Master II Recherche en Informatique Concept aux Systèmes
Université de Versailles Saint-Quentin
2008-2009

Système de médiation à base de services pour l'évaluation de la qualité des données

Réalisé par

Mohamed Reda BOUADJENEK
Soutenu le 17 Septembre 2009

Laboratoire PRiSM Université de Versailles Saint-Quentin
45, avenue des Etats-Unis, 78035
Versailles Cedex

Directeur de recherche :
Pr Mokrane BOUZEGHOUB

Remerciements

J'adresse tout d'abord mes plus vifs remerciements au Professeur Mokrane BOUZEGHOUB qui m'a accueilli au sein de son équipe au laboratoire PRiSM et qui m'a permis d'accomplir ce travail.

Je remercie également le Professeur Djamel Abdelkader ZIGHED pour m'avoir aidé à trouver ce stage.

Je tiens aussi a remercier tout les membres de notre équipe AMIS Stéphane LOPES,Daniela GRIGORY, Zoubida KEDAD, Ahmed, Sofiane et Isma pour leur soutien et leurs encouragements.

Je tiens à remercier en particulier Fernando LEMOS pour son aide précieuse et ses encouragements à l'accomplissement de ce travail.

Je tiens aussi à remercier tous les membres du laboratoire PRiSM : Redouane, Nazim, Lamia, Nabila, Amir, Ahmed K, Boubkeur, Louness, Chahinez, Houssem, Hamza, Samir, Yacine, Merouane, etc et ceux qui m'ont aidé de prés ou de loin à l'élaboration de ce travail.

J'exprime ma gratitude à tous les auteurs qui m'ont autorisé l'accès à leurs publications lors de la phase de recherche bibliographique.

Enfin je souhaite remercier tous les membres de ma famille qui m'ont beaucoup aidé et encouragé à la réalisation de ce travail.

Résumé

La gestion de la qualité des données est un problème clé au sein de toutes les organisations qu'elles soient privées ou publiques. Une large collection d'outils commerciaux et open source est proposée pour gérer les problèmes de qualité des données dans les systèmes d'information. Cependant, chacun de ces outils a sa propre vision de la qualité des données. D'un côté, faire interagir ces outils entre eux, demeure un défi technique en raison de l'hétérogénéité de leurs modèles et méthodes d'accès. D'un autre côté, les analystes de la qualité exigent de plus en plus de facilité d'intégration, leur permettant de consolider et de regrouper plusieurs mesures de qualité acquises et provenant de différentes observations. Ce document présente l'architecture de la *QBox-Services*, une évolution de la plate-forme *QBox-Foundation* qui a été définie dans le cadre du projet ANR *Quadris*. Cette plate-forme fournit une infrastructure d'intégration à base de services qui permet l'interopérabilité de plusieurs outils de qualité trois tiers et une analyse multidimensionnelle des mesures effectuées basée sur OLAP.

Mots clés : Qualité des données, Médiation, Services Web, Patterns d'accès, Normalisation des données.

Abstract

Data quality management is a key problem for all kinds of public and private organizations. A large spectrum of commercial and open source tools have been proposed for dealing with data quality problems in information systems. However, each quality tool has its particular view of data quality. On the one side, the interoperability among these tools remains a technical challenge because of the heterogeneity of their models and of their access interfaces. On the other side, quality analysts require more and more integration facilities that allow them to consolidate and aggregate multiple quality measures acquired from different observations. This document focuses on the architecture of the *QBox-Services*, an evolution of the *QBox-Foundation* platform which has been developed within the ANR *Quadris* project. This platform supplies a service-based integration infrastructure that allows interoperability among several third part quality tools and provides an OLAP-based quality model to support multidimensional analysis.

Keys words : Data quality, Mediation, Web Services, Access patterns, Data normalisation.

Table des matières

Table des figures

Liste des tableaux

Introduction

Contexte

La technologie dont nous disposons aujourd'hui permet le développement de systèmes d'information complexes, qui offrent un accès à une vaste quantité d'informations distribuées dans des sources de données hétérogènes. Bien que ces systèmes soient proposés depuis plus d'une décennie, ils sont devenus de plus en plus importants ces dernières années, aussi bien dans le domaine académique que dans le domaine industriel. Le besoin croissant de ce type de système est dû principalement à la prolifération des données disponibles sur des sites distants, souvent stockées dans diverses plate-formes et avec divers formats.

Les besoins d'accéder de façon uniforme à des sources de données multiples sont chaque jour de plus en plus forts, particulièrement, dans les systèmes décisionnels qui ont besoin d'une analyse compréhensive des données. Les Systèmes d'Intégration de Données (DIS pour Data Integration System en alglais) sont apparus pour répondre à ces besoins. Un système d'intégration de données est un système d'information qui intègre des données de sources différentes et qui fournit à l'utilisateur une vision d'une base de données unique. Les systèmes d'intégration de données répondent aux requêtes des utilisateurs avec des données obtenues à partir de plusieurs sources de données. L'intégration de l'information est le problème qui consiste à combiner des données disponibles dans diverses sources de données et de fournir à l'utilisateur un schéma global et unifié de ces données, qui représente les besoins potentiels des utilisateurs. Le schéma global représente une vue générale de l'information qui peut être interrogée par l'utilisateur. Ce type de système libère l'utilisateur d'avoir à repérer les sources de données pertinentes à une requête, à interagir avec chaque source séparément, et à combiner manuellement les données provenant de ces différentes sources.

Les systèmes de médiation sont des exemples de systèmes d'intégration de données, qui fournissent un accès à des données extraites à partir de diverses sources de données, et permet de les intégrer de façon transparente pour répondre aux requêtes des utilisateurs. Les entrepôts de données permettent aussi d'extraire, de transformer et d'intégrer des données issues de diverses sources de données mises à la disposition des managers, pour la prise de décisions stratégiques. Les portails Web sont d'autres exemples de systèmes d'intégration de données qui fournissent un accès à l'information obtenue et synthétisée à partir de sources Webs, lesquelles dissimulent généralement une grande quantité d'informations.

Avec le développement de ce type de systèmes , la qualité de l'information est devenue une propriété de premier plan de plus en plus demandée par les utilisateurs. Comme le

potentiel de la récupération des données augmente, les utilisateurs sont de plus en plus préoccupés par la qualité des données [1]. Diverses études ont montré l'importance de la qualité des données pour l'utilisateur, en particulier, lorsqu'on interroge des sources de données distribuées et autonomes, dont les données sont hétérogènes [1][2][3].

Assurer une bonne qualité des données fournies à l'utilisateur est un important défi, qui est lié au succès des systèmes d'information. Par exemple dans le cas de systèmes d'intégration de données, le problème est particulièrement complexe car les données sont fournies par diverses sources, lesquelles ont probablement une qualité des données différente. A cause du grand nombre de sources de données et de leurs diversités en plus de leurs autonomies, il est important d'avoir une bonne connaissance de leurs qualités et de la prendre en considération durant le processus d'intégration. Résoudre des problèmes de qualité de données, permet d'envisager la production de données comme n'importe quel autre processus de production.

Motivations et problèmes

Chaque domaine d'application a sa propre vision de la qualité des données ainsi qu'une suite de solutions pour résoudre ses problèmes liés à la qualité. Cependant il ya un besoin croissant de réutilisation des connaissances de qualité et des méthodes de mesure qui ont été définies, et aussi d'avoir un modèle générique d'évaluation de la qualité, que l'on peut réutiliser et personnaliser suivant le domaine d'application.

La gestion de la qualité des données est un problème clé au sein de toutes les organisations qu'elles soient privées ou publiques. Une large collection d'outils commerciaux et open source est proposée pour gérer les problèmes de qualité des données dans les systèmes d'information. Ces outils supportent un grand nombre de source de données (Base de données relationnel, fichier XML, fichier texte, etc) accessibles de plusieurs façons (JDBC, ODBC, ADO.NET, etc) et fournissent différents types de fonctionnalités (mesure, analyse, amélioration des données, etc). DataCleaner [4], DQguru [5], Talend Open Profiler [6] sont des exemples de ce type d'outils.

En général, les outils de qualité ne fournissent pas de mécanismes pour invoquer leurs fonctionnalités à partir d'application trois tiers. En plus, chaque outil gère des concepts de qualité différents, à différents niveaux d'abstraction exprimés avec des technologies ad-hoc. Ces limites génèrent un écart important entre les besoins des utilisateurs en termes de qualité (qui sont généralement complexes et qui combinent un grand nombre d'indicateurs de qualité) et les indicateurs de qualité qui peuvent être efficacement calculés à partir d'outils externes. De nombreuses organisations expérimentent le problème d'avoir plusieurs outils de qualité sophistiqués mais conduisant à la gestion manuelle des résultats. Les limites citées mettent en évidence l'importance d'avoir une plate-forme de gestion de la qualité qui gère un catalogue de concept de qualité et qui permet d'intégrer des outils de qualité existants.

Objectifs

Ce stage de master est défini dans le cadre du projet *ANR QUADRIS[7] (Qualité des Données et des Systèmes d'Informations Multi-Sources)* dont les objectifs sont de :

– Proposer un métamodele pour l'évaluation de la qualité.
– Proposer un métamodele qui permet de raisonner en terme de qualité.
– Proposer un métamodele extensible et réutilisable.
– Permettre l'integration d'outils de qualité existants.
– Fournir un environnement interactif pour exécuter des mesures de qualité et analyser les résultats obtenus. .

Dans ce stage de Master, on s'est intéressé essentiellement à la qualité des données et aux multiples dimensions qui permettent de caractériser cette qualité. Nous présentons entre autres un système de médiation à base de services, qui intègre les fonctionnalités de plusieurs outils de qualité. Ce système s'appuie sur un prototype de recherche déjà existant appelé *QBox*[1], dont nous avons amélioré et étendu les fonctionnalités. Entre autre, ce document présente :

1. une étude bibliographique sur la qualité des données en général.
2. une analyse critique du prototype existant.
3. la spécification de nouvelles fonctionnalités.
4. la présentation du nouveau prototype qui intègre ces nouvelles fonctionnalités.

Organisation du rapport

Le présent rapport est organisé en quatre chapitres, correspondant aux étapes de notre démarche méthodologique.

– Le premier chapitre présente une étude bibliographique qui porte sur la qualité des données en général, et les multiples dimensions et facteurs qui permettent de caractériser, d'évaluer et de classifier les données livrées à l'utilisateur. Il présente aussi la plate-forme existante appelée *QBox-Foundation*, un métamodele d'évaluation de la qualité.
– Le deuxième chapitre présente en détail la nouvelle plate-forme *QBox-Services*, une architecture de médiation de services qui résout les problèmes d'évolution de la *QBox-Foundation*.
– Le troisième chapitre présente en détail le prototype développé, ses composants et les technologies utilisées pour les développer.
– Dans le dernier chapitre nous proposons un scénario d'exécution sur des données réelles et en exploitation, lesquelles permettent de mettre en application le prototype développé.
– Enfin, nous conclurons en évoquant de nouvelles perspectives de recherches.

1. La QBox a été définie dans le cadre du projet ANR *Quadris* http ://deptinfo.cnam.fr/xwiki/bin/view/QUADRIS/

Chapitre 1

État de l'art

Ce chapitre résume l'étude bibliographique effectuée. Il porte sur la qualité des données en général et les multiples dimensions qui permettent de caractériser, d'évaluer et de classifier les données livrées à l'utilisateur. Il présente aussi l'architecture existante appelée QBox-Foundation.

1.1 Introduction

Les besoins d'accéder de façon uniforme à des sources de données multiples sont chaque jour plus forts, particulièrement dans les systèmes décisionnels qui ont besoin d'une analyse compréhensive des données. Avec le développement des Systèmes d'Intégration de Données (SID), la qualité de l'information est devenue une propriété de premier plan de plus en plus exigée par les utilisateurs.

La qualité des données en général se réfère à la conformité des données aux usages prévus, dans les modes opératoires, les processus, les prises de décision et la planification. De même, les données sont jugées de grande qualité si elles représentent correctement le mode de fabrication auquel elles se réfèrent. Ces deux points de vue peuvent souvent entrer en contradiction, y compris lorsqu'un même ensemble de données est utilisé avec un objectif commun.

1.2 Historique

La plupart des technologies sur les données informatiques sont nées du désir d'envoyer des informations par courrier. Avant l'émergence de serveurs bon marché, les mainframes étaient utilisés pour mettre à jour les données (noms, adresses, et autres attributs) afin que les courriers électroniques arrivent correctement à leur destination. Les mainframes utilisaient des règles métiers pour corriger les défauts dans les données (fautes sur les champs nom et date, défauts de structuration), ainsi que pour suivre les clients qui avaient changé d'adresse, disparu, fusionné, ou expérimenté d'autres événements.

Aux États-Unis, les agences de gouvernement commencèrent à mettre à la disposition de quelques sociétés de service des données postales pour gérer les entreprises selon le

registre de changement d'adresse national (NCOA). Cette technologie a fait économiser à de grandes entreprises de grandes sommes d'argent (millions de dollars) en comparaison à la gestion manuelle et individuelle des données client. Elles ont réduit leurs coûts d'affranchissement en rendant plus précises les coordonnées de leurs clients.

Bien que la plupart des entreprises pensent au nom et à l'adresse quand elles se préoccupent de la qualité des données, on reconnaît aujourd'hui que la qualité des données est la façon d'améliorer tous les types de données, comme les données sur la chaîne logistique, les données des progiciels de gestion intégrée, les données transactionnelles, etc.

Alors que les données sur les noms et adresses ont un standard clair avec les définitions des autorités postales, les autres types de données ont peu de standards reconnus. Il y a une tendance de fond aujourd'hui dans l'industrie pour standardiser certaines données qui ne sont pas des adresses. Le groupe GS1 [2] *The global language of business* fait partie des groupes qui sont fers de lance dans ce mouvement.

1.3 Causes des problèmes liés à la qualité des données

L'une des plus grandes causes qui rend difficile la correction des problèmes de qualité des données est l'incompréhension de la nature et de la cause de ces problèmes. Trop souvent les organisations conceptualisent la qualité des données comme étant un problème système, technique ou de processus, alors qu'en réalité ce ne sont pas les seuls facteurs. Le problème de qualité des données est non seulement un problème technologique mais aussi un problème de comportement.

Les problèmes technologiques sont souvent liés aux processus qui manipulent les données, et qui peuvent affecter la qualité à un certain degré. Les systèmes d'intégration de données sont des exemples de systèmes qui peuvent altérer la qualité des données qu'ils intègrent. Ces données peuvent être dupliquées, conflictuelles, incohérentes (ne respectent pas les regèles d'intégrité), nulles, etc.

Que signifie un problème de comportement ? Tout simplement, que les données sont le résultat des actions de l'utilisateur car elles sont saisies, utilisées, et mises à jour par l'utilisateur. Elles sont peut être stockées et manipulées par le système, mais à un certain moment elles interagissent avec et sont influencées par l'utilisateur. Ainsi les utilisateurs ont la possibilité et la responsabilité d'assurer la qualité des données. Aussi, étant donné que la qualité des données implique les actions de l'utilisateur *(telles que la garantie de la pertinence et de la crédibilité des données)*, on peut dire qu'elle est dépendante de l'utilisateur et n'est pas simplement un problème technologique.

Il existe beaucoup de facteurs qui contribuent aux problèmes de comportement et qui propagent des données erronées au sein d'une organisation. Beaucoup de facteurs sont organisationnels tandis que d'autres sont environnementaux. Par exemple l'incapacité de traiter les problèmes des données en raison : du manque de temps, des priorités concurrentes, de l'impossibilité d'accéder à des informations correctes et précises, du manque d'autorité, ou de l'indisponibilité du support administratif. Toutes ces causes peuvent

2. GS1 (The global language of business) est une organisation mondiale dédiée à la conception et l'application de normes standard http ://www.gs1.org/.

empêcher l'utilisateur d'améliorer la qualité de ses données. En outre, des questions telles que la motivation, le désir de se concentrer sur plus d'un travail ou le dégoût de corriger les problèmes crées par des collègues peuvent aussi pousser les utilisateurs à ignorer les problèmes de qualité des données.

1.4 Dimensions de la qualité des données

Traditionnellement la qualité des données est décrite ou caractérisée par plusieurs attributs ou facteurs qui peuvent aider à classer les données livrées à l'utilisateur *(ex : la fraîcheur, l'exactitude, la complétude, etc)* ou bien le processus qui manipule ces données *(ex : temps de réponse, fiabilité, sécurité, etc)*[8]. Beaucoup de listes de facteurs de qualité ont été proposées, quelque unes contiennent un bon nombre de définitions. Par exemple *Redman* a identifié quatre dimensions de qualité : *l'exactitude, la complétude, la crédibilité et la consistance* [9] alors que *Wang* et *Strong* ont analysé les attributs de qualité par rapport au point de vue de l'utilisateur [1].

Chaque dimension de qualité est composée de plusieurs facteurs ; chaque facteur traite un problème particulier ou un type de système particulier. Par exemple l'exactitude des données est une dimension de qualité qui comprend la correction sémantique, la correction syntaxique et la précision des données [8]. Chaque facteur de qualité est composé a son tour de plusieurs métriques ; chaque métrique représente l'instrument utilisé pour mesurer un certain facteur de qualité ; ex : le pourcentage des données du système d'information qui matchent les données du monde réel est une métrique qui mesure la correction sémantique.

Parmi les dimensions de qualité qui ont été proposées nous allons décrire quelques unes dans les sections qui suivent.

1.4.1 La fraîcheur des données

Le concept de la fraîcheur des données introduit l'idée de l'age qu'ont les données : sont elles suffisamment fraîches par rapport à l'attente de l'utilisateur ? Est ce que cette source de données a les données les plus récentes ? Quand ces données ont été produites ?[8]. La fraîcheur des données est décrite comme l'une des dimensions les plus importantes dans la qualité des données [1]. Pour cette dimension de qualité on distingue deux facteurs :

1.4.1.1 Facteur d'actualité

Ce facteur exprime à quel point les données sont dépassées par rapport à la ressource. Pour la mesure de ce facteur trois métriques sont définies :

1. **Currency** Cette métrique mesure le temps passé depuis que les données ont été extraites de la source *(la différence entre le temps de livraison et le temps d'extraction de la donnée)*.
2. **Taux de fraîcheur** Cette métrique mesure le pourcentage de tuples d'une vue qui sont mis à jour *(qui n'ont pas été mis à jour depuis leurs extraction)*.
3. **Obsolescence** Cette métrique mesure le nombre d'opérations/transactions de mises à jour d'une source depuis l'extraction des données.

1.4.1.2 Facteur d'age

Ce facteur exprime l'age des données *(depuis leur création/mise à jour dans la source)*. Il peut être mesuré avec la métrique suivante :

– **Age** Cette métrique mesure le temps passé depuis la dernière mise à jour dans une source *(la différence entre le temps de livraison de la donnée et de sa mise à jour)*.

La figure 1.1 illustre la différence entre ces deux facteurs

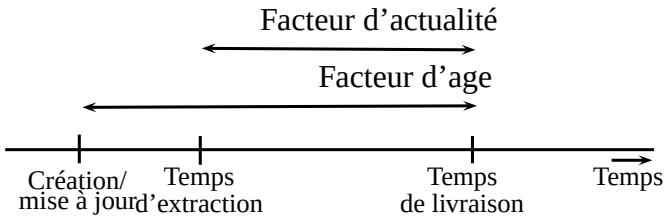

FIGURE 1.1 – Facteurs d'age et d'actualité

1.4.2 L'exactitude des données

La plupart des études sur la qualité des données traite l'exactitude des données comme étant l'une des dimensions clé pour différents types de système d'intégration de données *DIS* [8]. Cependant il n'existe pas de définition commune entre toutes les recherches effectuées sur l'exactitude des données.

Nous allons la définir comme étant liée à l'exactitude, l'impureté et à la précision avec laquelle les données du monde réel sont représentées dans un système d'information [8]. Intuitivement, le concept d'exactitude des données introduit l'idée de comment ou à quel point la donnée est précise, valide ou saine : Est ce que les données correspondent au monde réel ? Est ce que les données sont saines *(pures)* ? Est ce que ces données tolèrent des erreurs ? Est ce que les données sont suffisamment précises par rapport à l'attente de l'utilisateur ? Est ce que le niveau de détail est adéquat à une tâche ?

L'exactitude des données comprend une famille de facteurs de qualité, chacun d'eux représente un aspect de l'exactitude par rapport à un système d'intégration particulier. Chaque facteur possède ses propres métriques. Parmi les facteurs de cette dimension nous citons :

1.4.2.1 Facteur de correction sémantique

Le concept de correction sémantique inclut une comparaison entre les données du système, et le monde réel ou une référence de données reconnue comme étant correcte. Ce facteur représente l'écart *(ou la distance sémantique)* qui existe entre les données qui sont dans le système et les données d'une référence externe, et est calculé avec les métriques suivantes :

1. **Booléen de correction sémantique** Cette métrique renvoie un booléen qui indique si une donnée correspond au monde réel par rapport à une référence externe.

2. **Taux de valeur inexacte** Cette métrique mesure le pourcentage des données du système qui contiennent des erreurs ou des valeurs *nulles*.
3. **Taux de mismembership** Cette métrique mesure le pourcentage des données qui n'ont pas de correspondance avec le monde réel.
4. **Taux de correction sémantique** Cette métrique mesure le pourcentage de données sémantiquement correctes dans le système. Ce pourcentage peut être calculé en divisant le nombre des données du système qui matche avec les données du monde réel par le nombre des données du système.
5. **Écart de la correction sémantique** Cette métrique mesure le pourcentage des données du système qui contiennent des erreurs ou des valeurs *nulles* dans un attribut. Le calcul de cette distance dépend du type de la donnée et de l'application. Ex : pour les données numériques, cela peut être la différence entre les valeurs (normalisées ou pas).
6. **Degré de correction sémantique** Cette métrique représente la confiance du degré de correction des données. Ce calcul peut être fait manuellement par un expert ou estimé en se basant sur des données historiques ou statistiques.

1.4.2.2 Facteur de correction syntaxique

Le concept de correction syntaxique exprime le degré d'impureté d'un ensemble de données par rapport aux erreurs syntaxiques *(comme les erreurs d'orthographe ou de format)*. Une donnée est jugé correcte syntaxiquement, si elle satisfait des règles syntaxiques et des contraintes imposées par l'utilisateur *(contrainte de format)*. Ce facteur peut être calculé avec :

1. **Taux correction syntaxique** Cette métrique mesure le pourcentage des données du système d'information qui sont conformes à des règles syntaxiques.
2. **Écart de correction syntaxique** Cette métrique mesure la distance qui existe entre une donnée du système et une référence considérée comme étant syntaxiquement correcte.

1.4.2.3 Facteur de précision

La précision des données implique le niveau de détail de la représentation des données. Ce niveau peut être calculé avec les métriques suivantes :

1. **Échelle** Cette métrique représente la précision associée à l'échelle de mesure.
2. **Erreur standard** Cette métrique représente l'écart type d'un ensemble de mesures.
3. **Granularité** Cette métrique représente le nombre et la portée des attributs qui sont utilisés pour représenter un concept. Ex : une adresse peut être représentée par seulement le nom du pays ou bien par un ensemble d'attributs comme le nom de la cité, de la rue, numéro de porte, code postal et nom du pays. Le second type de représentation fournit beaucoup plus de détails. Un exemple de métrique est le nombre de valeurs *(non nulles)* qui représentent un concept.

1.4.3 La complétude des données

Généralement, dans la littérature un ensemble de données est dit complet si toutes les données qui lui sont nécessaire sont inclues : *"toutes les valeurs pour une certaine variable sont inclues"*. La complétude peut être vue comme étant l'habilité d'un système d'information à représenter chaque étape significative du monde réel[10]. Ainsi la complétude n'est pas liée à des données relatives à des concepts comme les attributs, les variables ou les valeurs. Une définition basée sur les états pour définir la complétude fournit une vision plus générale qu'une définition basée sur les données. En particulier, cela s'applique plus sur des combinaisons de données plutôt que sur des valeurs nulles. Cela permet aussi aux attributs d'être obligatoires ou facultatifs selon les valeurs des autres attributs.

On peut dire aussi que c'est le degré des données d'un domaine particulier enregistré dans un système d'information, et cela implique aussi que tous les faits relatifs au monde réel soient représentés dans le système d'information. On distingue deux facteurs par rapport à la complétude :

1.4.3.1 Densité

Ce facteur indique si toute les valeurs pour un attribut spécifique sont présentes *(et non nulles)*. Il peut être mesuré avec la métrique suivante :

- **Taux de densité** Cette métrique mesure le taux des données dupliquées (données qui représentent la même entité du monde réel).

1.4.3.2 Couverture ou portée

Ce facteur indique si toutes les entités requises pour une classe sont présentes ou pas. Il peut être mesuré avec la métrique suivante :

- **Taux de couverture** Cette métrique mesure le pourcentage des entités du monde réel représentées dans une source.

1.4.4 La consistance des données

Dans la littérature, la consistance fait référence à plusieurs aspects des données. En particulier, à la valeur de la donnée, à la représentation des données, et à la représentation physique des données. Dans ce contexte nous allons nous intéresser à la valeur de la donnée. En clair une valeur pour une donnée doit toujours être la même pour une même situation [10]. La consistance des données exprime aussi le degré auquel un ensemble de données satisfait un ensemble de contraintes d'intégrité. Nous ne citons qu'un seul facteur pour cette dimension.

1.4.4.1 Consistance

Le facteur de consistance vérifie si une donnée est consistante ou pas. Une donnée est dite consistante si elle satisfait ses contraintes. La plus commune des contraintes consiste à vérifier les valeurs *nulles*, l'unicité de la clé primaire et les dépendances fonctionnelles. Une des métriques qui peut mesurer ce facteur est :

- **Taux de consistance** Cette métrique mesure le pourcentage des données du système qui satisfont les contraintes d'intégrité.

1.5 Goal-Question-Metric (GQM)

Le processus d'évaluation de la qualité fonctionne traditionnellement de façon *top-down* c'est à dire qu'on commence l'évaluation à partir d'un niveau général (abstrait) puis on entame un processus de raffinement petit à petit.

La plate-forme *QBox-Foundation* est basée sur le paradigme *Goal-Question-Metric (GQM)* [11], lequel est une approche d'analyse *top-down* qui aide à déterminer les métriques qu'une organisation veut ou doit mettre en place afin de répondre à ses objectifs.

Goal Question Metric abrégé *GQM* est une approche basée sur l'hypothèse que, pour qu'une organisation puisse effectuer des mesures dans le cadre de buts précis, elle doit d'abord spécifier ses buts ; puis tracer un chemin allant de ses buts vers les données qui définissent ces buts, et enfin produire une plate-forme pour l'interprétation de ces données par rapport à ses objectifs fixés [12]. Cette approche propose trois niveaux d'abstraction :

1.5.1 Niveau conceptuel (Goal)

Un but est défini pour un objet, pour plusieurs raisons, par rapport à plusieurs modèles de qualité, depuis plusieurs points de vue et est relatif à un environnement particulier. Ce niveau exprime des buts de qualité à haut niveau. *Ex. améliorer la qualité des coordonnées des étudiants.*

1.5.2 Niveau opérationnel (Question)

Un ensemble de questions est utilisé pour caractériser la manière d'évaluer un but spécifique. La question va essayer de caractériser un objet à mesurer et essayer de déterminer sa qualité par rapport à un point de vue précis. *Ex. est ce que les adresses des étudiants sont correctement écrites ?*

1.5.3 Niveau quantitatif (Metric)

Un ensemble de données est associé à chaque question pour lui apporter une réponse dans un but quantitatif. Entre autre une métrique constitue une façon quantitative pour répondre à une question spécifique. *Ex. le taux d'erreurs syntaxiques.*

En résumé l'approche *GQM* est une structure hiérarchique (voir la figure 1.2) qui commence par déterminer un but spécifique, qui est raffiné en un ensemble de questions et chaque question est raffinée à son tour en plusieurs métriques. Une même métrique peut être utilisée pour répondre à différentes questions pour un même but.

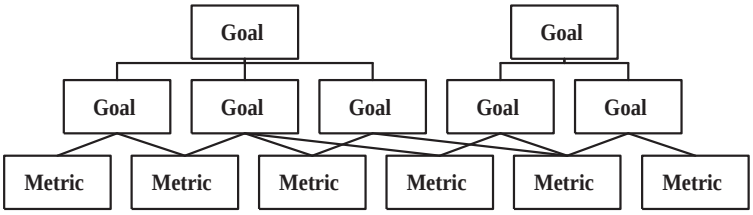

FIGURE 1.2 – Structure hiérarchique du modèle GQM

1.6 QBox-Foundation

La *QBox-Foundation* est une plate-forme pour l'évaluation de la qualité basée sur un métamodele qui est une amélioration du modèle DWQ [13], qui est à son tour, une extension du modèle GQM décrit dans la section 1.5. Cette plate-forme permet de définir des buts de qualité de haut niveau et une spécialisation des méthodes de mesure par rapport à ces buts de qualité [11]. Elle propose entre autre :

1. Un métamodele d'évaluation de la qualité *(Quality Assessment Metamodel QAM)* pour raisonner avec le concept de qualité.
2. Une collection extensible et réutilisable de métriques de qualité et de méthodes de mesure.
3. Un environnement interactif qui permet d'instancier des métriques de qualité et des méthodes de mesures pour répondre à des buts et des questions spécifiques.
4. Une interface agréable et conviviale qui permet de gérer les différentes entités du métamodele et d'exécuter les méthodes de mesures spécialisées.

La figure 1.3 résume le métamodele de la *QBox-Foundation*.

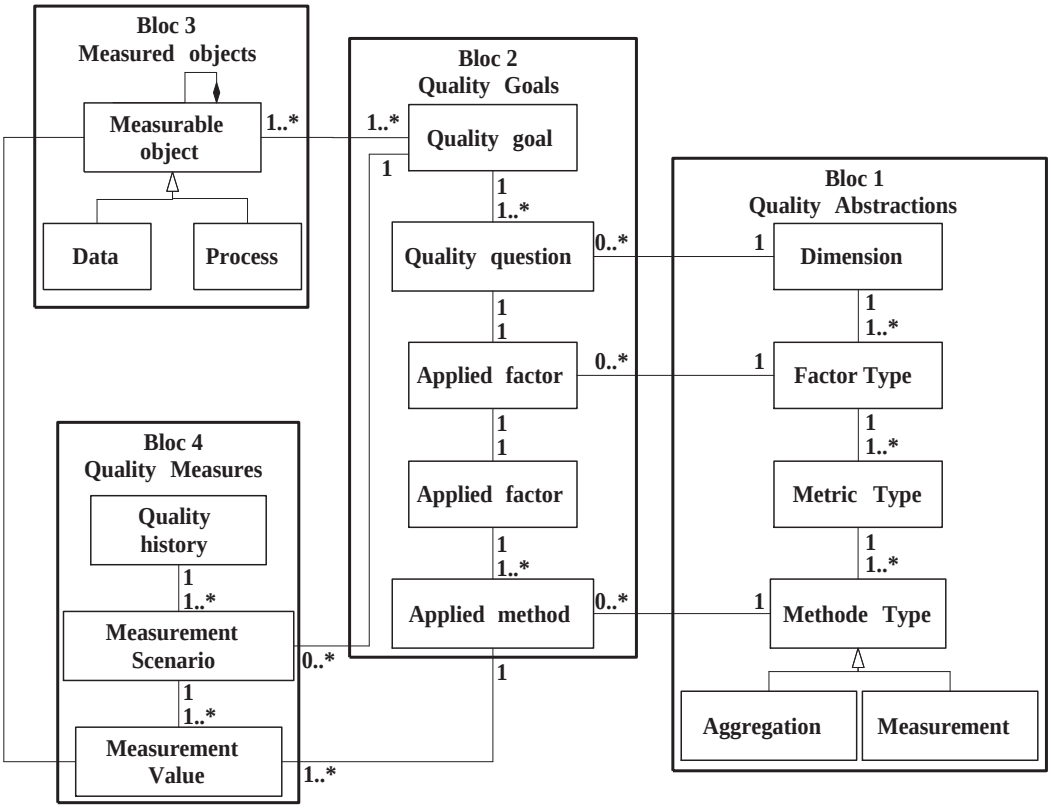

FIGURE 1.3 – Métamodele d'évaluation de la qualité

1. Le premier bloc du métamodele représente une librairie de type abstrait (voir section 1.4) qui permet de caractériser un but spécifique. Ce bloc est composé de :

- *Dimensions de qualité :* la qualité d'information est caractérisée généralement par plusieurs dimensions qui permettent de classer les données. Une dimension représente une facette de haut niveau de qualité.
- *Facteurs de qualité :* représentent un aspect particulier d'une dimension, une dimension peut avoir plusieurs facteurs.*Ex. la dimension d'exactitude des données inclut la correction sémantique, la correction syntaxique et la précision des données [8].*
- *Métriques de qualité :* est un instrument utilisé pour mesurer un certain facteur de qualité. Il peut y avoir plusieurs métriques pour mesurer le même facteur.*Ex. le pourcentage des données système qui matchent le monde réel, est une métrique pour la correction sémantique.*
- *Méthode de qualité :* c'est un process qui implémente une métrique de qualité. On distingue deux types de méthodes ; les méthodes de mesure et les méthodes d'agrégation. Il peut y avoir plusieurs méthodes pour implémenter une même métrique.

Dans le but d'adapter des concepts de qualité à un domaine d'application spécifique, les facteurs de qualité peuvent être spécialisés en ajoutant un aspect sémantique pour répondre au mieux aux questions de qualité *(Ex. le facteur correction syntaxique peut être spécialisé en facteur correction syntaxique d'adresse)*. Les métriques et méthodes de qualité peuvent aussi être spécialisées pour accéder aux objets du système correspondant. *Les facteurs, métriques, méthodes adaptés* représentent les concepts de qualité adaptés.

2. Le deuxième bloc représente l'approche GQM (voir section 1.5) avec un raffinement spécifique des métriques, en prenant en considération l'abstraction introduite dans le premier bloc.
   - *Niveau Goals :* représente un haut niveau de besoin en terme de qualité.
   - *Niveau Questions :* représente le raffinement ou la décomposition d'un goal. Un but peut être caractérisé par plusieurs questions.
   - *Niveau Métriques :* dans cette approche ce niveau est raffiné en trois sous-niveaux associé à la hiérarchie introduite dans le premier bloc.
3. Le troisième bloc correspond aux objets du système d'information qui peuvent être mesurés et qui sont associés à un but. Les objets peuvent être soit des objets de données *(Data source, table, vue, attribut, Colonne, etc)* soit des processus *(Package, Processus, Activité/Tache, Opération, etc)*.
4. Le quatrième bloc correspond aux mesures. Chaque mesure d'un but est appelée mesure de scénario, et est composée de l'ensemble des valeurs associées à un ensemble de questions qui définissent ce but. Le résultat de l'exécution de plusieurs scénarios de mesure est appelé un historique de mesure et peut servir à l'analyse du comportement des objets mesurés.

1.7 Limites de la QBox-Foundation

Les limites de cette plate-forme concernent principalement son évolution : incomplétude des méthodes de qualité et le manque de connectivité pour intégrer les fonctionnalités d'outils existants. D'une part, développer pour la *QBox-Foundation* toutes les méthodes

de qualité peut prendre beaucoup de temps, être très coûteux, et le résultat peut ne pas être efficace par rapport aux attentes de l'utilisateur. D'autre part, les outils de qualité de données existants fournissent des méthodes génériques qui peuvent être appliquées sur différentes sources de données. Il est intéressant d'exploiter le potentiel qu'offrent ces outils en les intégrant dans notre plate-forme. Enfin, la *QBox-Foundation* ne propose pas une approche spécialement adaptée à l'exploration et à l'analyse des mesures effectuées.

La technologie des services web permet d'intégrer ces outils de qualité, d'où la *QBox-services* qui est présentée dans le chapitre suivant.

1.8 Conclusion

Le but de ce chapitre est de donner un aperçu global sur la qualité des données, les causes des problèmes liés à la qualité des données, comment des données erronées se propagent dans les systèmes d'information et comment la qualité des données est vue par les chercheurs. On a vu entre autre une façon qui permet de caractériser la qualité des données via les différents dimensions et facteurs qui sont définis. Actuellement il n'existe pas de consensus pour la définition des dimensions de qualité mais on a essayé de donner une vue d'ensemble de ces dimensions. On a aussi présenté l'architecture existante de la *QBox-Foundation* qui est un métamodele pour l'évaluation de la qualité des données [11]. On a par ailleurs pointé les limites de cette dernière.

Le chapitre suivant décrit en détail la solution proposée aux limites de la *QBox-foundation*, en l'occurrence la *QBox-Services* une plate-forme de médiation de services. Nous présenterons aussi quelques techniques qui nous permettent de normaliser les données mesurées (traitement des *outliers* et rendre leurs modalités dans un intervalle de [0,1]) à des fins d'analyse, de projection et de comparaison.

Chapitre 2

QBox Services

Dans ce chapitre nous allons tout d'abord, présenter la technologie qui permet de mettre en oeuvre la solution apportée aux limites de la QBox-Foundation. Ensuite nous verrons en détail la nouvelle plate-forme QBox-Services, l'architecture qui résout les problemes d'évolution de la QBox-Foundation. Nous présenterons aussi des techniques qui permettent de normaliser les données mesurées (traitement des *outliers* et rendre leurs modalités dans un intervalle de [0,1]) à des fins d'analyse.

2.1 Introduction

Les architectures orientées service (SOA pour Service Oriented Architecture) sont destinées à faire face aux types de demandes d'évolution citées dans la section 1.7. Elles s'inscrivent dans l'ensemble des solutions destinées à assurer l'intégration et la manipulation des différentes briques et composants applicatifs d'un système informatique et de gérer les liens qu'ils entretiennent. Comme son nom l'indique, cette approche repose sur la réorganisation des applications en ensembles fonctionnels appelés services.

2.2 Architecture orientée services

Les architectures orientées services (SOA) présentent une approche qui permet de construire des systèmes distribués, permettant de fournir les fonctionnalités d'une application sous forme de services (soit à l'utilisateur soit à une autre application). La collaboration dans une architecture orientée services obéit au paradigme *Find, Bind and invoke (trouver, lier et invoquer)*. Le consommateur de services cherche dynamiquement des services spécifiques répondant à des besoins particuliers en interrogeant un annuaire qui les répertorie. Si le services est trouvé, l'annuaire de services fournit au consommateur une interface et un point d'accès qui permet de l'invoquer. La figure 2.1 présente le schéma d'une architecture orientée services.

Il n'existe pas à proprement parler de spécifications officielles d'une architecture SOA, néanmoins les principales notions fédératrices que l'on retrouve dans une telle architecture sont les suivantes :

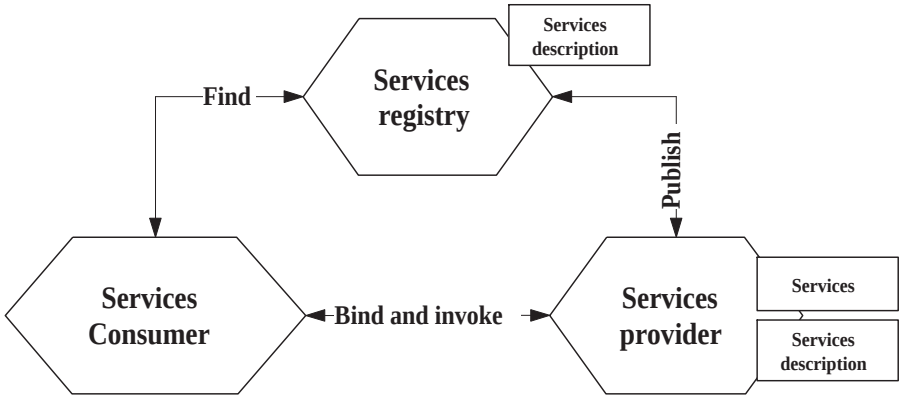

FIGURE 2.1 – Collaboration des entités de l'architecture SOA

- La notion de service, c'est-à-dire une fonction encapsulée dans un composant que l'on peut interroger à l'aide d'une requête composée d'un ou plusieurs paramètres et fournissant une ou plusieurs réponses. Idéalement chaque service doit être indépendant des autres afin de garantir sa réutilisabilité et son interopérabilité.
- La description du service, consistant à décrire les paramètres d'entrée du service et le format et le type des données retournées. Le principal format de description de services est WSDL (Web Services Description Language), normalisé par le W3C.
- La publication (en anglais advertising) et la découverte (discovery) des services. La publication consiste à publier dans un registre (en anglais registry ou repository) les services disponibles aux utilisateurs, tandis que la notion de découverte recouvre la possibilité de rechercher un service parmi ceux qui ont été publiés. Le principal standard utilisé est UDDI (Universal Description Discovery and Integration), normalisé par l'OASIS.
- L'invocation, représentant la connexion et l'interaction du client avec le service. Le principal protocole utilisé pour l'invocation de services est SOAP (Simple Object Access Protocol).

2.3 QBox-Services

La *QBox-services* est une architecture à base de services qui permet de régler le problème d'évolution de la *QBox-Foundation*. En effet, les méthodes de mesure sont vues comme des services de qualité abstraits et une liaison est établie entre ces services de qualité abstraits et une implémentation spécifique dans un outil externe. Cette architecture joue le rôle de médiateur entre l'analyste dont la vue est le métamodele de qualité et ces services de qualité. Le coeur de la *QBox-Foundation* est un métamodele d'évaluation de qualité qui permet de raisonner en terme de qualité. Ce métamodele est légèrement modifié pour extérioriser les méthodes de mesure afin qu'elles puissent être invoquées comme des services.

Le nouveau métamodele est présenté dans la figure 2.2.

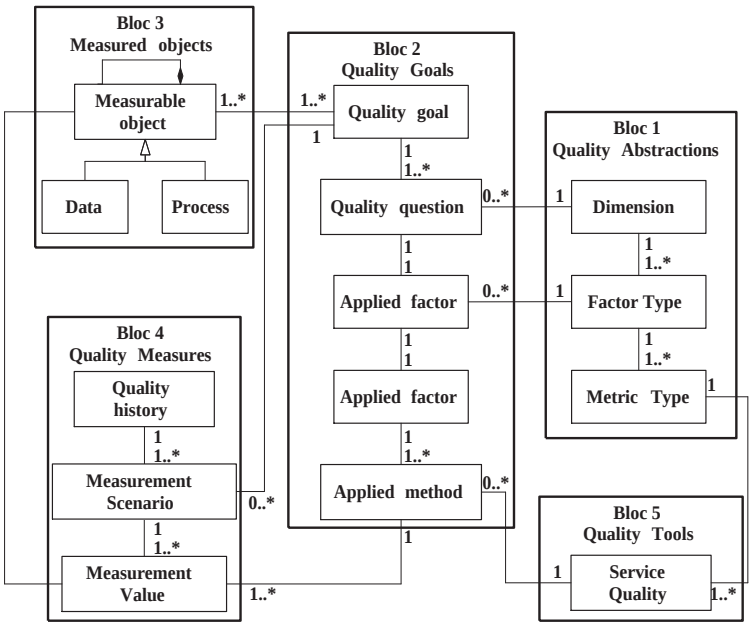

FIGURE 2.2 – Métamodele d'évaluation de la qualité de la QBox-Services

Les quatre premiers blocs sont décries dans la section 1.6. Le cinquième bloc du nouveau métamodele constitue une librairie d'outils de qualité. Dans la *QBox-Foundation* cette librairie était inclue dans le premier bloc qui est une librairie de type abstrait. Dans cette version améliorée, elle est extériorisée dans le but de pouvoir intégrer une large collection d'outils externes, qui sont répertoriés dans un annuaire de services. Cette collection inclut des outils de mesure, d'analyse et d'amélioration des données.

La *QBox-Services* fournit aussi une approche d'analyse multidimensionnelle des mesures de la qualité des données appelée *qOLAP (Quality On-Line Analytical Processing)*[14].

2.4 Architecture de la QBox-Services

Nous allons maintenant décrire l'architecture de la *QBox-Services*. Nous allons commencer par décrire les principaux composants de l'architecture et la manière dont ils sont reliés et interagissent entre eux. Dans la section suivante nous allons décrire la partie médiation de services de qualité *(découverte et invocation de services de qualité)* en détail.

Comme il a été dit précédemment, la *QBox-Services* conserve toutes les fonctionnalités de la *QBox-Foundation*, sauf que les méthodes de mesures ont été extériorisées pour être invoquées comme des services. Ceci permet entre autre l'utilisation d'outils d'évaluation de qualité externe et par conséquent, rendre la *QBox-services* ouverte pour l'intégration d'outils de qualité existants. Cette plate-forme est composée de trois principaux composants *QBox-Foundation, QManagement et QMediator*, qui sont représentés dans la figure 2.3.

Le composant *QBox-Foundation* offre les fonctionnalités qui permettent de définir des objectifs de qualité *(goal)*, des facteurs et des métriques de qualité raffinés et spécialisés pour répondre au mieux aux besoins de ces objectifs. Ce processus de raffinement et de spécialisation permet de dériver du modèle de qualité générique, un modèle de qualité personnalisé abrégé *PQM (Personalized Quality Model)*. On dira donc qu'un *PQM* est un ensemble de facteurs et de métriques personnalisés qui correspondent à un but spécifique et qui fait référence à un objet du système d'information *(Data source, table, vue, attribut, Colonne, etc)*. Ce composant offre aussi une interface graphique à l'utilisateur pour exécuter les requêtes, afficher les résultats obtenus et les analyser par rapport à plusieurs dimensions *(Temps, Objet, etc)*[14].

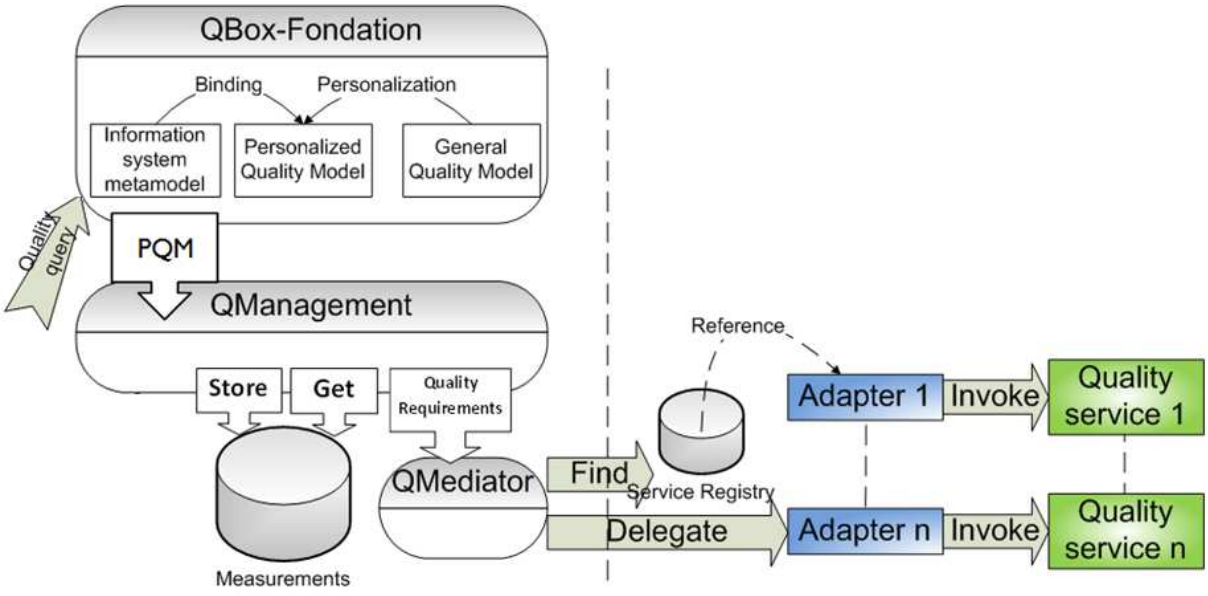

FIGURE 2.3 – Architecture de la QBox-Services

Pour répondre aux besoins d'un *PQM*, des services de qualité doivent être exécutés. Ces services peuvent être de trois types différents : *des services de mesure* qui calculent des métriques de qualité *(service qui calcule le pourcentage des valeurs nulles)*, *des services d'analyse* qui analysent un ensemble de mesure et calculent des indicateurs complexes *(service qui analyse le comportement de la dimension de fraîcheur de données)* ou *des services d'amélioration de données* qui améliorent la qualité des données *(service qui élimine les doublant)*.

Le composant *QMediator* fournit les fonctionnalités qui nous permettent de trouver des services de qualité *(publié dans un annuaire de services "Service Registry")* qui sont appropriés à des besoins spécifiques exprimés en PQM, et nous permet aussi d'exécuter ces services et de retourner les résultats obtenus.

Le composant *QManagement* exécute le *PQM*. Un service de qualité peut être exécuté soit périodiquement, soit invoqué par l'utilisateur ou bien par un autre service. L'analyse des résultats obtenus est effectuée par une requête de qualité appelée *QOLAP*[3]. Les résultats obtenus sont sauvegardés d'une façon multidimensionnelle. Si une requête est

3. Voir QOLAP

posée sur une mesure qui n'est pas disponible, le service correspondant peut être exécuté pour l'obtenir.

2.5 Médiation de services de qualité

QMediator joue le rôle de médiateur entre les besoins en terme de qualité et les services de qualité répertoriés dans l'annuaire de services. Les fonctionnalités d'un service de qualité sont décrites dans l'annuaire de services comme étant un service de qualité abstrait et nous proposons un mécanisme qui permet de lier ce service de qualité abstrait à son implémentation dans un outil externe. Dans ce qui suit nous allons décrire en détail les composants qui participent à la partie médiation de services (voir la figure 2.4) à savoir : *Les outils de qualité, les services de qualité abstraits, l'annuaire de services, le gestionnaire d'annuaire de services, le sélectionneur de services, les adaptateurs d'objets, le gestionnaire de patterns d'accès, les adaptateurs et les services de qualité.*

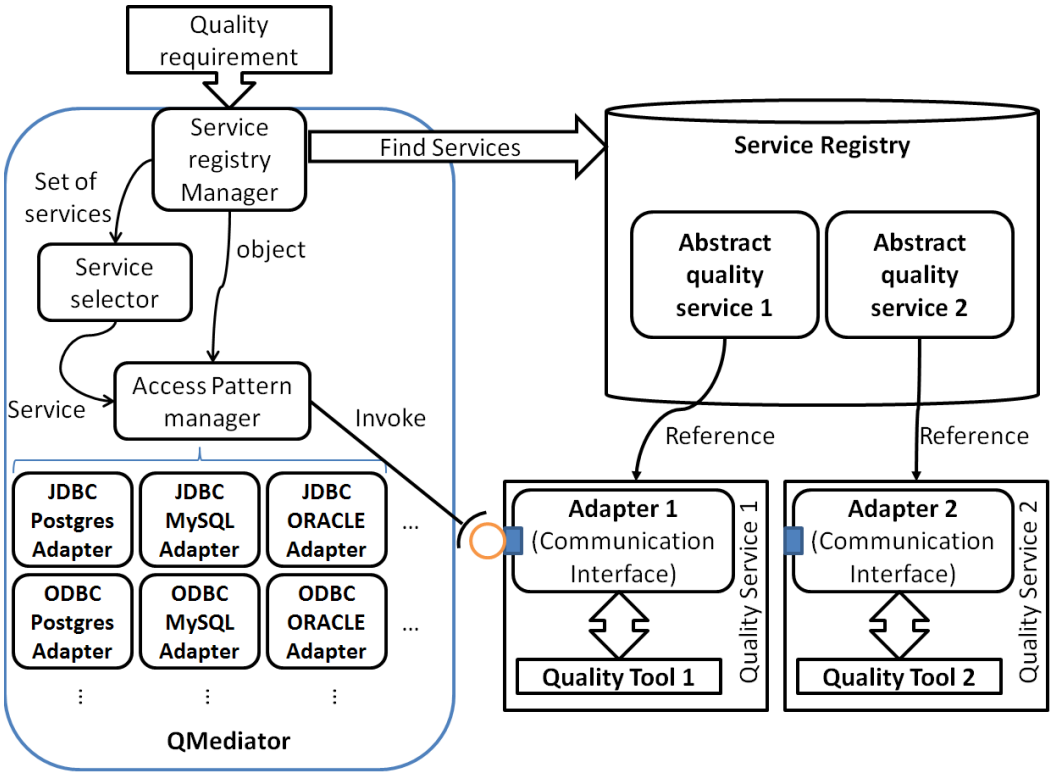

FIGURE 2.4 – Architecture de médiation

1. *Outils de qualité (Quality Tool)* fournissent des fonctionnalités d'évaluation de qualité de données qui peuvent être soit développées par une personne au sein d'une organisation soit fournies par d'autres outils de qualité externe.
2. *Services de qualité abstraits (Abstract Quality Service)* est une description des fonctions de qualité fournie par un service de qualité *(Quality Service)*. Cette description inclut le concept de qualité que ce service adresse (dimensions, facteurs ou métriques), le type de fonctionnalité qu'il fournit (Mesure, Analyse ou Amélioration

des données), la catégorie d'objets traités (Tables, Attributs, Fichier XML, etc), le type d'objets supportés (String, Numérique, Date, etc), le mode de communication avec les sources de données (JDBC, ODBC, SQLCLI, etc), et le point d'accès au service.

3. *Annuaire de services (Service Registry)* fournit un moyen qui nous permet de publier et de découvrir des services de qualité spécifiques à des besoins particuliers. L'annuaire de services va publier chaque service de qualité sous forme de service de qualité abstrait.
4. *Gestionnaire d'annuaire de services (Services Registry Manager)* composant du *QMediator*, il englobe les méthodes qui nous permettent d'accéder à l'annuaire de services. Il nous permet entre autre de chercher des services suivant des critères particuliers *(concepts adressés, fonctions fournies, catégorie d'objets traités et le type d'objets supportés)* et de récupérer par conséquent un ensemble de services de qualité abstraits qui leur correspondent.
5. *Sélectionneur de services (Services selector)* composant du *QMediator*, qui implémente des algorithmes qui nous permettent de sélectionner le meilleur service à exécuter parmi l'ensemble des services retournés par le *Gestionnaire d'annuaire de services*. Pour cela il choisit le service le plus utilisé, et offre aussi la possibilité d'effectuer un choix explicite.
6. *Adaptateurs d'objets (Objects adapters)* sont des implémentations du pattern *adapter (défini par GoF de « Gang of Four », d'après les quatre créateurs du concept [15])* qui vont adapter l'interface de l'objet à mesurer (voir la figure 2.5) à une interface qui correspond à celle du *service de qualité* sélectionné.

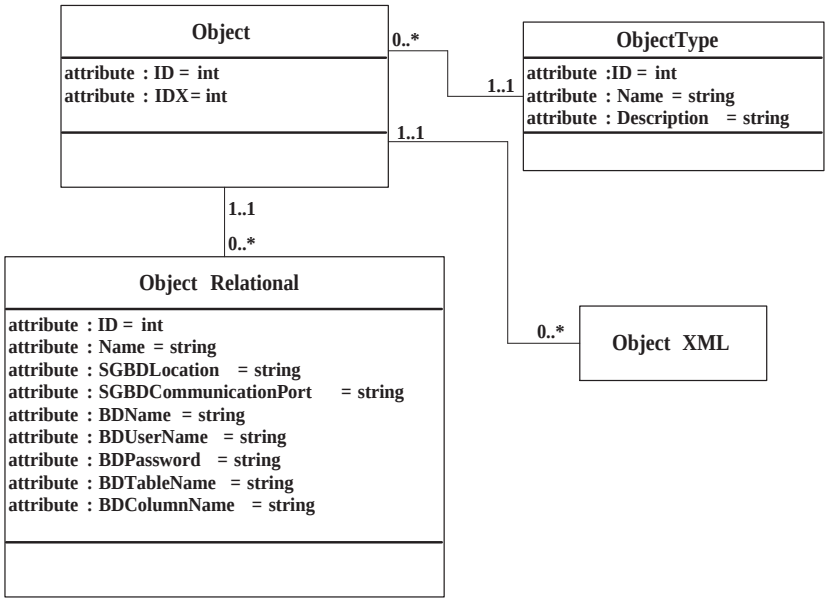

FIGURE 2.5 – Interface des objets à mesurer

7. *Gestionnaire des patterns d'accès (Access Patterns Manager APM)* est aussi un composant du *QMediator*, il va choisir *l'adaptateur d'objet* à utiliser suivant :

- Le type de connexion aux sources de données (JDBC, ODBC, SQLCli, etc) qu'utilise le service de qualité choisi par *le sélectionneur de services*.
- Le type du *SGBD* ou est stocké l'objet (Postgres, ORACLE, MySQL, etc).

8. *Adaptateurs (Adapter)* est une interface qui implémente une des signatures spécifiée dans le gestionnaire de patterns d'accès, suivant le mode de connexion aux sources de données d'un outil de qualité particulier. Il fournit entre autre la manière d'invoquer un service en faisant abstraction des détails technologiques d'un outil de qualité. On peut voir un adaptateur comme étant une couche implémentée sur un outil de qualité.
9. *Service de qualité (Quality Service)* est un outil de qualité qui est associé à un adaptateur spécifique. L'interface de communication qui est publiée dans *l'Annuaire de services* est celle de l'adaptateur.

2.5.1 Gestionnaire des patterns d'accés

Nous avons modélisé la partie access pattern comme le montre la figure 2.6.

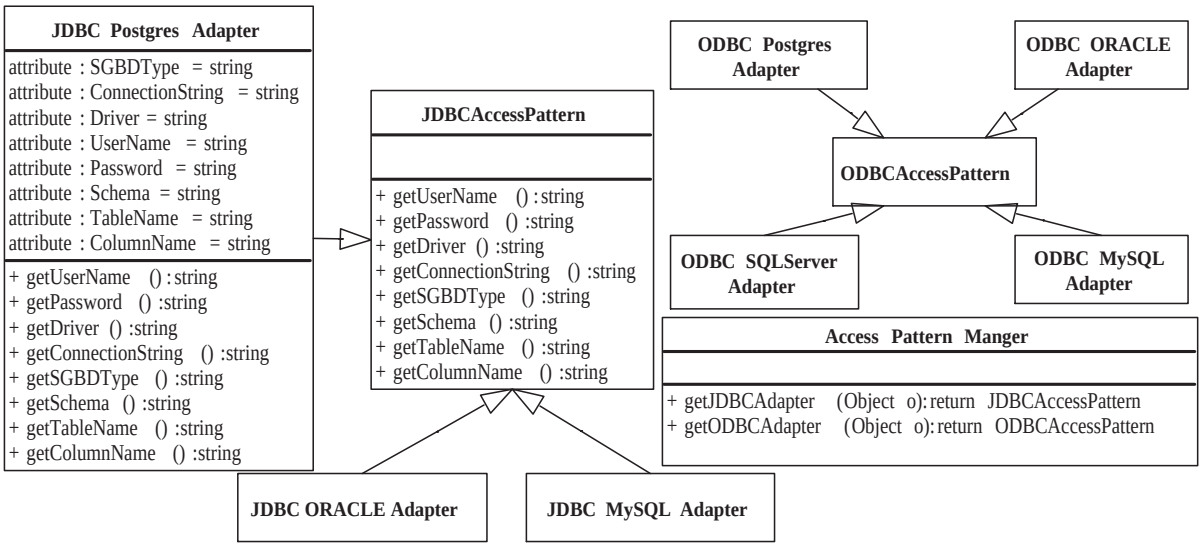

FIGURE 2.6 – Diagramme de classes de la partie Access Pattern

Comme il a été dit précédemment, les patterns d'accès sont gérés par un *Gestionnaire de patterns d'accés*, qui va suivant le type de connexion supportée par le service, choisir une catégorie de pattern qui peut être soit *JDBC Access Pattern, ODBC Access Pattern, SQLCli Access Pattern, etc.* La catégorie de pattern est une classe abstraite qui représente la signature à laquelle doivent se conformer les adaptateurs pour matcher les interfaces des services de qualité. Cette procédure de sélection de la catégorie de patterns d'accès est formalisée dans l'algorithme 1.

Une fois la catégorie choisie, l'*APM* va instancier la catégorie avec un adaptateur d'objets qu'il choisira suivant le type du SGBD dans le quel l'objet est stocké *(JDBCAccessPattern ap=new JDBCPostgresAdapter(o) ;).*

Algorithm 1: Algorithme de sélection de la catégorie de patterns

```
Data: Service s
Result: patterns category c
initialization;
LET i := readInterface(s);
LET APc := readAccessPattern(i);
if APc=="JDBCAccessPatter" then
 | return JDBCAccessPatter;
else if APc=="ODBCAccessPatter" then
 | return ODBCAccessPatter;
else if APc=="SQLCliAccessPatter" then
 | return SQLCliAccessPatter;
else if APc=="ADO.NETAccessPatter" then
 | return ADO.NETAccessPatter;
else
 | return unavailableCategorie;
end
```

Les adaptateurs d'objet ont été identifiés comme étant des patterns de type *Adapter (Wrapper)*, dont le but est de convertir l'interface d'une classe selon les besoins, et aussi de permettre aux classes de communiquer entre elles si leurs interfaces ne sont pas compatibles [15]. Ce pattern *(Adapter)* est utilisé dans les cas suivants :

- Si on veut utiliser une interface qui ne matche pas avec celle dont nous avons besoin.
- Si on veut créer une classe qui coopère avec une technologie qui n'est pas basée sur le concept objet.
- Si on veut fournir une interface multiple à un objet lors de sa conception.

La figure 2.7 illustre un exemple simple du pattern *(JDBCAccessPatter)* avec lequel on adapte l'interface de la classe objet, à celle de l'interface du service.

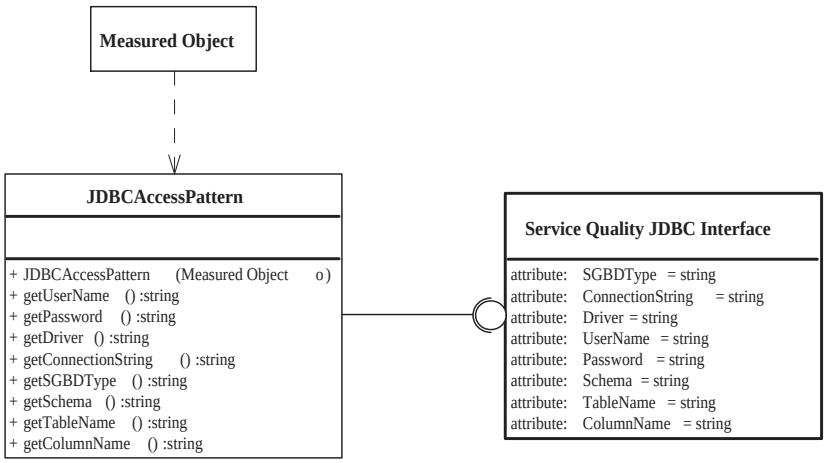

FIGURE 2.7 – ExempleAdapter

Le choix de l'adaptateur d'objets est aussi formalisé dans l'algorithme 2.

Algorithm 2: Algorithme de sélection de l'adaptateur d'objets pour la catégorie JDBCAccessPatter

```
Data: Object o
Result: Access Pattern Adapter ap
initialization;
LET SGBDType := readSGBDType(o);
if SGBDType=="PostgreSQL" then
    JDBCAccessPattern ap=new JDBCPostgresAdapter(o);
    return ap;
else if SGBDType=="ORACLE" then
    JDBCAccessPattern ap=new JDBCORACLEAdapter(o);
    return ap;
else if SGBDType=="SQLServer" then
    JDBCAccessPattern ap=new JDBCSQLServerAdapter(o);
    return ap;
else if SGBDType=="MySQL" then
    JDBCAccessPattern ap=new JDBCMySQLAdapter(o);
    return ap;
else if SGBDType=="DB2" then
    JDBCAccessPattern ap=new JDBCDB2Adapter(o);
    return ap;
else if SGBDType=="MSAccess" then
    JDBCAccessPattern ap=new JDBCMSAccessAdapter(o);
    return ap;
else if SGBDType=="Paradox" then
    JDBCAccessPattern ap=new JDBCParadoxAdapter(o);
    return ap;
else
    return unavailableAdapter;
end
```

Ex. si notre service de qualité fonctionne avec le mode de connexion JDBC et que l'objet est stocké dans PostgreSQL, le Gestionnaire des patterns d'accés va choisir comme catégorie JDBCAccessPattern pour le mode de connexion JDBC, et comme adaptateur JDBCPostgresAdapter pour PostgreSQL, le SGBD dans lequel l'objet à mesurer est stocké.

2.5.2 Scénario d'exécution d'un modèle de qualité personnalisé

La fonctionnalité principale du QMediator est de trouver et d'exécuter des services qui matchent au mieux des besoins spécifiques. Pour cela le composant *Services Registry Manager* qui encapsule les méthodes d'accès à l'annuaire de services, doit trouver les services de qualité abstraits qui correspondent à ces besoins en terme de qualité. Une fois ces services trouvés, le composant *Services Selector* va sélectionner le service le mieux adapté parmi l'ensemble de ces services en choisissant le service le plus exécuté (Ce choix peut être modifié explicitement). Le composant APM va à son tour selectionner l'adaptateur d'objets à utiliser suivant le type du SGBD où est stocké l'objet (PostgreSQL, MySQL, Oracle, etc), et le mode de connexion aux sources de données que supporte le

service sélectionné (JDBC, ODBC, ADO.NET, etc). Enfin, le composant APM va utiliser la nouvelle interface de l'objet adapté pour invoquer le service, qui va à son tour retourner un résultat. La figure 2.8 décrit à l'aide d'un diagramme de séquence l'exécution complète d'un modèle de qualité personnalisé PQM.

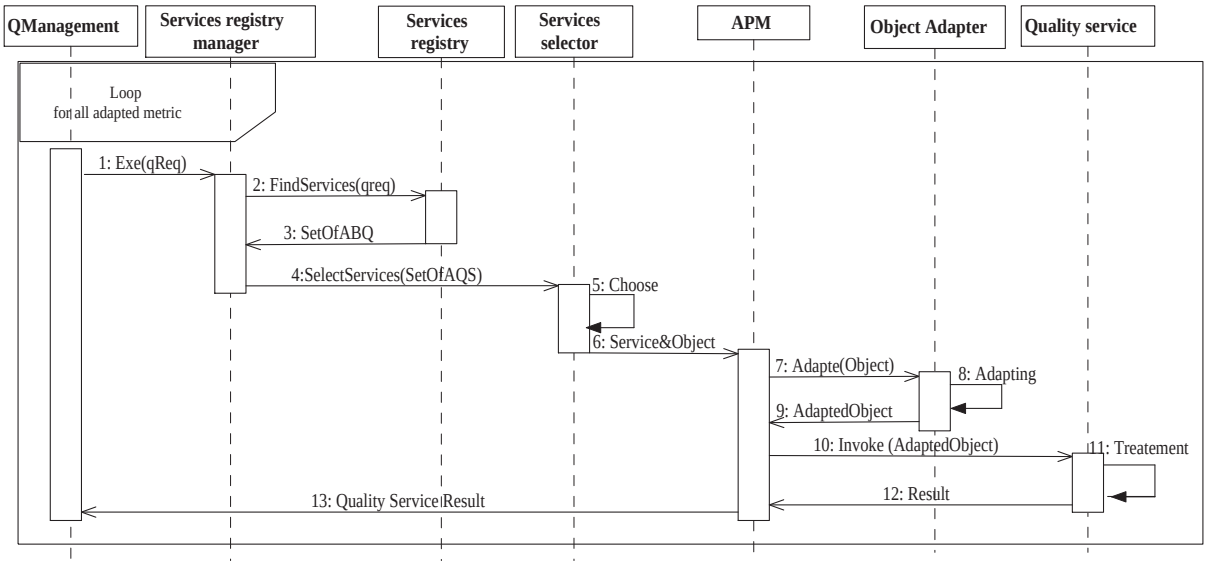

FIGURE 2.8 – Exécution d'un PQM

2.6 Normalisation des données

Les données qui représentent les résultats des mesures effectuées doivent être normalisées (c-à-d les rendre numériques et leurs modalités comprises entre [0,1]) afin de permettre leurs analyses, leurs projections et leurs comparaisons. Des techniques de normalisation sont présentées ci-dessous (ces techniques sont extraites des techniques de normalisation pour les réseaux de neurones [16]) :

2.6.1 Normalisation de variables continues

Le plus souvent les mesures effectuées sur les objets sont quantitatives et non qualitatives, et ne sont pas comprises dans un intervalle précis, c'est à dire que le plus souvent ces mesures seront des variables continues [4].

Plusieurs techniques existent pour normaliser ce type de variables *(quelque une sont décrites dans l'annexe A à la page 40)*, nous décrivons ci-dessous la technique que nous avons choisie pour normaliser et projeter les mesures effectuées :

Supposons que la modalité de notre variable X se trouve dans l'intervalle $[d^{min}, d^{max}]$ et non pas dans l'intervalle $[0, 1]$. On veut la transformer dans l'intervalle $[0, 1]$. Notons d la valeur originale de notre variable et δ la valeur de notre variable normalisée. Il

4. Pour plus d'information sur la normalisation des variables, se référer à l'annexe A page 40

existe beaucoup de moyens pour normaliser cette variable. En principe pour agréger une séquence de nombres dans un intervalle $[0, 1]$ on a besoin de les rendre positifs et les diviser par un nombre plus grand que le dénominateur. En utilisant ce principe, on peut utiliser n'importe quelle inégalité pour normaliser cette valeur.

Comme on connaît toujours l'intervalle $[d^{min}, d^{max}]$ de notre variable, alors on utilise la transformation suivante :

$$\delta = \frac{d - d^{min}}{d^{max} - d^{min}} \tag{2.1}$$

Cette fonction transformera notre variable dans un intervalle de $[0, 1]$. Si $d = d^{min}$ alors $\delta = 0$. Si $d = d^{max}$ alors $\delta = 1$. Un cas particulier est à prendre en considération quand $d^{max} = 0$. Si on sait que la valeur de notre variable est toujours soit égale à 0 soit positive alors on met $d^{min} = 0$ et la fonction peut être simplifiée en :

$$\delta = \frac{d}{d^{max}} \tag{2.2}$$

Le graphe de cette fonction est linéaire et dépend de d^{max}

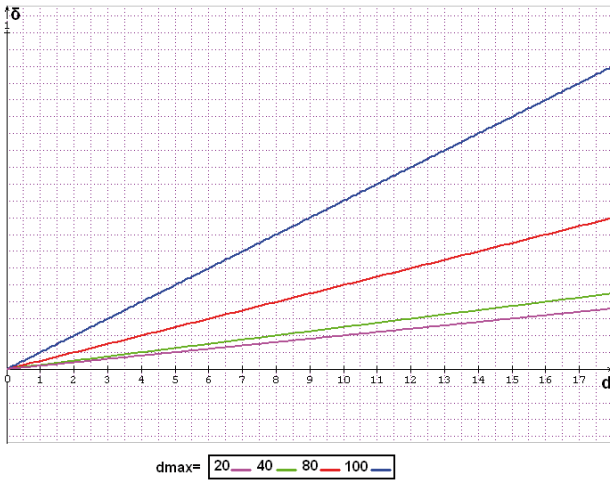

FIGURE 2.9 – Graphe de la fonction A.6

Dans le cas d'une variable négative, une technique est proposée dans l'annexe A pour la normaliser.

2.6.2 Normalisation de variables discrètes

Si le service de qualité qu'on exécute retourne une valeur discrète (la plage de valeur est connue), alors on utilise la technique suivante pour la normaliser :

1. Convertir les valeurs que peut prendre cette variable en rang ($r = 1$ jusqu'à R).
2. Normaliser ce rang en valeur standard de 0 jusqu'à 1; $[0, 1]$ en appliquant cette fonction :

$$\delta = \frac{r - 1}{R - 1} \tag{2.3}$$

Exemple

Plage de valeur originale	-2	-1	0	1	2	= i
Convertie à l'échelle	1	2	3	4	5	= r R = max(r)=5
Echelle normalisée	0	1/4	1/2	3/4	1	$\delta = \frac{r-1}{R-1}$

2.6.3 Normalisation de variables qualitatives

Dans le cas ou le service exécuté renvoie une variable qualitative (variable pour laquelle la valeur mesurée sur chaque individu ne représente pas une quantité), sa normalisation cause problème car elle doit faire apparaître une relation d'ordre parmi ses modalités qui est souvent artificielle et qui peut induire en erreur.

On peut par exemple créer des variables binaires dont la valeur 0 ou 1 signifie que la variable qualitative a ou non cette modalité.

Ex : une variable X qui prend ses valeurs dans : mauvais, moyen ou bon, peut être normalisée comme suit :

- X :
- $Mauvais \longrightarrow 0$
- $Moyen \longrightarrow 0,5$
- $Bon \longrightarrow 1$

2.7 Conclusion

Le but de ce chapitre est de décrire la *QBox-Services* une plate-forme qui fournit une infrastructure d'intégration à base de services qui permet l'interopérabilité de plusieurs outils de qualité trois tiers pour ainsi répondre aux problèmes d'évolution de la QBox-Foundation. Ce chapitre présente aussi quelques techniques qui permettent de normaliser les valeurs des mesures effectuées à des fins de projection et d'analyse (QOLAP[14]).

Le chapitre suivant décrit en détail le prototype de la plate-forme *QBox-Services*, ainsi que les technologies que nous avons utilisé pour le développer. Il décrit aussi des services de qualité qui font essentiellement de l'analyse *Cleansing (nettoyage de données)* de données, et que nous avons adapté au prototype.

Chapitre 3

Prototype

Dans ce chapitre nous allons présenter le prototype développé, qui intègre toutes les fonctionnalités de la QBox-foundation anisi que les autres fonctionnalités décrites dans le chapitre précédent.

3.1 Introduction

Ce chapitre sera entièrement consacré à la description du prototype conçu, d'une manière générale. Nous allons entre autres décrire les principaux composants qui constituent notre prototype et les technologies qui nous ont permis de les développer.

3.2 Prototype de la QBox

Nous allons introduire dans cette section le prototype que nous avons développé, et qui implémente l'architecture décrite précédemment dans la section 2.4. Ce prototype est composé de deux principaux modules : le module *QBox* et le module des *services de qualité*. Il inclut aussi les composants *Juddi*[17] et *JBoss-WS*[18]. La figure 3.1 montre le diagramme de déploiement du prototype de la *QBox-Services*.

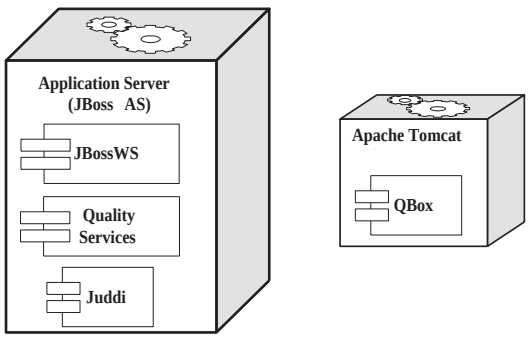

Figure 3.1 – Diagramme de déploiement de la QBox-Services

Dans les sections qui suivent nous allons détailler les principaux modules du prototype ; à savoir le module *QBox* et le module des *services de qualité*.

3.3 Services de qualité

Le prototype de la *QBox-Services* inclut des services fournis par *DataCleaner*[4], *Custom quality services* et *DQguru* [5]. Ce dernier fournit des services qui concernent essentiellement des méthodes de Cleansing (nettoyage de données).

3.3.1 Services de qualité de DataCleaner

DataCleaner est un outil qui fournit des méthodes d'évaluation de la qualité à partir de différentes sources de données. Pour intégrer les méthodes fournies par DataCleaner dans notre prototype, des adaptateurs sont développés en Java et leurs interfaces sont publiées comme services Web. La figure 3.2 montre la logique d'accès de l'adaptateur au noyau de *DataCleaner*.

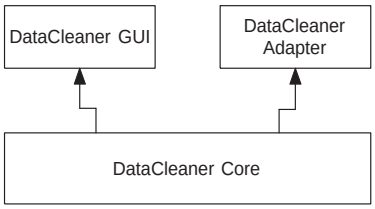

FIGURE 3.2 – Logique d'accès de l'adaptateur au noyau de DataCleaner

Les adaptateurs sont développés en Java, et leurs interfaces sont publiés comme services Web en utilisant *JAX-WS* [19], une implémentation fournie par le Framework JBoss Web Service (JBossWS), et sont déployés sur le serveur d'application JBoss [20].

Les méthodes de *DataCleaner* qui sont incluses dans le prototype, travaillent avec des bases de données relationnelles et y accèdent en utilisant *JDBC*. Ces méthodes ont besoin de recevoir comme paramètres d'entrées toutes les données nécessaires pour se connecter aux sources de données via *JDBC*. Le tableau 3.1 présente et décrit les paramètres JDBC requis pour établir une connexion.

Nom du paramètre	**Description**	**Exemple**
SGBDType	Nom du SGBD	Postgres
Driver	JDBC driver	org.postgresql.Driver
ConnectionString	Chaine de connexion à la base de données	Jdbc :postgresql ://localhost :5432/qbox
Schema	Schema de données	Public
Username	Nom d'utilisateur pour l'accédé a la BD	Postgres
Password	Password pour l'accédé à la BD	database

TABLE 3.1 – Paramètres JDBC

Le tableau 3.2 présente la description et les paramètres requis pour l'exécution du service de qualité *getEmptyValues*, qui est fourni par *DataCleaner*. Ce service de qualité

retourne le nombre de valeurs nulles dans une colonne précise. Le tableau 3.3 présente un résumé des méthodes de *DataCleaner* qui sont incluses dans le prototype.

Nom	**Type**	**Direction**	**Description**
JDBCParameters	N/A	IN	Paramètres JDBC comme spécifié dans le tableau 3.1.
tablename	String	IN	Nom de la table.
columnname	String	IN	Nom de la colonne.
emptyvalues	long	OUT	Nombre de valeurs nulles dans la colonne spécifiée.

TABLE 3.2 – Paramètres du service de qualité getEmptyValues

Catégorie dans DataCleaner	**Nom**
Standard Measures	getEmptyValues
	getHighestValue
	getLowestValue
	getNullValues
	getRowCount
Number Analysis	getGeometricMean
	getHighestValue
	getLowestValue
	getMean
	getStandardDeviation
	getSum
	getVariance
String Analysis	getCharCount
	getLowercaseChars
	getMaxChars
	getMaxWords
	getMinChars
	getMinWords
	getNonLetterChars
	getUppercaseChars
	getWordCount
Validation	getDictionaryValidation
	getJavascriptValidation
	getNotNullValidation
	getRangeValidation
	getRegularExpressionValidation

TABLE 3.3 – Résume des méthodes fournies par DataCleaner et incluses dans le prototype

3.3.2 Custom Quality Services

D'autres services qui implementent quelque métriques de qualité spécifiques sont présents dans le prototype. Ces services sont développés en Java et publiés comme services Web en utilisant aussi JAX-WS, qui est une implémentation fournie par le composant de JBossWS. Ces services sont aussi déployés sur le serveur d'application JBoss.

Le tableau 3.4 présente la description et les paramètres requis pour le service de qualité *getSyntacticCorrectnessRatioDictionary*. Ce service de qualité retourne une valeur entre 0 et 1, qui représente le pourcentage de valeurs syntaxiquement correctes par rapport à une référence donnée.

Nom	Type	Direction	Description
JDBCParameters	N/A	IN	Paramètres JDBC comme spécifiés dans le tableau 3.1.
tablename	String	IN	Nom de la table.
columnname	String	IN	Nom de la colonne.
Dictionarypath	String	IN	Référence a un dictionnaire.
syntacticcorrectnessratio	Double	OUT	Valeur entre 0 et 1, qui représente le pourcentage de valeurs syntaxiquement correctes par rapport à la référence.

TABLE 3.4 – Paramètres du service de qualité getSyntacticCorrectnessRatioDictionary

Le tableau 3.5 présente un résumé des méthodes de *Custom Quality Services* qui sont inclues dans le prototype.

Métrique	Nom
SyntacticCorrectnessRatio	getSyntacticCorrectnessRatioDictionary
	getSyntacticCorrectnessRatioRegexp
DensityRatio	getDensityRatio
RelationIntegrityRatio	getRelationIntegrityRatio
DuplicaRatio	getDuplicaRatio

TABLE 3.5 – Résume des méthodes fournies par Custom Quality Services incluses dans le prototype

3.3.3 DQguru Quality services

DQguru est un outil qui permet de corriger des données, valider et corriger des adresses, et identifier et supprimer les valeurs en double dans une base de données. Il permet entre autre d'avoir des données complètes et précises et une vue consolidée sur l'ensemble des informations essentielles.

La figure 3.3 présente l'interface utilisateur de *DQguru*.

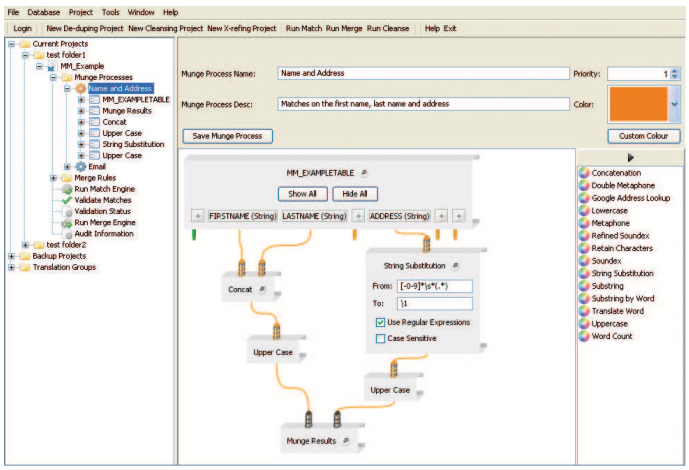

FIGURE 3.3 – Interface utilisateur de DQguru

DQguru est un outil facile à utiliser et très intuitif crée par les designers d'entrepôt de données. Il contient de nombreuses fonctionnalités qui aident à assurer l'intégrité des données. Le *Tableau 3.6* résume les principales caractéristiques de DQguru.

Fonctionnalités	**Source de donnée**	**Licence**	**Language de programmation**	**Dernière Version**
Calcul et correction	JDBC	Open source GPL	JAVA	0.9.5

TABLE 3.6 – Résumé des caractéristiques de DQguru

Pour intégrer les méthodes de DQguru dans le prototype, des adaptateurs sont développés en Java et publiés comme étant des services web. La *Figure 3.4* montre comment est positionné le noyau de l'outil DQguru par rapport à un adaptateur, et l'interface publiée.

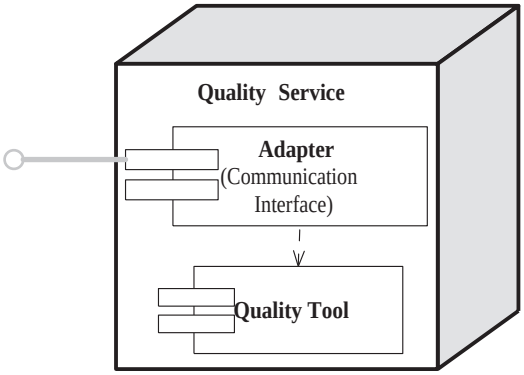

FIGURE 3.4 – Architecture de l'adaptateur de DQguru

Les interfaces des adaptateurs sont publiées comme service web en utilisant la technologie JAX-WS fournie par JBoss Web Service (JBossWS), et déployées sur le serveur d'application JBoss.

DQguru est un outil qui supporte JDBC comme interface de communication avec des sources de données, et travaille avec des bases de données relationnelles. Ces méthodes doivent recevoir comme paramètres d'entrées toutes les données nécessaires à DQguru pour se connecter à la base de données à travers JDBC.

Le tableau 3.7 présente la description et les paramètres requis pour le service de qualité *SetColumnUpperCase*. Ce service met en majuscule les valeurs d'une colonne spécifique.

Nom	**Type**	**Direction**	**Description**
JDBCParameters	N/A	IN	Paramètres JDBC comme spécifié dans le tableau 3.1.
tablename	String	IN	Nom de la table.
columnname	String	IN	Nom de la colonne.
StatOperation	Bool	OUT	Booléen qui dit si l'opération s'est bien déroulée.

TABLE 3.7 – Paramètres de SetColumnUpperCase de DQguru

Le *Tableau 3.8* résume les méthodes de *DQguru* incluses dans le prototype.

Nom de la méthode
Bool **SetColumnEmptyStringToNull**(String SGBDType, String driver, String connectionstring, String schema, String username, String password, String tablename, String columnname)
Bool **SetColumnLowerCase**(String SGBDType, String driver, String connectionstring, String schema, String username, String password, String tablename, String columnname)
Bool **SetColumnRetainCharacters**(String SGBDType, String driver, String connectionstring, String schema, String username, String password, String tablename, String columnname, String RetainCharacters)
Bool **SetColumnSortWords**(String SGBDType, String driver, String connectionstring, String schema, String username, String password, String tablename, String columnname)
Bool **SetColumnSubstring**(String SGBDType, String driver, String connectionstring, String schema, String username, String password, String tablename, String columnname, int beginIndex , int endIndex)
Bool **SetColumnTrimSpaces**(String SGBDType, String driver, String connectionstring, String schema, String username, String password, String tablename, String columnname)
Bool **SetColumnUpperCase**(String SGBDType, String driver, String connectionstring, String schema, String username, String password, String tablename, String columnname)

TABLE 3.8 – Résumé des méthodes de DQguru inclues dans le protoype

3.4 Annuaire de services

Le prototype utilise comme annuaire de services *Appache Juddi*[17]. Cet annuaire de services est déployé sur le serveur d'application JBoss. Nous avons publié trois fournisseurs de services dans cet annuaire a savoir : DataCleaner, QBoxFoundation et DQguru qui sont résumés dans le tableau 3.9.

Nom	**Description**
DataCleaner	Outil open source qui permet d'évaluer, de valider et de comparer des données.
QBoxFoundation	Plate-forme pour l'évaluation de la qualité basée sur le paradigme GQM.
DQguru	Outil open source qui permet de corriger des données, valider et corriger des adresses, et identifier et supprimer les valeurs en double dans une base de données

TABLE 3.9 – Fournisseurs de services publiés dans l'annuaire de services

Afin de classifier et de décrire ces services publiés dans l'annuaire de services, par rapport aux fonctionnalités qu'ils fournissent, cinq taxonomies sont publiées. Le tableau 3.10 présente et décrit ces taxonomies.

Nom	**Description**
qbox-org :quality :indicator	Taxonomie pour représenter les concepts de qualité.
qbox-org :quality :operation	Taxonomie pour représenter les opérations de qualité.
qbox-org :object :type	Taxonomie pour représenter les objets mesurés.
qbox-org :service :object-type	Taxonomie pour représenter le type de données de l'objet traité par le service.
qbox-org :access :accesspattern	Taxonomie pour représenter le pattern d'accès du service.

TABLE 3.10 – Taxonomies publiées pour catégoriser les services de qualité

Enfin, quelques uns des services de qualité décrits précédemment ont été publiés dans l'annuaire de services. Le tableau 3.11 présente ces services de qualité et leurs classifications par rapport aux taxonomies qui sont publiés dans l'annuaire de services.

Fournisseur	**Nom**	**Indicateur**	**Opération**	**Catégorie d'objet**	**Access Pattern**	**Type d'objet**
QBox Foundation	Syntactic Correctness Ratio Dictionary	Syntactic Correctness Ratio	calculus	column	JDBC AP	String
	Syntactic Correctness RatioRegexp	Syntactic Correctness Ratio	calculus	column	JDBC AP	String
	DensityRatio	DensityRatio	calculus	column	JDBC AP	All
	RelationIntegrityRatio	DensityRatio	calculus	column	JDBC AP	String
	DuplicaRatio	DuplicaRatio	calculus	column	JDBC AP	All
DataCleaner	NullValues	Data granularity	calculus	column	JDBC AP	All
	EmptyValues	Data granularity	calculus	column	JDBC AP	All
	RowCount		calculus	Table	JDBC AP	String
	Dictionary Validation		calculus	column	JDBC AP	String
	Javascript Validation		calculus	column	JDBC AP	String
	Regular Expression Validation		calculus	column	JDBC AP	String

TABLE 3.11 – Services de qualité publiés et leurs catégorisations

3.5 Module QBox

Le composant principal du prototype développé est le composant *QBox* (voir fugure 3.1), qui est à son tour composé d'un ensemble de composants plus spécifiques. Ces composants sont présentés dans la figure 3.5. Ce module est deployé sur le serveur d'application *Apache Tomcat* [21].

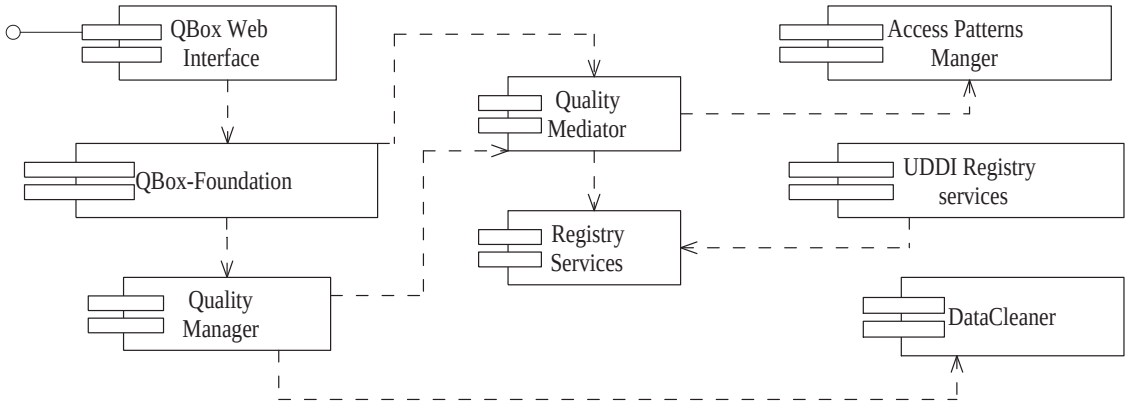

FIGURE 3.5 – Composants du module QBox

Le composant **Registry Service** encapsule les méthodes qui permettent d'accéder à l'annuaire de services ; spécifiquement le composant **UDDI Service Registry** implémente l'accès à l'annuaire de services *JUDDI*.

Le composant **Quality Mediator** fournit une interface qui permet de trouver des services appropriés à des besoins spécifiques en terme de qualité. Ce composant est lié au le composant **Registry Service** qui permet de chercher des services de qualité qui matchent au mieux ces besoins. Il est aussi lié au composant **Access Pattern Manager**

qui permet de choisir l'adaptateur à utiliser pour adapter l'interface de l'objet à mesurer à celle du service. Il est aussi chargé d'invoquer le service.

Le composant **Quality Management** reçoit les besoins en terme de qualité exprimés en *PQM* et les exécute.

Le composant **QBox-Foundation** offre les fonctionnalités qui permettent de définir des objectifs de qualité (goal), des facteurs et des métriques de qualité raffinés et spécialisés pour répondre au mieux aux besoins de ces objectifs.

Le composant **DataCleaner** fournit les fonctionnalités qui nous permettent de récupérer le type de données des objets à mesurer (String, Numrique, Date, etc).

Enfin, le composant **QBox Web Interface** est une interface Web conviviale et interactive développée en JAVA avec la technologie *JSF (Java Server Face)*, qui offre la possibilité à l'utilisateur d'utiliser toutes les fonctionnalités de la *QBox-Services*.

La figure 3.6 présente une interface Web de la *QBox-Services* qui permet de créer un *PQM*.

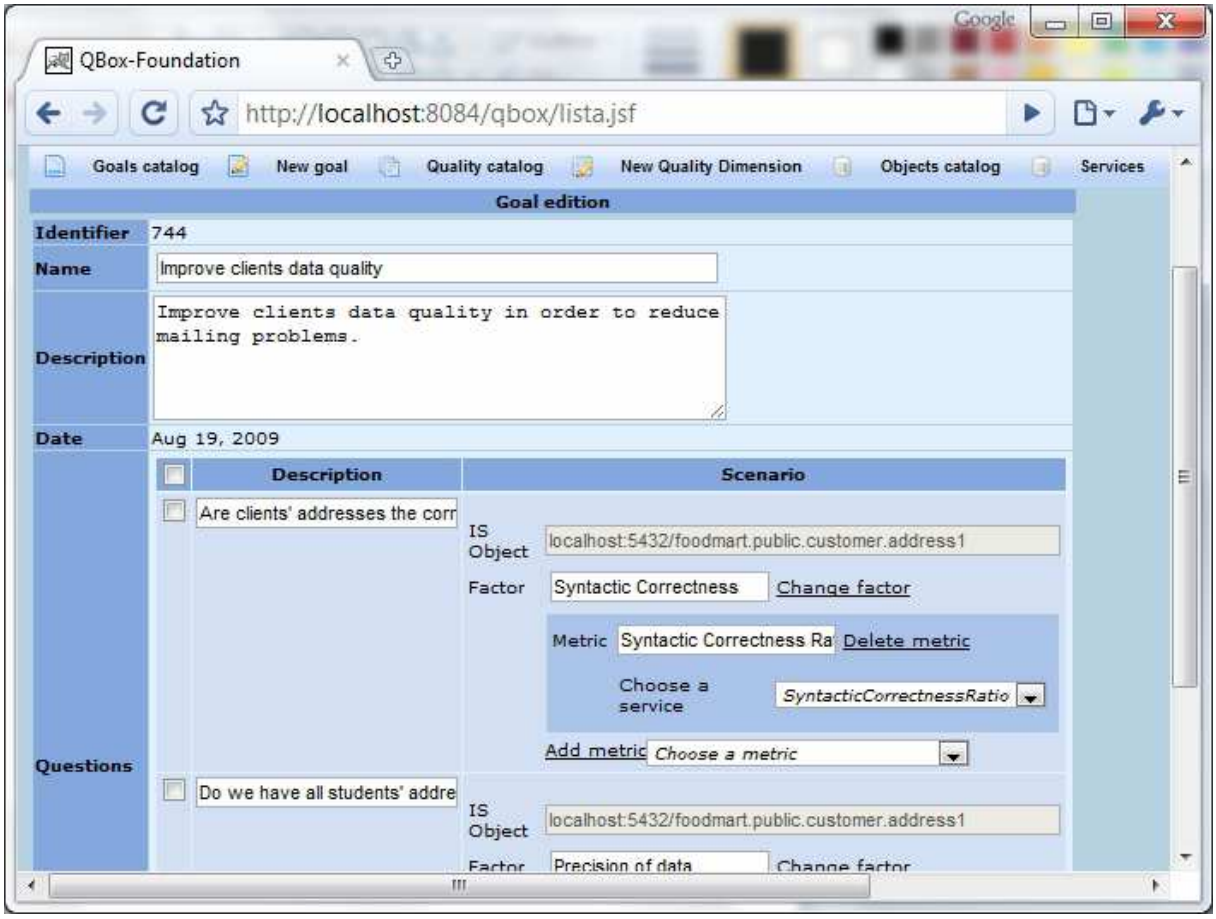

FIGURE 3.6 – Interface utilisateur de DQguru

3.6 Conclusion

Dans ce chapitre, nous avons présenté le prototype développé et les principales technologies que nous avons utilisé. Nous avons notamment présenté une large collection de services de qualité fournis par trois outils différents *(DataCleaner, Custom Quality Services et DQguru)* qui font essentiellement de *l'Analyse et du Cleansing* de données. Nous avons aussi présenté les différentes taxonomies publiées dans l'annuaire de services qui nous permettent de classifier et de catégoriser les services publiés, pour permettre une recherche plus précise des services. Enfin, nous avons présenté globalement, l'architecture interne du module *QBox*.

Dans le chapitre suivant, nous présentons une étude de cas qui va permettre de mettre en application ce prototype sur des données réelles et en exploitation.

Chapitre 4

Etude de cas

Les chapitres précédents ont présenté une nouvelle plate-forme la *QBox-Services*, une architecture à base de services qui permet d'intégrer des outils de qualité externes, et de définir un modèle de qualité personnalisé **(PQM)** pour évaluer la qualité des données. Ce chapitre présente une étude de cas faite sur des données réelles et en exploitation.

4.1 Exemple scénario d'analyse

Prenons un exemple de scénario d'analyse pour notre étude de cas, qui correspond à un système d'information de gestion commerciale. Plus précisément, nous avons choisi d'effectuer notre étude de cas sur une table *Tiers* dont le schéma est le suivant :

Tiers (pcf-code, cpt-numero, pcf-rs, pcf-rue, pcf-etat, pcf-cp, pcf-ville, pay-code, pcf-tel1, pcf-tel2, pcf-fax, pcf-email).

- pcf-code : Code du client.
- cpt-numero : Code comptable du client.
- pcf-rs : Raison sociale du client.
- pcf-rue : Rue du client.
- pcf-etat : L'état ou se trouve le client.
- pcf-cp : Code postale du client.
- pcf-ville : Ville ou se trouve le client.
- pay-code : Code pays du client.
- pcf-tel1 : Numéro téléphone du client.
- pcf-tel2 : Numéro téléphone du client.
- pcf-fax : Numéro de fax du client.
- pcf-email : Adresse email du client.

Nous distinguons quatre différents utilisateurs de la *QBox* :

- L'expert en gestion de qualité : Responsable de la gestion et de la maintenance des concepts de qualité (Bloc 1 du métamodele).
- Gestionnaire commercial : Responsable de la définition des buts et des questions de qualité, en plus de leurs associations avec les facteurs de qualité et les objets du système d'information (première partie du bloc 2).
- L'administrateur système : Responsable des accès aux objets du système d'information (Bloc 3).

– L'analyste de la qualité : Responsable de la spécialisation des facteurs et métriques, de l'exécution des services et de l'analyse des résultats obtenus (deuxième partie du bloc 2 et bloc 4).

Dans le but d'aider l'expert en gestion de qualité, nous avons implémenté une librairie initiale de dimensions, facteurs et métriques de qualité. Le tableau 4.1 énumère les facteurs et métriques de la dimension *Accuracy*.

Dimension	**Accuracy**	concerne l'exactitude et la précision qui existent entre les données présentes dans le système d'information et les données qui représentent le monde réel.
Facteur	Semantic correctness	Décrit la façon dont les données représentent le monde réel.
Métrique	Semantic correctness Boolean	Un booléen qui indique si un système de référence correspond au monde réel.
	Semantic correctness Degree	Représente la confiance/impression du degré de correction des données.
Facteur	Syntactic correctness	Exprime le degré d'une donnée à ne pas contenir d'erreur syntaxique.
Métrique	Syntactic correctness Boolean	Un booléen qui indique si les données satisfont des règles syntaxiques.
Facteur	Precision	Le niveau de détail de présentation des données.
Métrique	Scale	La précision associée à l'échelle de mesure.
	Standard error	L'écart-type d'un ensemble de mesure.

TABLE 4.1 – Exemple d'une dimension définie dans le système

4.2 Définition du modèle de qualité personnalisé

Le gestionnaire commercial définit des buts de qualité, et les décompose en un ensemble de questions auxquelles il associe des objets du système d'information et des facteurs de qualité. Le tableau 4.2 illustre la décomposition d'un but de qualité en un ensemble de questions, en plus de leurs associations à des objets du système d'information et des facteurs de qualité. Les facteurs de qualité sont sélectionnés à partir de la librairie de facteurs que propose le bloc 1 du metamodele. Ils peuvent être renommés ou adaptés dans le but de répondre au mieux aux questions.

L'analyste de la qualité qui doit avoir une bonne compréhension du domaine d'application, des objets du système d'information et de la librairie de qualité, doit choisir les métriques de qualité appropriées et les associer aux questions de qualité. L'instanciation de métriques de qualité consiste en la sélection de la métrique, et éventuellement adapter son nom et sa description pour répondre au mieux à la question.

But : Evaluer les données Clients(Adresse, numéro de téléphone, etc)		
Question	**Objet du SI**	**Facteur**
1- Est ce qu'il y a des clients non identifiés ?	PCF-RS	Density
2- Est ce qu'il y a des clients en double ?	PCF-RS	Uniqueness
3- Est ce qu'il y a tous les numéros de téléphone ?	PCF-TEL1	Density
4- Est ce qu'il y a des clients avec les mêmes numéros de téléphone ?	PCF-TEL1	Uniqueness
5- Est ce que les numéros de téléphones sont corrects ?	PCF-TEL1	Syn.Corr
6- Est ce qu'il y a tous les emails des clients ? ?	PCF-EMAIL	Density
7- Est ce qu'il y a des clients qui ont les mêmes emails ?	PCF-EMAIL	Uniqueness
8- Est ce que les emails sont corrects ?	PCF-EMAIL	Syn.Corr

TABLE 4.2 – Décomposition d'un but de qualité et son association à des objets du SI et des facteurs de qualité

Le tableau 4.3 montre un exemple de métriques adaptées (spécialisées) et de services de qualité, associés aux questions de qualité du tableau 4.2.

Question	**Métrique**	**Service**	**Paramètre**
1	Raison sociale Density Ratio	Density Ratio	Paramètres JDBC
2	Raison sociale Duplica Ratio	Duplica Ratio	Paramètres JDBC
3	Téléphone Density Ratio	Density Ratio	Paramètres JDBC
4	Téléphone Duplica Ratio	Duplica Ratio	Paramètres JDBC
5	Téléphone Syntactic Correctness	Syntactic Correctness	Paramètres JDBC ; RegEx
6	Email Density Ratio	Density Ratio	Paramètres JDBC
7	Email Duplica Ratio	Duplica Ratio	Paramètres JDBC
8	Email Syntactic Correctness	Syntactic Correctness	Paramètres JDBC ; RegEx

TABLE 4.3 – Instanciation des métriques de qualité pour les questions du tableau 4.2

4.3 Exécution du modèle de qualité personnalisé

Le modèle de qualité personnalisé *(PQM)* défini dans la section 4.2 a pour but, *d'évaluer les données Clients*. Il a été exécuté sur des images de la table *Tiers*, dont le schéma est défini dans la section 4.1. Ces images sont prises aux dates suivantes : *10/06/2009, 15/06/2009, 22/06/2009, 30/06/2009, 09/07/2009, 20/07/2009, 06/08/2009, 15/08/2009, 26/08/2009*, elles vont nous permettre d'analyser la qualité de ces données, et de voir comment elles évoluent dans le temps. La figure 4.1 présente les résultats obtenus suite à l'exécution de ce PQM, sur les objet *PCF-EMAIL et PCF-TEL1* et aux dates : *10/06/2009 et 26/08/2009*.

On peut déduire de la figure 4.1 que :

- Au *10/06/2009* la densité des données de l'objet *PCF-EMAIL* était de 8% et n'a pas changé au *26/08/2009*.
- Au *10/06/2009* le taux de duplication des données de l'objet *PCF-EMAIL* était de 0% et n'a pas changé au *26/08/2009*.

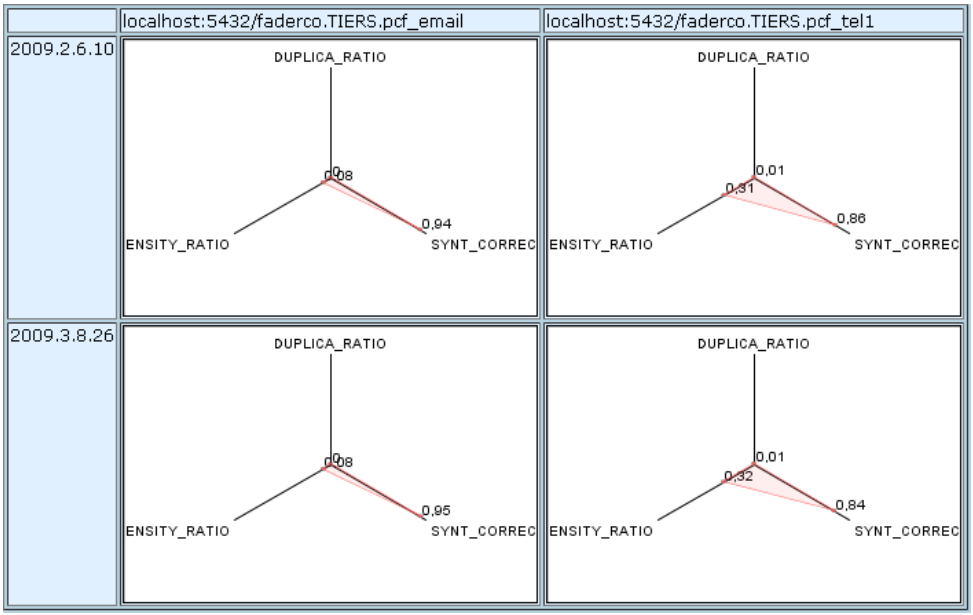

FIGURE 4.1 – Résultat de l'exécution du PQM

- Au *10/06/2009* le taux de correction syntaxique des données de l'objet *PCF-EMAIL* était de 94% et est devenu 95% au *26/08/2009*.
- Au *10/06/2009* la densité des données de l'objet *PCF-TEL1* était de 31% et est devenue 32% au *26/08/2009*.
- Au *10/06/2009* le taux de duplication des données de l'objet *PCF-TEL1* était de 1% et n'a pas changé au *26/08/2009*.
- Au *10/06/2009* le taux de correction syntaxique des données de l'objet *PCF-TEL1* était de 86% et est devenu 84% au *26/08/2009*.

On peut conclure que sur une période de 3 mois, les valeurs des indicateurs de qualité de ces deux objets ont changé. Ex. dans le cas de l'objet *PCF-EMAIL*, sa densité a augmenté (de 31% à 32% donc des données ont probablement été saisies), ce qui a pu influencer négativement sa correction sémantique (de 86% à 84% probablement les données saisies sont erronées).

En analysant ces résultats, on peut étudier les corrélations qui existent entre les différents concepts et facteurs de qualité, ce qui nous permet d'améliorer au mieux la qualité des données, en plus d'anticiper les comportements et les changements de la qualité des données. Enfin, dans ce chapitre nous sommes arrivés à définir un modèle de qualité personnalisé, relatif à une base de données réelles et en exploitation, et de l'exécuter afin d'évaluer et d'estimer la qualité des données de cette base.

Conclusion et perspectives

Bien que la *QBox-Foundation* soit une plate-forme pertinente et utile pour évaluer la qualité des données dans les systèmes d'information, ses limites ont été révélées concernant principalement son évolution :

- Incomplétude des méthodes de qualité.
- Impossibilité d'intégrer les méthodes déjà existantes.
- Le développement de nouvelles méthodes peut prendre beaucoup de temps et être très coûteux.
- Ne propose pas une approche d'analyse des mesures effectuées.

Dans ce document nous avons présenté la *QBox-Services* une évolution de la plate-forme *QBox-Foundation*, qui fournit une infrastructure d'intégration à base de services permettant l'interopérabilité de plusieurs outils de qualité trois tiers en plus d'une analyse multidimensionnelle des mesures effectuées basée sur *OLAP*[14]. Cette nouvelle plate-forme fournit aussi à l'utilisateur une large collection de services de qualité d'analyse et de Cleansing (nettoyage de données) pour réduire les efforts d'implémentation.

La *QBox-Services* conserve toute la facilité qui permet de définir et de personnaliser des modèles de qualité par rapport aux buts de qualité définis par l'utilisateur. Le *QMediator* raffine la partie recherche de services, il contient entre autre un mécanisme qui permet de sélectionner le meilleur service à exécuter parmi un ensemble de services. Enfin une large collection d'adaptateurs d'objets est développée pour prendre en charge les connexions **JDBC**.

Une piste possible dans la continuité de nos travaux, est de faire évoluer la partie recherche de services en intégrant **Web Ontology Language for Web Services (OWL-S)**, qui permet la description des propriétés et des capacités d'un service Web, pour ainsi sélectionner le service qui répond au mieux aux besoins en terme de qualité. On peut aussi étudier des formalismes qui nous permettent d'exprimer des requêtes de qualité dans des domaines d'application variés. Enfin, on peut étudier la composition de services pour ainsi obtenir les fonctions d'un service qui n'est pas implémenté.

Annexe A

Normalisation des données

A.1 Rappel statistique

A.1.1 Définition variable aléatoire

Une variable aléatoire est une fonction définie sur l'ensemble des éventualités, c'est-à-dire l'ensemble des résultats possibles d'une expérience aléatoire.

A.1.2 Variable aléatoire discrète

C'est une variable aléatoire qui prend ses valeurs dans un intervalle fini **0, 1, 2, 3, etc.**.

A.1.3 Variable aléatoire continue

C'est une variable aléatoire qui est définie par une fonction de densité de probabilité. Exemple : la loi de poisson est une variable aléatoire qui suit une loi de probabilité définie par la fonction

$$P(k) = P(X = k) = e^{-\lambda}\frac{\lambda^k}{k!} \tag{A.1}$$

A.2 Techniques de normalisation de variables continues

Supposons que la modalité de notre variable X se trouve dans l'intervalle $[d^{min}, d^{max}]$ et non pas dans l'intervalle $[0, 1]$. On veut la transformer dans l'intervalle $[0, 1]$. Notons d la valeur originale de notre variable et δ la valeur de notre variable normalisée. Il existe beaucoup de moyens pour normaliser cette variable. En principe pour agréger une séquence de nombres dans un intervalle $[0, 1]$ on a besoin de les rendre positifs et les diviser par un nombre plus grand que le dénominateur. En utilisant ce principe, on peut utiliser n'importe quelle inégalité pour normaliser cette valeur. Ci-dessous quelques transformations simples qui peuvent être utilisées pour une large classe d'application (il faut faire attention aux conditions de chaque transformation)

1. Une des méthodes de normalisation pour les variables continues est d'utiliser cette fonction :

$$\delta' = \frac{d}{\sqrt{d^2 + a}} \tag{A.2}$$

La valeur de δ' se trouvera dans l'intervalle -1 jusqu'à $+1$ pour chaque $a > 0$. Cette équation peut être facilement ramenée à l'intervalle $[0, 1]$ avec cette transformation :

$$\delta = \frac{1 + \delta'}{2} \tag{A.3}$$

Le résultat donnera la fonction suivante :

$$\delta = \frac{1}{2}(1 + \frac{d}{\sqrt{d^2 + a}}), a > 0 \tag{A.4}$$

La figure A.1 représente la variation de la fonction A.4 par rapport à la valeur de a.

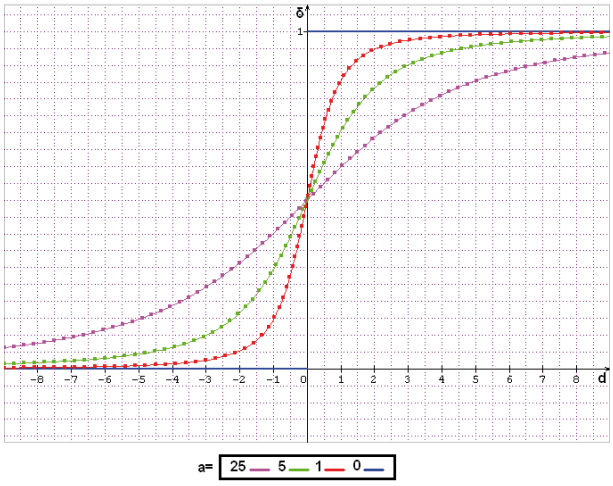

FIGURE A.1 – Graphe de la fonction A.4 suivant le paramètre a

En général quand $d < 0$ alors $\delta < 0,5$, et quand $d > 0$ alors $\delta > 0,5$. Pour $a = 0$, la fonction va produire une valeur binaire de 0 ou $+1$ avec une discontinuité quand $d = 0$. Le cas ou $a = 0$ ne doit pas être utilisé. La valeur du paramètre a dépend de la largeur de la variable d.
Exemple, $d = -4$ et $a = 100$ alors $\delta = \frac{1}{2}(1 + \frac{-4}{\sqrt{(-4)^2+100}}) = 0,31$

2. Si on connaît l'intervalle $[d^{min}, d^{max}]$ de notre variable, alors la transformation devient :

$$\delta = \frac{d - d^{min}}{d^{max} - d^{min}} \tag{A.5}$$

Cette fonction transformera notre variable dans un intervalle de $[0, 1]$. Si $d = d^{min}$ alors $\delta = 0$. Si $d = d^{max}$ alors $\delta = 1$. Un cas particulier est a prendre en considération

quand $d^{max} = 0$. Si on sait que la valeur de notre variable est toujours soit égale à 0 soit positive alors on met $d^{min} = 0$ et la fonction peut être simplifiée en :

$$\delta = \frac{d}{d^{max}} \tag{A.6}$$

Le graphe de cette fonction est linéaire et dépend de d^{max}

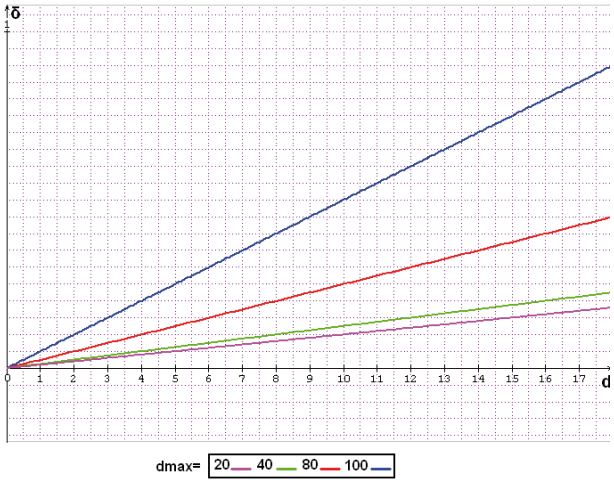

FIGURE A.2 – Graphe de la fonction A.6

3. Dans ce cas on sait que la valeur de notre variable est soit égale à 0 soit positive, mais on ne connaît pas sa valeur maximum. Supposons que le nombre de variables qu'on a est égal a n alors on peut utiliser le total de ces indices pour remplacer la valeur maximum comme suit :

$$\delta = \frac{d_i}{\sum_{i=1}^{n} d_i} \tag{A.7}$$

La normalisation de cette fonction est plus petite que celle de la fonction précédente car $\sum_{i=1}^{n} d_i \geq d^{max}$. Il faut éviter le cas ou les indices sont égaux à 0 car on aura une division par 0.

4. Si notre variable peut prendre des valeurs négatives alors on peut normaliser chaque indice en prenant la valeur absolue ou le carré de chaque valeur pour le total :

$$\delta = \frac{|d_i|}{\sum_{i=1}^{n} |d_i|} \tag{A.8}$$

ou

$$\delta = \frac{d_i^2}{\sum_{i=1}^{n} d_i^2} \tag{A.9}$$

5. On sait mathématiquement que pour n'importe quelle valeur positive, la moyenne arithmétique est toujours plus grande ou égale à la moyenne géométrique. On peut utiliser cette propriété pour normaliser notre variable. En supposant que $d_i > 0$ on a :

$$\delta = \frac{n(\prod_{i=1}^{n} d_i)^{\frac{1}{n}}}{\sum_{i=1}^{n} d_i} \tag{A.10}$$

Par exemple : $d_1 = 5$ et $d_2 = 7$ on obtient : $\delta = \frac{2\sqrt{5\times7}}{5+7} = 0,986$

6. Une autre fonction tirée des mathématiques théoriques, dit que la valeur absolue d'une moyenne arithmétique est plus petite ou égale à la moyenne quadratique. On peut utiliser cette propriété pour normaliser notre variable pour une valeur réelle d_i

$$\delta = \frac{|\sum_{i=1}^{n} di|}{\sqrt{n \sum_{i=1}^{n} d_i^2}} \tag{A.11}$$

Par exemple si : $d_1 = -5$ et $d_2 = 7$ alors on obtient $\delta = \frac{|-5+7|}{\sqrt{2((-5)^2+7^2)}} = 0,040$ Il existe aussi des techniques pour normaliser un couple de valeur x, y en une seule valeur dans l'intervalle $[0, 1]$ et aussi des techniques pour normaliser les valeurs négatives.

Annexe B

QBox Installation Guide

Title :	QBox Installation Guide
Version :	1.2
Date :	10/09/2009
Download :	http://www.megaupload.com/...

TABLE B.1 – Document Properties

Version	**Revision Date**	**Modified By**	**Description**
1.0	27/01/2009	Laura GONZÁLEZ	Initial Version
1.1	30/01/2009	Laura GONZÁLEZ	Corrections
1.2	10/09/2009	F.LEMOS et R.BOUADJENEK	Intégration

TABLE B.2 – Change History

B.1 System Requirements

- JDK6
- PostgreSQL 8.2 or superior
- Apache Tomcat 6.0 or superior
- JBoss AS 4.2.3
- JBoss WS 3.0.5 Native

B.2 Data Bases installation

In order to setup the database for the QBox application the following steps has to be followed :

1. Create a postgresql user with the following details :
   - Username : postgres
   - Password : database
2. Copy qbox.sql, juddi.sql and qolap.sql files in to <PostgreSQL-HOME>\scripts

3. Execute <PostgreSQL-HOME>\scripts\runpsql.bat and let default all the parameters
4. In the program command line enter the following commands :
   \i qbox.sql <PRESS ENTER>
   \i juddi.sql <PRESS ENTER>
   \i qolap.sql <PRESS ENTER>

B.3 Application Installation

To run the QBox Services prototype four components are required. In order to deploy these components an instances of the Apache Tomcat and JBoss Application Server has to be configured. Figure B.1 shows a simple deployment diagram that shows the distribution of these components.

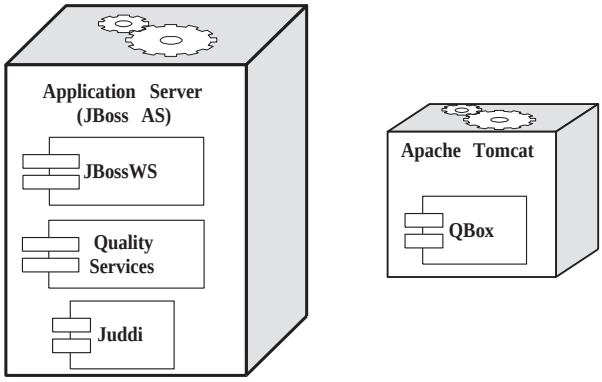

FIGURE B.1 – Simple Deployment Diagram for the QBox Services Prototype

B.3.1 JBossWS and juddi installation

In order to install JBossWS and juddi in the JBoss AS the following steps have to be followed :

1. Install JBossWS 3.0.5 in the JBoss AS installation as detailed in the JBossWS distribution. This will install de JBossWS component along with a UDDI registry (juddi).
2. Place the file qbox-services-setup-v1-0.rar\resources\juddi\juddi-ds.xml in JBOSS-HOME\server\default\deploy
3. Modify the following line in the file JBOSS-HOME\server \default\deploy\juddiservice.sar\META service.xml : <attribute name="DataSourceUrl">java :\JuddiDS<\attribute>
4. Modify the following line in the file JBOSS-HOME\server\default\deploy\juddiservice.sar\juddi.w INF\juddi.properties :
   jUDDI DataSource to use juddi.dataSource=java :\JuddiDS

5. Place the file qbox-services-setup-v1-0.rar\resources\postgres\postgresql-8.2-505.jdbc3.jar in JBOSS-HOME\server\default\lib

B.3.2 Quality Services Installation

In order to install the Quality Services component in JBoss AS, copy the files Custom-Services.war, DataCleanerServices.war and DQguruServices.war in JBOSS-HOME\server\default\deplo

B.3.3 Main Application Installation

In order to install the main QBox application in Apache Tomcat copy the file QBox.war in <tomcat>\webapps\. Open the file <tomcat>\webapps\qbox\META-INF\context.xml and edit the qolap database parameters. Open the file <tomcat>\webapps\qbox\WEBINF\classes\hibe and edit the qbox database parameters.

B.3.4 Accessing the QBox-Service Application

Finally to run the application you have to run first the JBoss AS by execute the file run.bat in <jboss>bin\folder and run Apache Tomcat by execute the file startup.bat in <tomcat>\bin.

In order to access the QBox-Service Application, open a browser an go to the following url : http://localhost:8084/qbox

Bibliographie

[1] Richard Y. Wang and Diane M. Strong. Beyond accuracy : What data quality means to data consumers. *Journal on Management of Information Systems*, 12 :29, 1996.

[2] Michael V. Mannino and Zhiping Walter. A framework for data warehouse refresh policies. *Technical Report CSIS-2004-001*, 2004.

[3] Bongsik Shin. An exploratory investigation of system success factors in data warehousing. *Journal of the Association for Information Systems*, 4, 2003.

[4] http ://datacleaner.eobjects.org/.

[5] Http ://www.sqlpower.ca/page/dqguru Group SQLPower DQguru, 2009.

[6] http ://www.talend.com/.

[7] Jacky Akoka, Laure Berti-Equille, Omar Boucelma, Mokrane Bouzeghoub, Isabelle Comyn-Wattiau, Mireille Cosquer, V. Goasdouí-Thion, Zoubida Kedad, Sylvaine Nugier, Verónika Peralta, and Samira Si-Said Cherfi. A framework for quality evaluation in data integration systems. *ICEIS*, 2007.

[8] Verónika PERALTA. *Data Quality Evaluation in Data Integration Systems*. PhD thesis, Université de Versailles Saint-Quentin-en-Yvelines (France) and Universidad de la República (Uruguay), 2006.

[9] Thomas C. Redman. *Data quality for the information age*. Artech House, Inc, 1997.

[10] Yair Wand and Richard Y. Wang. Anchoring data quality dimensions ontological foundations. *COMMUNICATIONS OF THE ACM*, page 10, November 1996.

[11] Lorena Etcheverry, Véronika Peralta, and Mokrane Bouzeghoub. Qbox-foundation : a metamodele platform for quality measurment. *Qualité de donnée et de connaissance*, page 10, 2008.

[12] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. The goal question metric approach. page 10, 1994.

[13] Panos Vassiliadis, Mokrane Bouzeghoub, and Christoph Quix. Towards quality-oriented data warehouse usage and evolution. page 17, 2000.

[14] Fernando Lemos and Mokrane Bouzeghoub. Qolap : Quality on-line analytical processing. Technical report, Laboratoire PRiSM, Université de Versailles, 2009.

[15] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns : Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

[16] Stéphane Tufféry. *Data mining et statistique décisionnelle*. Technip, nouvelle édition edition, mai 2007.

[17] http ://ws.apache.org/juddi/.

[18] http ://jboss.org/jbossws/.

[19] http ://jcp.org/en/jsr/detail ?id=224.

[20] http ://www.jboss.org/.

[21] http ://tomcat.apache.org/.