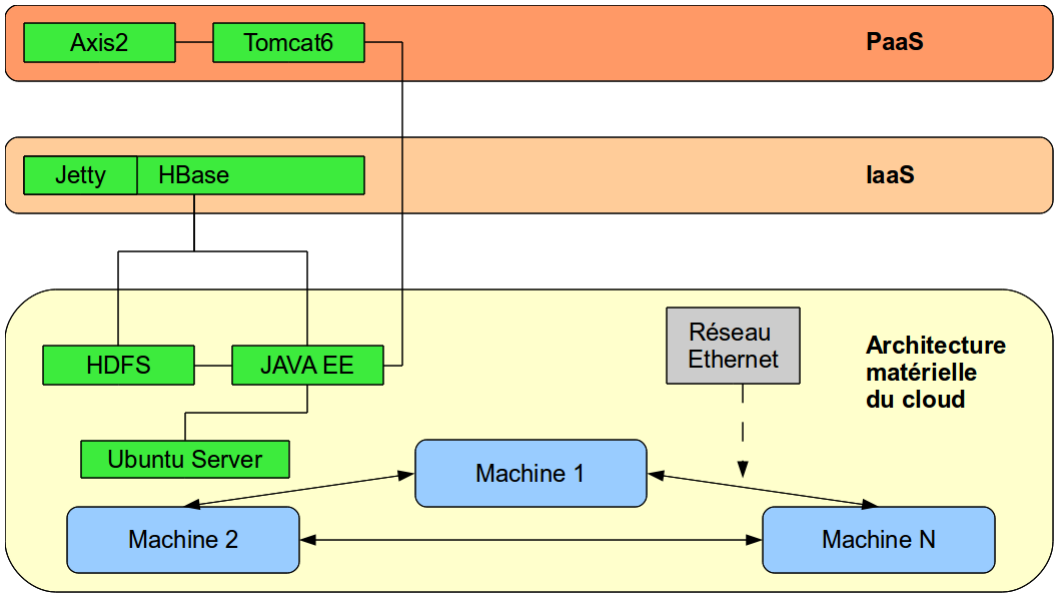

FIGURE 10 – PaaS cloud

La solution BI

La solution BI est présentée par la figure 12. Celle-ci sera donc basé sur une architecture orientée service. Cela permettra de développer les différents outils en suivant une approche composant. Une approche modulaire apportera une plus grande facilité en terme de lisibilité et maintenance du code. Les services proposés seront donc réalisés de manière à ce qu'ils soient utilisables comme des entités indépendante les unes des autres. Leurs découplage apportera plusieurs points importants :

- une plus grande facilité dans le déploiement des services de chaque outil.
- des avantages en terme de maintenance et de mises à jour.
- dans le cadre d'un travail en équipe, la possibilité de spécialiser et décomposer les tâches à réaliser.

Les différents outils développés interagiront ensemble en rapport aux services qu'ils doivent rendre. Par exemple, lors de la phase de peuplement initial, phase consistant à entreposer des données dans l'entrepôt, l'outil ETL requerra aux services d'écriture dans l'entrepôt fourni par l'outil DATA WAREHOUSE pour réaliser le chargement des données. De la même manière l'outil de DATA MINING nécessitera des services de lecture de l'entrepôt. La réalisation des web services sera donc basé sur ce principe.

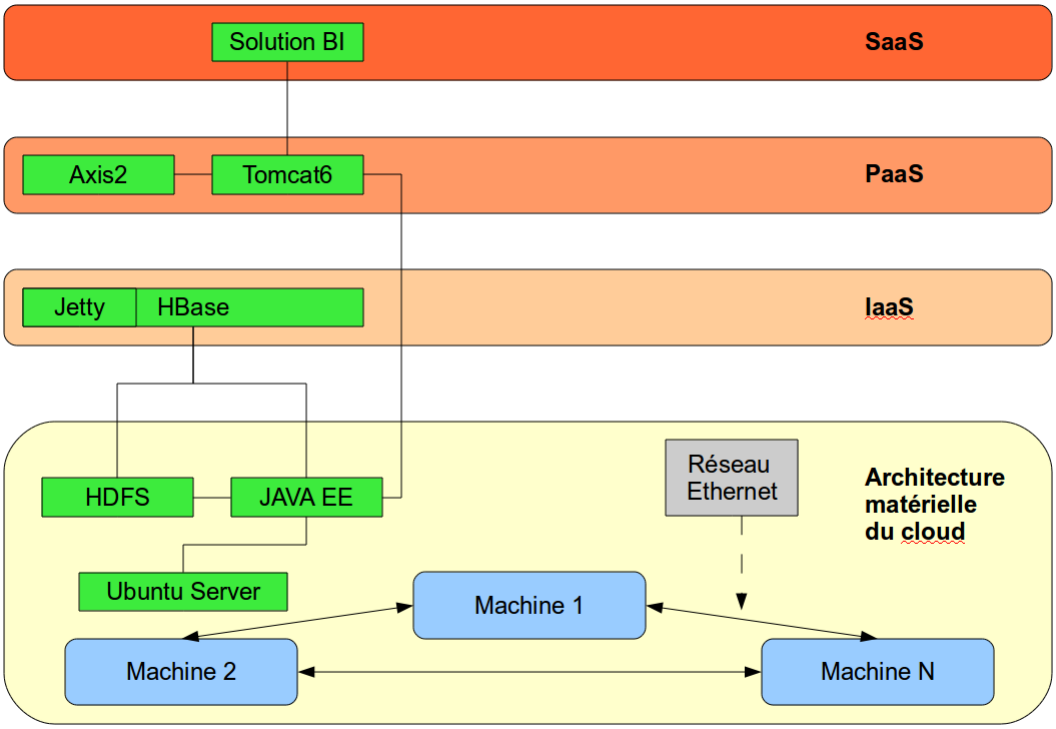

FIGURE 11 – SaaS cloud

3.2 migration SGBD SQL vers SGBD NoSQL

Afin de réaliser les tests de HBase nous nous sommes intéressés à la réalisation d'une procédure permettant de faciliter la migration à partir du SGBD de l'ERP (firebird) vers le SGBD de la solution BI (HBase)13.

Pour présenter chaque étape, nous nous appuierons sur un exemple d'une base de données représentant les ventes d'une entreprise.

3.2.1 Obtention du modèle relationnel de la base source

Cette étape consiste à obtenir la représentation du modèle relationnel de la base source. Pour cela nous avons besoin de récupérer le nom des tables, leurs attributs et leurs relations.

Les informations sur les relations peuvent être retrouvées à partir des définitions des contraintes de clés primaires et étrangères de chaque table.

Cette étape s'appuiera sur un modèle représentant le modèle relationnel 15. Ce modèle se compose de tables, relations et attributs. Les attributs sont dotés d'attributs permettant de définir s'il s'agit d'une clé primaire ou d'une

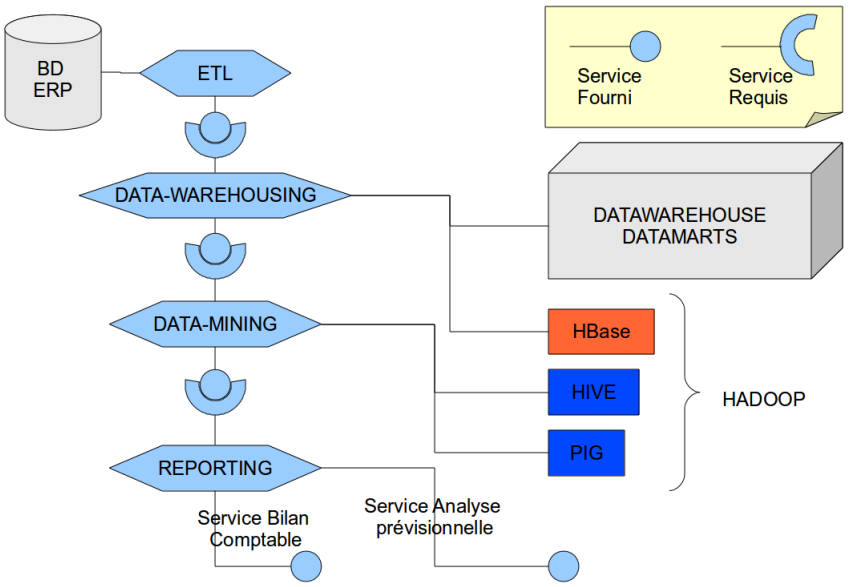

FIGURE 12 – solution BI

clé étrangère.

La figure 15, est une instance de notre modèle représentant une partie de notre base source exemple.

3.2.2 2éme étape : Définition du modèle en étoile

Les données de l'entrepôt seront organisée suivant un modèle multidimensionnel appelé modèle en étoile. Un modèle étoile est un modèle relationnel avec des spécifications supplémentaires tel que présentés dans le paragraphe 2.2.3. Cette étape consiste à définir le modèle qui permettra d'établir la manière dont seront organisées les données dans l'entrepôt. La figure 16 présente le modèle que nous avons défini pour organiser les données que nous allons récupérer de notre base source.

Dans ce modèle étoile, le fait est "LIGNEDEVENTE" et les dimensions sont "TEMPS", "VENDEUR" et "DEPARTEMENT".

3.2.3 3éme étape : Construction de la requête de migration

Au cours de cette étape il s'agit de définir le mapping entre les modèles issus des précédentes étapes. Cela consiste à définir quels attributs du modèle relationnel de la base source seront reliés a quels attributs du modèle en étoile cible.

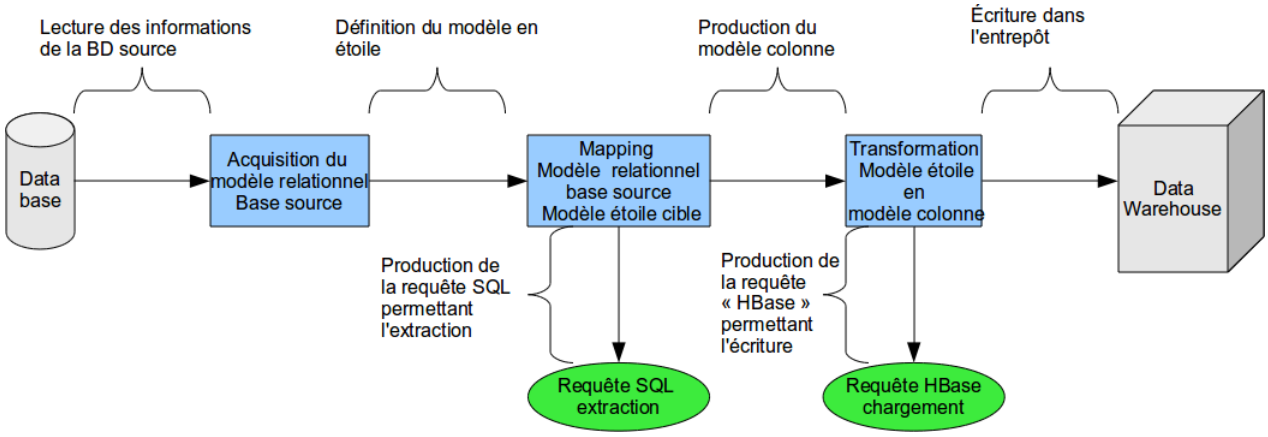

FIGURE 13 – procédure de Migration

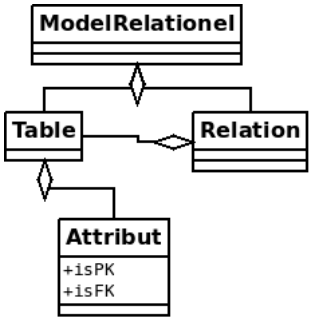

FIGURE 14 – modèle relationnel

La figure 17 montre un exemple d'association à réaliser, ces associations seront enregistrées (1, 2) et permettront de construire les requêtes SQL afin de récupérer les enregistrements correspondant aux attributs sélectionnés.

Dans un premier temps il s'agira de définir la requête permettant de charger les enregistrements correspondant a la table des faits, pour cela on sélectionnera l'ensemble des associations relative a cette table afin de générer une requête de la forme :
"select tableSource.attr , ...,tableSourcesN.attr from tableSource, ..., tableSourceN orderedby idUnique "

La requête sera réalisée selon une clé unique permettant en cas de panne de reprendre plus facilement.

Dans un second temps il s'agira de définir les requêtes permettant de charger les dimensions de la même manière ces requêtes seront réalisée suivant l'idUnique établi précédemment.

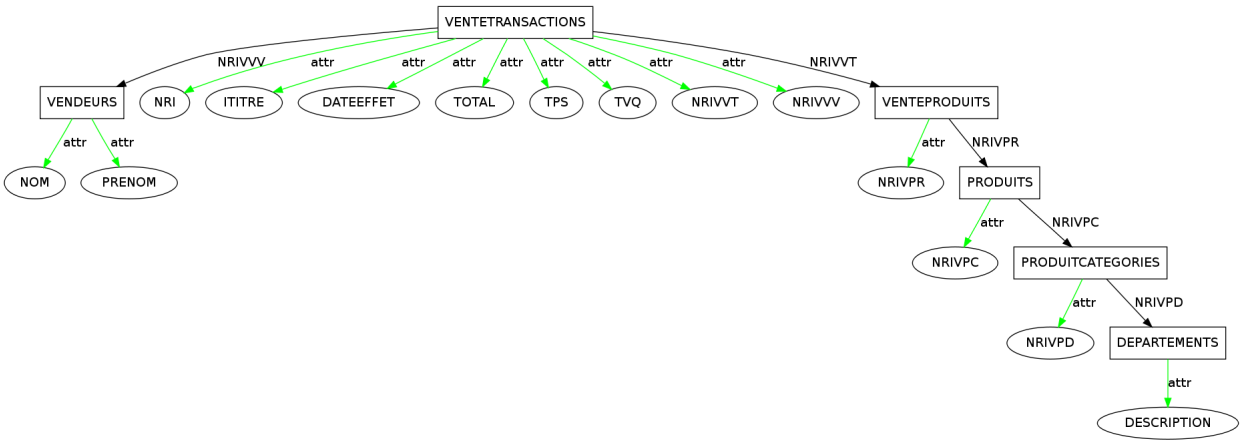

FIGURE 15 – modèle relationnel

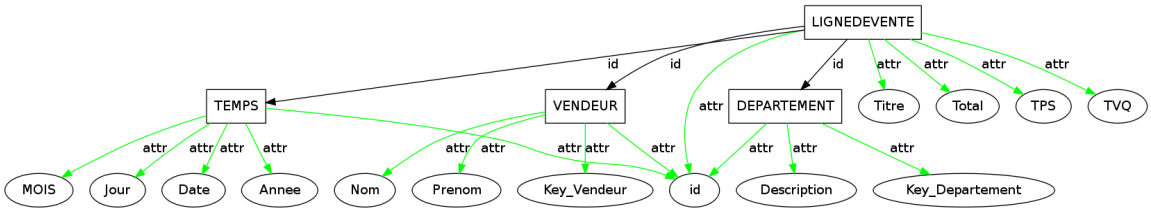

FIGURE 16 – modèle étoile

3.2.4 4éme étape : Écriture dans l'entrepôt

Au cours de cette étape, il s'agit de réaliser l'écriture des données récupérées par les requêtes précédentes dans l'entrepôt pour cela nous avons réalisé un modèle représentant la structure d'un entrepôt. Il se compose de famille de colonnes, de colonnes et de lignes. Une famille de colonnes pouvant être un datamart ou un data warehouse 18.

Dans un premier temps, il s'agit de construire la structure correspondant au modèle en étoile défini dans l'étape 2. Nous obtenons ainsi le modèle représenté par la figure 19.

La seconde étape consiste à réaliser le chargement de notre structure. Pour cela, il s'agit de générer les requêtes basés sur le langage de Hbase, les requêtes d'écriture étant générée par la transformation du modèle étoile au modèle columnOriented.

3.3 Les résultats attendus

A travers ces travaux nous souhaitons valider le choix du passage a une architecture cloud computing, en vérifiant que le SGBD HBase est exploitable,

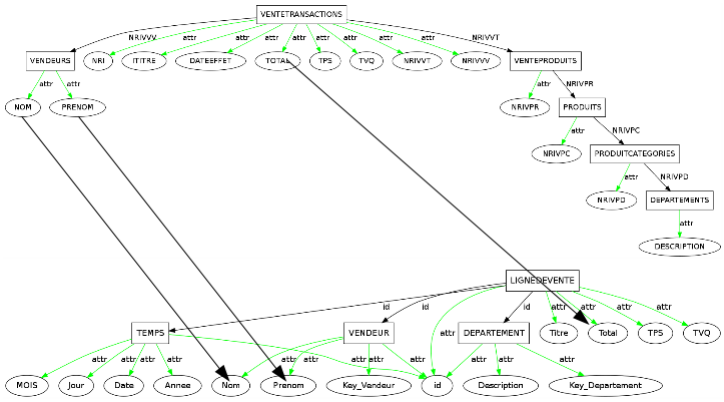

FIGURE 17 – mapping

model.pathModel.PathModelS@1114460	model.pathModel.PathModelR@8ff4cf
bdName : VENTE	bdName : INTEGRATIK
tableName : DEPARTEMENT	tableName : VENTETRANSACTIONS
fieldName : id	fieldName : NRI

TABLE 1 – exemple d'association entre les modèles source et cible

que celui-ci apporte en performance et enfin qu'il est adapté pour l'entreposage et l'analyse de données d'entreprises.

4 Description du prototype réalisé

Le prototype a été réalisé suivant la méthode AGILE, au fur et à mesure de son avancement, celui-ci était présenté au patron d'Intégratik et les fonctionnalités et orientation à prendre était rediscutée. Afin de décrire le prototype réalisé nous présenterons les langages utilisés, les choix concernant l'architecture du prototype et enfin les principales fonctionnalités offertes.

4.1 Le choix des langages

Nous avons choisi le langage Java pour l'implémentation du code métier, car il s'agit du langage d'implémentation du SGBD HBase.

Le souhait de l'entreprise étant d'intégrer ce projet à son framework écrit en C# , nous avons décidé d'utiliser axis2 nous permettant de réaliser des web services à partir du code java. Axis2 présente l'avantage de pouvoir générer les fichiers de description wsdl (web service description language) à partir des web services réalisés.

model.pathModel.PathModelS@edf389	model.pathModel.PathModelR@16be68f
bdName : VENTE	bdName : INTEGRATIK
tableName : VENDEUR	tableName : VENTETRANSACTIONS
fieldName : Key Vendeur	fieldName : NRIVVV

TABLE 2 – exemple d'association entre les modèles source et cible

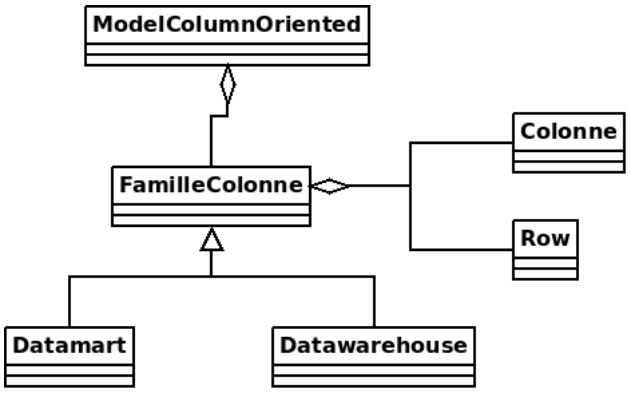

FIGURE 18 – modèle colonne

L'environnement .NET de visual studio permet la consommation de web services à partir de wsdl. Cela permettra de développer un client permettant d'exploiter l'application écrite en java.

En outre HBase est un SGBD NoSQL, qui ne possède pas de langage défini. Nous avons donc implémenté un pseudo-langage utilisant les principales primitives nous permettant de l'administrer.

Administration d'un entrepôt

- createCube(String nom) : *création d'une famille de colonne.*
- deleteCube(String nom) : *suppression d'une famille de colonne.*
- addColumn(String cubeName, String nomColumn) : *ajout d'une colonne à une famille de colonne.*

Accés a un entrepôt

- getCubeList() : *obtenir la liste des familles de colonnes.*
- getColumns(String cubeName) : *obtenir la liste des colonnes d'une famille de colonne.*
- getRows(String tableName, String columnName) : *obtenir la liste des lignes d'une famille de colonne.*

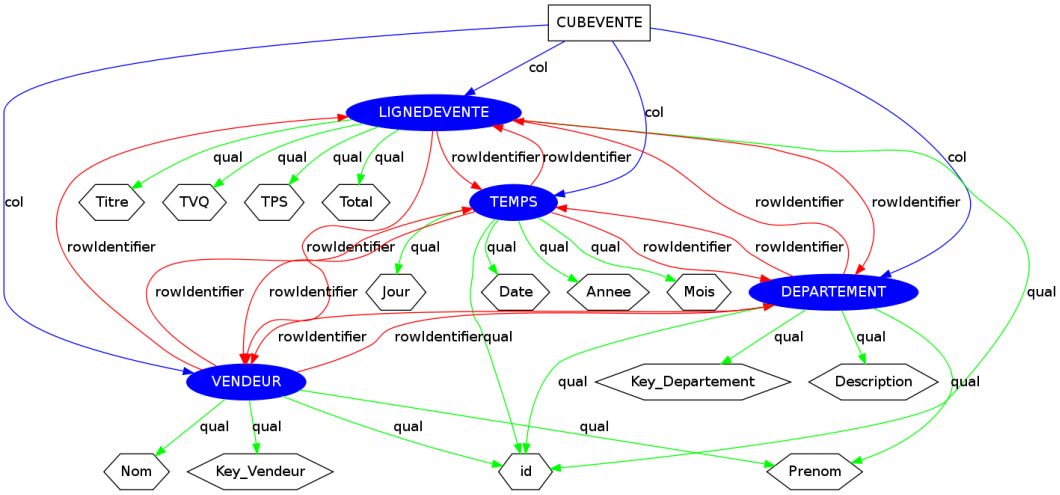

FIGURE 19 – modèle de cube

- getQualifiers(String cubeName, String columnName) : *obtenir la liste des "attributs" d'une colonne.*
- getSetofValuesByQualifier(String cubeName, String columnName, String qualifierName) : *obtenir les enregistrements suivant : colonne/ligne/attribut.*
- getRowIdentifiersByValue(String cubeName, String columnName, String qualifierName, String value) : *obtenir la liste des lignes contenant l'enregistrement value suivant : colonne/attribut/value.*
- getData(String cubeName, String columnName, String qualifierName) : *obtenir les enregistrements à partir : famille de colonnes/colonne/attribut.*

4.2 La structure

Nous avons représenté sur la figure 20 les principaux packages du projet. Nous avons représenté la structure utilisée pour implémenté nos modèles, il s'agit du patron Composite, tous les autres modèles implémentés héritent de la classe modèle. Pour des raisons de lisibilité, nous n'avons introduit que les packages principaux.

Pour le package WEBSERVICES, nous avons mis les quatre premiers web services réalisés pour lequel nous avons généré les wsdl.

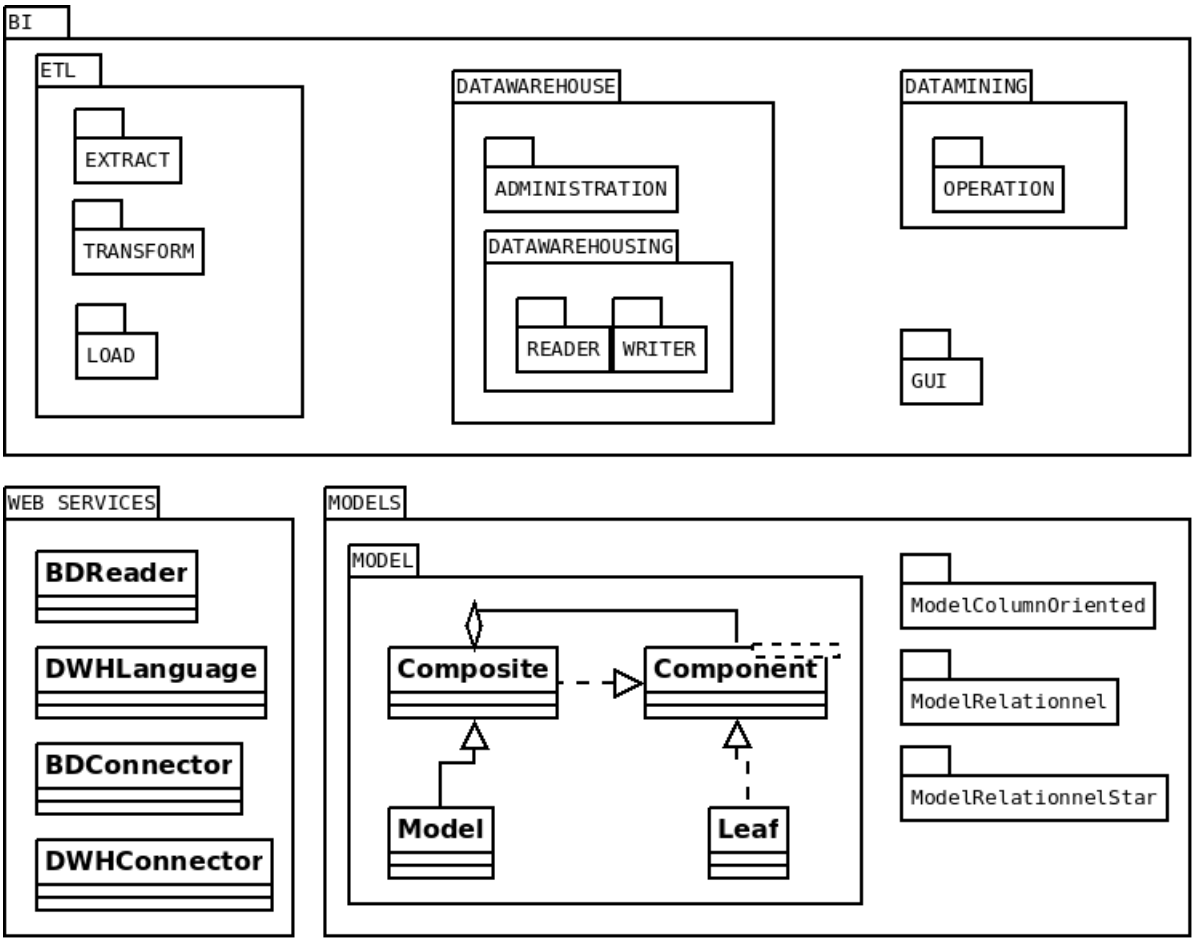

FIGURE 20 – le projet

4.3 Le prototype

ETL

Notre ETL permet donc de se connecter à une base de données FireBird et de consulter les différentes tables comme le montre la figure 21.

Il permet aussi de réaliser le mapping entre, le modèle source et le modèle cible, avec la possibilité de sérialiser et exporter dans un fichier le mapping réalisé.

DATA-WAREHOUSING

L'outil de DATA-WAREHOUSING permet de manager l'entrepôt, il permet de mettre la création et la visualisation de la structure et des valeurs

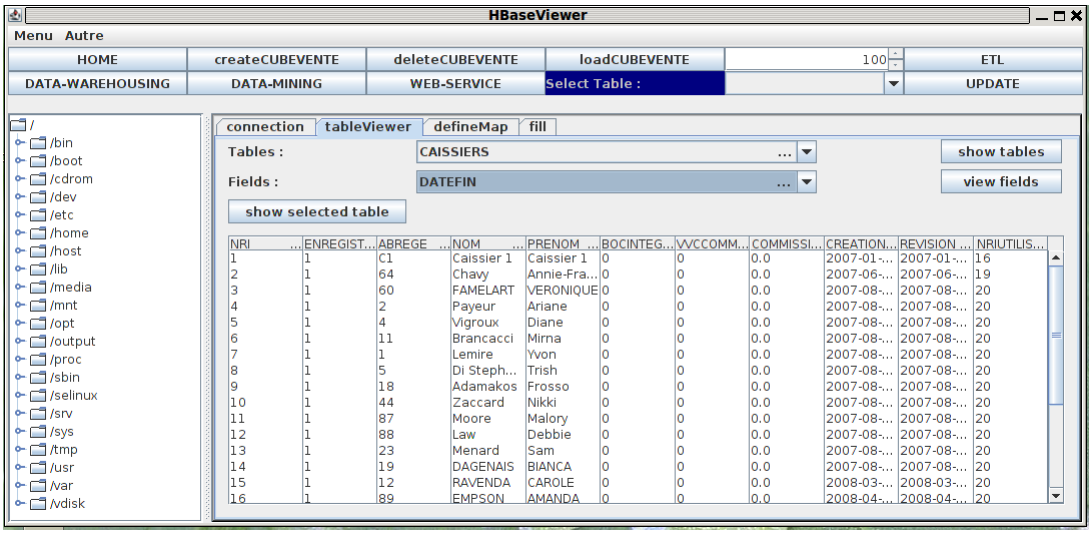

FIGURE 21 – visualisation de la BD source

d'un entrepôt comme le montre les figures 22.

DATA-MINING

L'outil de DATA-MINING a été réalisé afin de réaliser l'affichage des résultats d'une requête sur l'entrepôt de données permettant d'obtenir le total de chaque vente ainsi que le total par départements.

WEB SERVICES

Voici un extrait du fichier DWHLangage.wsdl présentant la manière dont axis2 permet de générer les wsdl. Nous pouvons donc déployer des web services permettant d'administrer notre prototype à partir des fichiers wsdl. Ceux-ci pouvant être consommé par la suite par des clients implémentés en C#.

```
1     <wsdl:portType name="DwhLangagePortType">
2         <wsdl:operation name="deleteCube">
3             <wsdl:input message="axis2:deleteCubeRequest"
                  wsaw:Action="urn:deleteCube"/>
4             <wsdl:output message="axis2:deleteCubeResponse"
                  wsaw:Action="urn:deleteCubeResponse"/>
5         </wsdl:operation>
6         <wsdl:operation name="addColumn">
7             <wsdl:input message="axis2:addColumnRequest" wsaw
                  :Action="urn:addColumn"/>
8             <wsdl:output message="axis2:addColumnResponse"
                  wsaw:Action="urn:addColumnResponse"/>
9         </wsdl:operation>
10        <wsdl:operation name="createCubeVente">
```

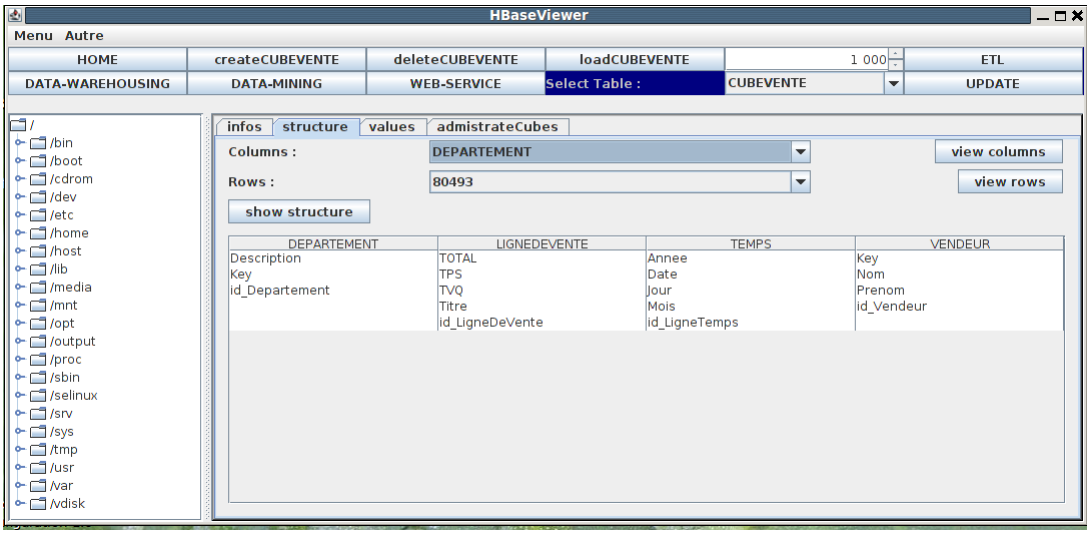

FIGURE 22 – visualisation de l'entrepôt

```
11              <wsdl:input message="axis2:createCubeVenteRequest
                    " wsaw:Action="urn:createCubeVente"/>
12              <wsdl:output message="axis2:
                    createCubeVenteResponse" wsaw:Action="urn:
                    createCubeVenteResponse"/>
13          </wsdl:operation>
14          <wsdl:operation name="createCube">
15              <wsdl:input message="axis2:createCubeRequest"
                    wsaw:Action="urn:createCube"/>
16              <wsdl:output message="axis2:createCubeResponse"
                    wsaw:Action="urn:createCubeResponse"/>
17          </wsdl:operation>
18          <wsdl:operation name="createCubeVenteBrut">
19              <wsdl:input message="axis2:
                    createCubeVenteBrutRequest" wsaw:Action="urn:
                    createCubeVenteBrut"/>
20              <wsdl:output message="axis2:
                    createCubeVenteBrutResponse" wsaw:Action="urn
                    :createCubeVenteBrutResponse"/>
21          </wsdl:operation>
22      </wsdl:portType>
```

5 Présentation des résultats

5.1 Tests

Le but du prototype étant de démontrer la possibilité d'exploiter HBase avec des données d'entreprises nous avons réalisé plusieurs tests.

Le premier test réalisé en local a consisté :

– création d'une structure : 1 famille de colonne, 10 colonnes, 1000 lignes.

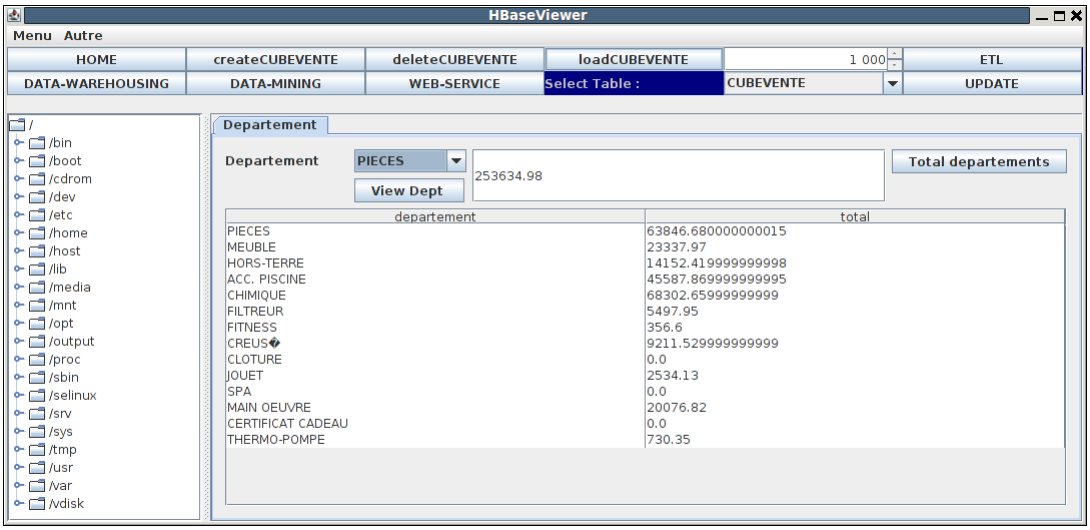

FIGURE 23 – datamining

– écriture : ajout d'une valeur sur chacune des colonnes, sur chaque ligne allant de 0 a 999.
– lecture : somme des valeurs de chaque colonne.

Le tableau 3 présente les résultats.

opération	temps (ms)
création structure	5429
écriture	509181
lecture	7054

TABLE 3 – test basique

temps total : 8 min 45 sec

Un second test en local a consisté à procéder a une migration de 10 000 puis 100 000 transactions (une transaction équivaut a 17 enregistrements). Le tableau 4 présente les résultats.

nbe de lignes	temps (min)
10 000	16
100 000	50

TABLE 4 – test migration

Un troisième test en local a consisté à procéder a l'analyse de 10 000 puis 100 000 enregistrements de transactions afin d'obtenir le total par départements, par mois et par année. Le tableau 5 présente les résultats.

nbe de lignes	temps (min)
10 000	16
100 000	42

TABLE 5 – analyse des ventes par départements (Mois et Année)

Le dernier test a été lancé sur un cluster composé de 4 machines :
Le test réalisé a été identique au premier. Celui-ci n'a pas fonctionné.

5.2 conclusion sur les tests

Ces premiers tests ont montré que le SGBD est utilisable, il a pu être employé pour réaliser notre prototype et son orientation colonne offre une structure correspondant à la représentation d'un cube M-OLAP, donc parfaitement adapté pour accueillir des données organisées suivant une modélisation multidimensionnelle.

Les tests réalisés en local ont présenté des performances faibles (les tests d'analyses réalisés sur les mêmes jeux de données à l'aide de FireBird n'ont pris que quelques secondes).

Le test sur le cluster n'a pas fonctionné. Il s'agit d'un premier test permettant de valider son montage, nous n'avons donc pas encore pu évaluer les performances d'HBase sur la configuration pour laquelle il est prévu.

Ces premiers tests permettent donc de mettre en avant les difficultés d'ordre technique pour exploiter HBase. Par ailleurs cette solution évolue très rapidement (une nouvelle version tous les deux mois en moyenne). Cependant, il s'agit d'une solution récente que nous avons rapidement pu employer pour réaliser notre prototype et dont le principe de fonctionnement est intéressant. Les prochains tests apporteront les précisions nécessaires pour savoir si cette solution peut être exploitable en production.

5.3 Les perspectives

Les perspectives de ce projet :

- à court terme : terminer les tests sur le cluster nous permettant de déterminer si la solution est exploitable (d'un point de vue performance), implémenter un client C# permettant d'administrer HBase.
- à moyen terme : exploiter l'architecture du cloud au niveau PaaS (voir S.O.A : Service Grids [HB02]).
- à long terme : développer les outils de la solution BI.

6 Conclusion

Nous nous sommes intéressés à la conception d'une solution BI basée sur le concept de cloud computing. Nous avons pour cela tenter de comprendre le concept de cloud computing et l'on a pu constater le succès des fournisseurs de services adoptant ce concept (Salesforce).

Par ailleurs nous avons réalisé une étude sur la business intelligence afin de définir les principaux outils constituant une solution BI.

Suite à cette étude, nous avons cherché à déterminer quel système d'information utiliser pour notre solution, pour cela nous nous sommes tournés vers un nouveau type de SGBD NoSQL. Au vue des performances de BigTable, nous avons opter pour HBase, un SGBD orienté colonne basé sur les mêmes principes que BigTable.

Le système d'information étant le cœur de notre projet, nous avons réalisé un prototype afin de tester les performances de notre SGBD. Les premiers tests réalisés à partir de ce prototype ne sont pas très concluant, nous n'avons encore pu établir le gain en performance.

Cependant, nous avons pu l'employer rapidement pour réaliser notre prototype et avons constaté que sa structure s'adapte bien à la réalisation d'un système d'information d'une solution BI.

D'un point de vue personnel, j'ai pu découvrir le domaine de la business intelligence ainsi que le concept de cloud computing. Par ailleurs, j'ai eu l'occasion d'observer et de m'impliquer dans la mise en place d'un projet de recherche au sein d'une entreprise.

7 Remerciements

Je tiens à remercier mes encadrants de stage : Alain April et Claude Ethier pour leur confiance et leur encadrement. Ils m'ont guidé et conseillé durant mes recherches. Parfaitement intégré au sein de l'entreprise Integratik, se stage s'est déroulé dans une ambiance studieuse et cordiale. Je remercie aussi très chaleureusement Marianne Huchard, Clémentine Nebut et Chouki Tibermacine pour le grand soutien qu'ils m'ont apporté ainsi que tous mes professeurs pour leurs enseignements. Enfin, je remercie pour leur soutien mes proches : amis et parents.

Table des figures

Liste des tableaux

Glossaire

Cadriciel En informatique, un framework, ou armature, est un ensemble de bibliothèques, d'outils et de conventions permettant le développement d'applications. Il fournit suffisamment de briques logicielles et impose suffisamment de rigueur pour pouvoir produire une application aboutie et dont la maintenance est aisée. Ces composants sont organisés pour être utilisés en interaction les uns avec les autres (voir urbanisation).Des tentatives de francisation du terme ont été faites. On trouve ainsi parfois les termes cadre d'applications, proposé par l'Office québécois de la langue française ou cadriciel. L'expression atelier de développement est également employée.

data warehouse Le terme entrepôt de données ou data warehouse désigne une base de données utilisée pour collecter et stocker de manière définitive des informations volatiles provenant d'autres bases de données.

DataMart (littéralement en anglais magasin de données) est un sous-ensemble d'une base de données relationnelle utilisé en informatique décisionnelle ; il est généralement exploité en entreprise pour restituer des informations ciblées sur un métier spécifique, constituant pour ce dernier un ensemble d'indicateurs à vocation de pilotage de l'activité et d'aide à la décision. Un DataMart, selon les définitions, est issu ou fait partie d'un DataWarehouse, et en reprend par conséquent la plupart des caractéristiques.

ERP Progiciel qui permet de gérer l'ensemble des processus d'une entreprise en intégrant l'ensemble de ses fonctions, dont la gestion des ressources humaines, la gestion comptable et financière, l'aide à la décision, mais aussi la vente, la distribution, l'approvisionnement et le commerce électronique.

ETL Extract-Transform-Load est connu sous le terme Extracto-Chargeur, (ou parfois : datapumping). Il s'agit d'une technologie informatique intergicielle (comprendre middleware) permettant d'effectuer des synchronisations massives d'information d'une base de données vers une autre. Selon le contexte, on traduira par « alimentation », « extraction », « transformation », « constitution » ou « conversion », souvent combinés.

hypermedia Un hypermédia est un média dans lequel les informations ne sont pas seulement de type texte, mais également de type image, son, vidéo ou encore multimédia, ont vocation à présenter l'information . C'est une extension de l'hypertexte à des données multimédias.

OLAP Online Analytical Processing désignait à l'origine les bases de données multidimensionnelles (aussi appelées cubes ou hypercubes) destinées à des analyses complexes sur des données.

Progiciel Mot-valise, contraction de produit et logiciel, est un logiciel applicatif commercial "prêt-à-porter", standardisé et générique, prévu pour répondre à des besoins ordinaires. Ce terme s'oppose aux logiciels sur mesure développés en interne dans une entreprise et conçus pour répondre à des besoins spécifiques.

Références

[AFG+09] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, et al. Above the clouds : A berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.

[Bro09] Julian Browne. Brewer's cap theorem, 2009. http ://www.julianbrowne.com/article/viewer/brewers-cap-theorem.

[CCS93] E.F. Codd, S.B. Codd, and C.T. Salley. Providing OLAP (on-line analytical processing) to user-analysts : An IT mandate, 1993.

[CDG+06] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.E. Gruber. Bigtable : A distributed storage system for structured data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI'06)*, 2006.

[FT02] R.T. Fielding and R.N. Taylor. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2) :115–150, 2002.

[HB02] J. Hagel and J.S. Brown. Service Grids : The Missing Link in Web Services. *White Paper*, 2002.

[KC04] R. Kimball and J. Caserta. *The data warehouse ETL toolkit : practical techniques for extracting, cleaning, conforming, and delivering data.* Wiley, 2004.

[Lea10] N. Leavitt. Will NoSQL Databases Live Up to Their Promise ? *Computer*, pages 12–14, 2010.

[LW96] C. Li and X.S. Wang. A data model for supporting on-line analytical processing. In *Proceedings of the fifth international conference on Information and knowledge management*, pages 81–88. ACM, 1996.

[Mic09] Figuiere Michael. Nosql europe : Tour d'horizon des bases de données nosql, 2009. http ://blog.xebia.fr/2009/11/18/devoxx-jour-1-nosql-avec-hbase/.

[Mic10] Figuiere Michael. Nosql europe : Tour d'horizon des bases de données nosql, 2010. http ://blog.xebia.fr/2010/04/21/nosql-europe-tour-dhorizon-des-bases-de-donnees-nosql/.

Annexes

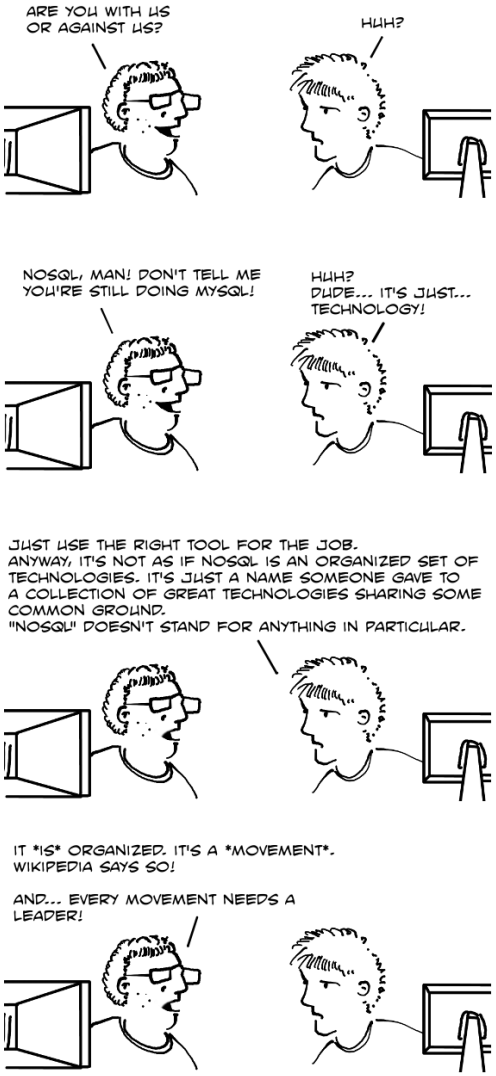

Figure 24 – NoSQL vs SQL (1)

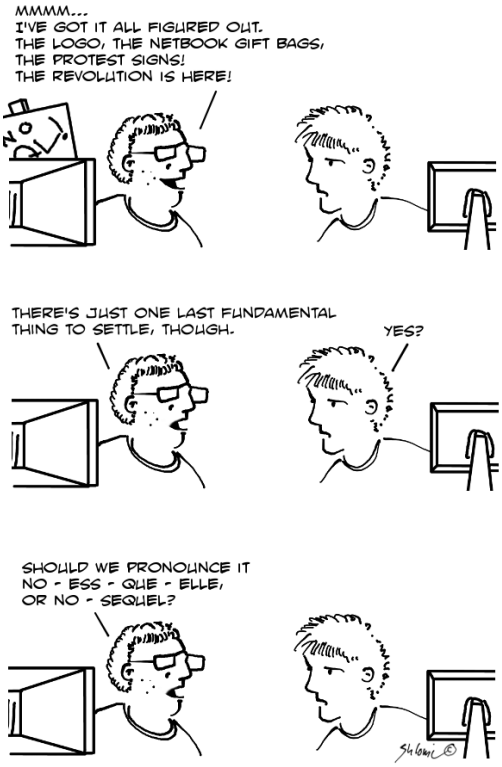

Figure 25 – NoSQL vs SQL (2)