Rapport de stage L1 CMI Informatique (2016-2017)

Développement d'un utilitaire de vérification et de reporting de la qualité des données en base

ERDIL Databases Quality Checker (EDQC)

Claire MARCHE

Tuteur au sein de l'entreprise : M. Arnaud WITSCHGER
Tuteur universitaire : M. Fabien PEUREUX
Stage effectué du 6 juin au 13 juillet 2017

Remerciements

Je remercie l'ensemble des membres de l'entreprise ERDIL de m'avoir accueillie dans un cadre amical.

Je remercie particulièrement M. Arnaud Witschger de m'avoir tutorée au sein de l'entreprise, ainsi que d'avoir relu ce rapport.

Pour cette même raison, je remercie aussi mon tuteur universitaire, M Fabien Peureux.

Je tiens également à remercier les professeurs de l'université pour leurs enseignements qui ont pu m'être utiles au cours de ce stage.

Table des matières

Table des figures

Glossaire

HTML Hyper Text Markup Language. Langage de présentation de document.

CSS Cascading StyleSheets. Langage de mise en page de documents HTML.

Java Langage de programmation orienté objet, actuellement développé par Oracle.

H2 Système de gestion de bases de données SQL développé entièrement en Java.

Eclipse Environnement de développement intégré (IDE)open source Java.

SQL Langage permettant d'effectuer des requêtes sur une base de données.

MySQL Système de gestion de bases de données.

verbatim Retour client qui peut être un commentaire spontané, une réponse à une enquête...

Introduction

Dans le cadre du Cursus Master en Ingénierie (CMI) de l'Université de Franche Comté, nous avons l'opportunité d'effectuer un stage de première année d'une durée de six semaines (du 6 juin au 13 juillet 2017) dans l'entreprise ERDIL, à Besançon.

ERDIL est une entreprise qui mêle informatique et linguistique pour l'étude de verbatim clients. Le projet décrit dans ce stage a été réalisé au sein de l'équipe de développement informatique de l'entreprise, en autonomie.

Ce projet consiste en un développement permettant d'analyser et de rendre compte de la qualité des données dans les bases clients. Ce développement est désigné par les initiales EDQC pour ERDIL Databases Quality Checker.

Afin de présenter le travail de manière structurée, nous allons tout d'abord présenter l'entreprise ERDIL et sa place sur le marché dans un premier chapitre, puis le second chapitre introduira la mission qui a été confiée. La démarche entreprise pour sa réalisation fera l'objet du troisième chapitre, puis on détaillera les résultats obtenus dans le quatrième. Enfin, un bilan de cette expérience viendra conclure ce rapport.

Chapitre 1

L'entreprise : ERDIL et ses missions

ERDIL est une entreprise qui propose des services dans le domaine de l'analyse de verbatim, qui joint les secteurs de l'informatique et de la linguistique.

1.1 L'entreprise

ERDIL (Entreprise de Recherche et Développement en Informatique et linguistique) vit le jour il y a dix ans en tant que SAS (Société par Actions Simplifiées), suite à un projet récompensé au concours national d'aide à la création d'entreprises de technologie innovantes et soutenu notamment par le Ministère de la Recherche. Elle compte dix-neuf salariés et est située au 6, rue Sophie Germain, au sein du Pôle Temis à Besançon. Elle a cette année atteint et dépassé un million de chiffre d'affaire.

Le service principal offert par ERDIL est une réponse au besoin spécifique qu'est l'analyse de verbatim clients (voir Glossaire). Dans ce cadre, ERDIL a l'opportunité de travailler avec de grands groupes, dans un nombre important de secteurs d'activités. On compte par exemple parmi ses clients PSA, LCL, la Française des Jeux ou encore ENGIE[1].

Le besoin d'une entreprise telle qu'ERDIL pour l'analyse de verbatim s'explique facilement par la quantité de données concernant les retours clients qui y transitent. En effet, toute analyse humaine d'une telle quantité de données serait sinon impossible, du moins très inefficace, contrairement au traitement informatique réalisé par ERDIL, qui assure une fiabilité certaine.

De plus, les données fournies par les clients peuvent être, après analyse, renvoyées sous forme de données quantifiées et d'analyses structurées aidant à l'interprétation et à la prise en compte des avis clients. Il est également possible d'exploiter les données fournies sous une forme graphique avec des services

1. Liste complète des clients d'ERDIL : https ://www.erdil.fr/customers

fournis : Esatis et E-Letters.

ERDIL est divisée en trois grands pôles :

— L'équipe linguistique dont le rôle consiste principalement en l'élaboration de grammaires linguistiques et en l'analyse proprement dite des retours clients.

— L'équipe de développement, s'occupant à la fois des projets internes et de la création de nouvelles solutions pour les clients (interface...).

— L'équipe d'administration système et réseaux, dont les tâches principales sont le déploiement, la maintenance et la supervision de tous les services proposés par ERDIL.

1.2 Les activités informatiques de l'entreprise

ERDIL fournit à ses clients tout un panel d'outils permettant d'observer les statistiques de leurs verbatims en fonction de plusieurs critères, tels que la localisation ou la date, et qui permet de produire un rapport existant sous plusieurs formes (PDF...) contenant les statistiques sur les retours des clients de l'entreprise faisant appel à ERDIL.

Les bases contenant les messages clients sont également administrés au sein d'ERDIL. Leur organisation générale varie. Dans la majorité des cas, un projet unique est contenu dans une seule base de données. Cependant, pour faciliter les comparaisons, certains projets du même client sont contenus dans la même base. Au contraire, certains projets de grande envergure sont stockées sur plusieurs bases de données, afin de marquer une scission plus claire pour différents critères (localisation...). Ces bases sont constituées pour toutes d'informations proprement dites fournies par le client et de données techniques. Pour toutes les bases, une table contient des colonnes qui sont toujours les mêmes, ainsi que des colonnes qui varient d'un projet à l'autre. Celles-ci sont dites les "paramètres". Pour celles-ci, il s'agit toujours de champs textes nommés génériquement.

1.3 Les besoins ou améliorations possibles

ERDIL conserve et administre actuellement plus de trois cents bases clients et ce nombre est en constante augmentation. Chacune de ces base contient une quantité importante de données, qui sont mises à jour très régulièrement à un débit important. Cependant les données fournies par les clients ne sont pas toujours sans défaut : c'est ici que le projet de ce stage intervient.

Par exemple, quelquefois, les données clients présentent des doublons, c'est à dire des informations présentes plusieurs fois. Ces doublons sont de plusieurs types ; ce type dépend du nombre des informations qui sont en double et influe sur la manière de les détecter.

Afin de détecter ces problèmes, il est nécessaire d'exécuter des requêtes périodiquement sur les bases afin de repérer les incohérences de la structure de données et d'en informer le client afin qu'il puisse éventuellement corriger ce qui

cause le dysfonctionnement en rectifiant le flux client vers ERDIL. Ces requêtes sont effectuées périodiquement, mais manuellement sur toutes les bases, ce qui prend du temps et n'est pas forcément optimal pour la vérification. L'entreprise cherche donc à automatiser ce processus.

En résumé, il y a trois grandes attentes du produit de ce projet : sa maintenabilité, la sécurité des données qui y seront traitées et la production d'un rapport clair et actualisé. Ces objectifs seront décrits plus en détail dans le chapitre suivant.

Chapitre 2

La mission : utilitaire de vérification de la qualité des données clients ou EDQC

La mission qui nous a été confiée va ici être décrite plus en détail, en évoquant tout d'abord les objectifs desquels elle est constituée, puis l'organisation du travail qui a été adoptée.

2.1 Descriptif des objectifs et attentes précises de la mission, cahier des charges

L'entreprise a donc besoin d'un utilitaire permettant d'automatiser la vérification de la qualité des données contenues dans les bases de données des clients. Ce projet, nommé, projet EDQC pour ERDIL Databases Quality Checker nous a été confié.

Le projet doit être réalisé en utilisant les technologies habituelles à l'entreprise. Il doit donc interagir avec les bases de données de l'entreprise, qui sont des bases de type MySQL. Il doit être capable de produire un rapport lisible au format HTML. Il est également impératif que le projet soit codé en Java pour la maintenabilité future. De plus, il devra être facile pour l'utilisateur d'ajouter ou de modifier les requêtes dont l'exécution est demandée, ainsi que les bases sur lesquelles celles-ci devront être exécutées.

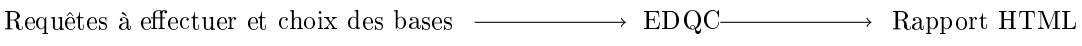


FIGURE 2.1 – Schéma du fonctionnement prévu de EDQC

2.1.1 Maintenabilité

Le projet doit être facilement maintenable et utilisable, ce qui signifie plusieurs choses.

Tout d'abord il doit comporter la documentation expliquant son fonctionnement à la fois à l'utilisation (aide à l'utilisation) et en interne (code commenté de manière compréhensible et explicite).

Ensuite, il doit être codé proprement et de manière compréhensible afin que d'éventuelles modifications puissent y être apportées rapidement et facilement.

De plus, pour des raisons pratiques, l'utilitaire doit pouvoir être exécuté sans recompilation systématique de son programme principal, même en cas de requêtes ajoutées ou modifiées ou de bases en plus à analyser. Cela signifie donc les informations nécessaires à la vérification ne devront pas être contenues dans le code lui-même mais externes à celui-ci. Ainsi, si on peut se contenter de sortir des blocs de codes les informations qui en principe ne sont pas susceptibles d'être modifiées pour toute exécution en les gardant dans le code du programme en les rendant plus accessibles, les informations de configurations devront être contenues dans des fichiers totalement indépendants du programme et de sa compilation. Par exemple, si on ajoute une base au nombre de celles qui sont contenues dans le rapport, on doit pouvoir exécuter l'utilitaire immédiatement sans le recompiler.

Enfin, le programme sera amené à être utilisé à la fois sur Windows et sur Debian/Linux, il doit donc être parfaitement portable (chemins non spécifiques à un OS...).

2.1.2 Sécurité

Les informations contenues dans les bases de données clients sont bien évidemment confidentielles, il faut donc les protéger au maximum, que ce soit en protégeant les mots de passe de connexion ou en protégeant les données elles-mêmes d'ajouts ou de délétions volontaires ou accidentelles (erreur dans la configuration des requêtes...). Ainsi, il faut donc protéger les données par la validation des entrées de l'utilisateur lors de la configuration.

2.1.3 Reporting

Le rapport produit doit contenir des informations actualisées et différencier celles-ci des informations déjà contenues dans les rapports précédents. Il devra de plus être compréhensible directement par des clients externes à l'entreprise.

2.2 Organisation et planification du travail

Le projet ne comportant pas une contrainte forte de temps, plaçant une plus grande priorité sur la maintenabilité du code par l'équipe, et étant seule à développer ce projet, une planification stricte n'a pas été jugée nécessaire. De même, les réunions de suivi ont pris la forme d'un point informel effectué de

manière hebdomadaire : une rapide révision du code ainsi qu'une discussion sur la marche à suivre pour la suite.

2.3 Contexte technique

Le poste sur lequel le projet doit être développé est un terminal Windows 10. Le projet doit être réalisé en Java pour une meilleure portabilité et interagir avec les bases relationnelles MySQL de l'entreprise.

Chapitre 3

Le développement

Le développement de ce projet consiste, en grande partie, en la recherche de réponses aux trois problématiques évoquées précédemment (2.1) : la maintenabiilté, la sécurité et le reporting de données actualisées.

3.1 Maintenabilité

La capacité du code à être exécuté sur des bases différentes, mais de structures similaires, en utilisant des requêtes potentiellement différentes est une des fonctionnalités de bases du programme. Il fallait donc en priorité trouver la manière de résoudre cette problématique.

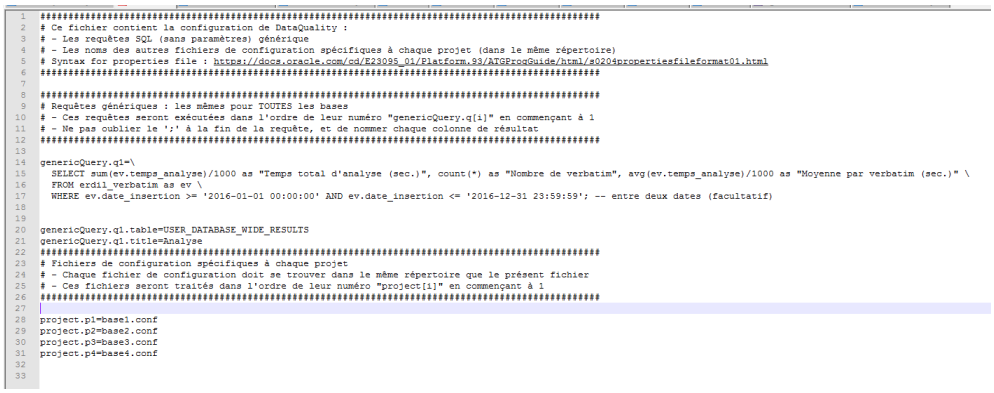
Une première approche avait déjà été réalisée, utilisant des fichiers à l'extension .conf, c'est à dire des fichiers de propriétés au sens Java, c'est-à-dire contenant des couples (noms de propriété, valeur) mais la liberté en ce qui concerne le choix de la méthode de stockage dans les fichiers externes était totale.

Nous avons choisi de poursuivre sur cette voie proposée par notre tuteur, M. Witschger. Ainsi, les fichiers de propriétés dans lesquels se trouvent les informations susceptibles d'être modifiées entre les exécutions successives de l'utilitaire suivent le schéma suivant :

— Un unique fichier Main.conf (figure 3.1) dans lequel sont listés les fichiers spécifiques à chaque base ainsi que les requêtes génériques à effectuer sur toutes les bases.

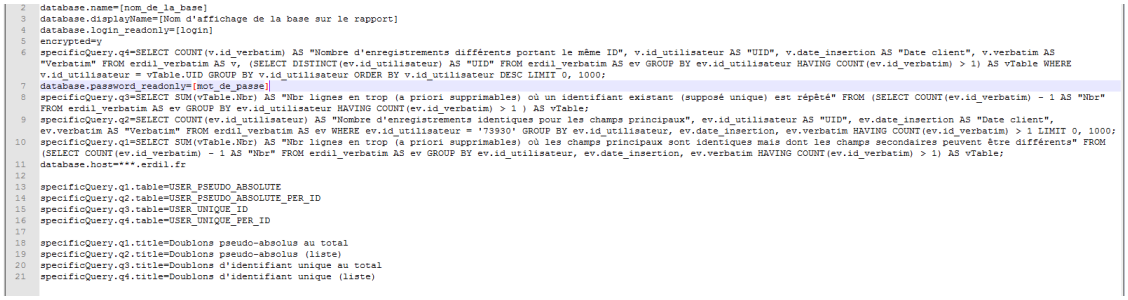
— Dess fichier spécifiques (figure 3.2) à chaque base client à analyser, contenant à la fois les informations de connexion à la base concernée, c'est à dire login, mot de passe, nom de la base et hôte, le nom d'usage de cette base permettant de l'identifier dans le rapport produit par l'utilitaire et les requêtes spécifiques à cette base.

Dans ces deux types de fichiers, chaque requête est désignée par un préfixe de nom de requête qui dépend du fichier dans lequel elle se trouve (`genericQuery.q` dans le Main.conf et `specificQuery.q` dans les autres fichiers), suivie d'un numéro qui lui est attribué séquentiellement (les numéros de toutes les requêtes



```
##########################################################################################################
# Ce fichier contient la configuration de DataQuality :
# - Les requêtes SQL (sans paramètres) générique
# - Les noms des autres fichiers de configuration spécifiques à chaque projet (dans le même répertoire)
# Syntax for properties file : https://docs.oracle.com/cd/E23095_01/Platform.93/ATGProgGuide/html/s0204propertiesfileformat01.html
##########################################################################################################

##########################################################################################################
# Requêtes génériques : les mêmes pour TOUTES les bases
# - Ces requêtes seront exécutées dans l'ordre de leur numéro "genericQuery.q[i]" en commençant à 1
# - Ne pas oublier le ';' à la fin de la requête, et de nommer chaque colonne de résultat
##########################################################################################################

genericQuery.q1=\
  SELECT sum(ev.temps_analyse)/1000 as "Temps total d'analyse (sec.)", count(*) as "Nombre de verbatim", avg(ev.temps_analyse)/1000 as "Moyenne par verbatim (sec.)" \
  FROM erdil_verbatim as ev \
  WHERE ev.date_insertion >= '2016-01-01 00:00:00' AND ev.date_insertion <= '2016-12-31 23:59:59'; -- entre deux dates (facultatif)


genericQuery.q1.table=USER_DATABASE_WIDE_RESULTS
genericQuery.q1.title=Analyse
##########################################################################################################
# Fichiers de configuration spécifiques à chaque projet
# - Chaque fichier de configuration doit se trouver dans le même répertoire que le présent fichier
# - Ces fichiers seront traités dans l'ordre de leur numéro "project[i]" en commençant à 1
##########################################################################################################

project.p1=base1.conf
project.p2=base2.conf
project.p3=base3.conf
project.p4=base4.conf
```

FIGURE 3.1 – Le fichier Main.conf listant toutes les bases et toutes les requêtes génériques



```
database.name=[nom_de_la_base]
database.displayName=[Nom d'affichage de la base sur le rapport]
database.login_readonly=[login]
encrypted=y
specificQuery.q4=SELECT COUNT(v.id_verbatim) AS "Nombre d'enregistrements différents portant le même ID", v.id_utilisateur AS "UID", v.date_insertion AS "Date client", v.verbatim AS
"Verbatim" FROM erdil_verbatim AS v, (SELECT DISTINCT(ev.id_utilisateur) AS "UID" FROM erdil_verbatim AS ev GROUP BY ev.id_utilisateur HAVING COUNT(ev.id_verbatim) > 1) AS vTable WHERE
v.id_utilisateur = vTable.UID GROUP BY v.id_utilisateur ORDER BY v.id_utilisateur DESC LIMIT 0, 1000;
database.password_readonly=[mot_de_passe]
specificQuery.q3=SELECT SUM(vTable.Nbr) AS "Nbr lignes en trop (a priori supprimables) où un identifiant existant (supposé unique) est répété" FROM (SELECT COUNT(ev.id_verbatim) - 1 AS "Nbr"
FROM erdil_verbatim AS ev GROUP BY ev.id_utilisateur HAVING COUNT(ev.id_verbatim) > 1 ) AS vTable;
specificQuery.q2=SELECT COUNT(ev.id_utilisateur) AS "Nombre d'enregistrements identiques pour les champs principaux", ev.id_utilisateur AS "UID", ev.date_insertion AS "Date client",
ev.verbatim AS "Verbatim" FROM erdil_verbatim AS ev WHERE ev.id_utilisateur = '73930' GROUP BY ev.id_utilisateur, ev.date_insertion, ev.verbatim HAVING COUNT(ev.id_verbatim) > 1 LIMIT 0, 1000;
specificQuery.q1=SELECT SUM(vTable.Nbr) AS "Nbr lignes en trop (a priori supprimables) où les champs principaux sont identiques mais dont les champs secondaires peuvent être différents" FROM
(SELECT COUNT(ev.id_verbatim) - 1 AS "Nbr" FROM erdil_verbatim AS ev GROUP BY ev.id_utilisateur, ev.date_insertion, ev.verbatim HAVING COUNT(ev.id_verbatim) > 1) AS vTable;
database.host=****.erdil.fr

specificQuery.q1.table=USER_PSEUDO_ABSOLUTE
specificQuery.q2.table=USER_PSEUDO_ABSOLUTE_PER_ID
specificQuery.q3.table=USER_UNIQUE_ID
specificQuery.q4.table=USER_UNIQUE_PER_ID

specificQuery.q1.title=Doublons pseudo-absolus au total
specificQuery.q2.title=Doublons pseudo-absolus (liste)
specificQuery.q3.title=Doublons d'identifiant unique au total
specificQuery.q4.title=Doublons d'identifiant unique (liste)
```

FIGURE 3.2 – Un exemple de fichier spécifique à une base

dans un fichier devant se suivre et commencer à 1). Elle possèdent également chacune un titre (`title`) qui permet de les identifier dans le rapport produit.

Ces fichiers sont alors lus par une classe utilitaire du projet, `ConfFileReader`, qui permet de lire leur contenu et de le stocker dans des objets `java.util.Properties`, qui contiennent des paires (nom, propriété), format particulièrement adapté, puisque c'est un format identique à celui d'un fichier de propriétés.

3.2 Sécurité

La sécurité des données clients est une autre problématique majeure du sujet. En effet, ces données sont confidentielles et ne doivent pas être modifiées par l'exécution de l'utilitaire, qui doit uniquement consulter leur état et non modifier celui-ci.

La solution à ce problème s'articule en deux parties. Tout d'abord, pour empêcher la modification accidentelle des données dans les bases client malgré

la totale liberté laissée à l'utilisateur dans la définition de ses requêtes SQL dans les fichiers de propriétés décrits ci-dessous, il suffit que les utilisateurs ne puissent accéder à la base qu'en lecture seule. Ensuite, une vérification supplémentaire (certes imparfaite) est effectuée pour vérifier qu'il s'agit bien d'une requête de type `SELECT` : on vérifie si la propriété associée à cette requête commence bien par `SELECT`

De plus, les mots de passe requis pour se connecter aux bases clients et qu'il est nécessaire d'écrire dans les fichiers de propriétés des bases respectives sont également protégés. En effet, lorsque l'utilisateur saisit des informations de connexion à une nouvelle base il saisit également une propriété générique, le flag `encrypted` qui peut prendre pour valeur `y` ou `n` (sa valeur est assumée à `n` si elle est différente de `y`) qui permet de spécifier si le mot de passe est chiffré. Lors de la première saisie du mot de passe en clair, `encrypted` doit valoir `n` (ou tout du moins ne pas valoir `y`). Alors, grâce à une classe `PasswordEncryption` dédiée, le mot de passe est chiffré dans le fichier selon l'algorithme AES-128 en mode CBC avec une clé définie dans celui-ci et `encrypted` est passé à `y` afin d'éviter de chiffrer le mot de passe une seconde fois. L'utilitaire utilise ensuite une autre méthode de la classe `PasswordEncryption` qui permet de renvoyer le mot de passe déchiffré. Le mot de passe reste alors bien sûr sous sa forme chiffrée dans le fichier. De cette façon, s'il y a une fuite des fichiers de propriétés, il sera impossible à partir de ces seuls fichiers d'accéder aux données clients.

3.3 Reporting

Il est nécessaire de présenter un rapport contenant des données actualisées, séparant clairement les nouveaux résultats représentant les problèmes les plus récents trouvés dans les bases clients des résultats déjà évoqués dans les rapports précédents, même s'ils sont encore présents.

Pour cela, nous nous sommes tournés vers une base embarquée en Java (de type H2) dans laquelle sont stockées les données du rapport avant de produire le rapport proprement dit.

Cette base est constituée d'une part de tables pour chaque requête. Ces tables sont génériques, c'est à dire que toutes les tables de la base ont la même structure : un champ contenant le numéro du rapport dans lequel l'information est produite, un champ contenant le nom de la base, dix champs textes neutres pouvant accueillir toutes les données renvoyées par la requête effectuée, et enfin un champ contenant le nombre de colonnes actives dans le résultats, c'est à dire le nombre de colonnes renvoyés par la requête de départ (les colonnes qui ne sont pas actives valant `NULL`. Certaines requêtes renvoient également des résultats spécifiques à l'identifiant unique ou UID (contenu du verbatim désigné par cet UID,...). Dans ce cas, la clé primaire est composée du numéro du rapport, du nom de la base et de cet UID. Sinon la clé primaire est simplement composée du numéro du rapport et du nom de la base (voir figure 3.3).

D'autre part, cette base contient également des tables "références", une ne contenant que les numéros de rapport déjà existant et l'autre ne contenant que

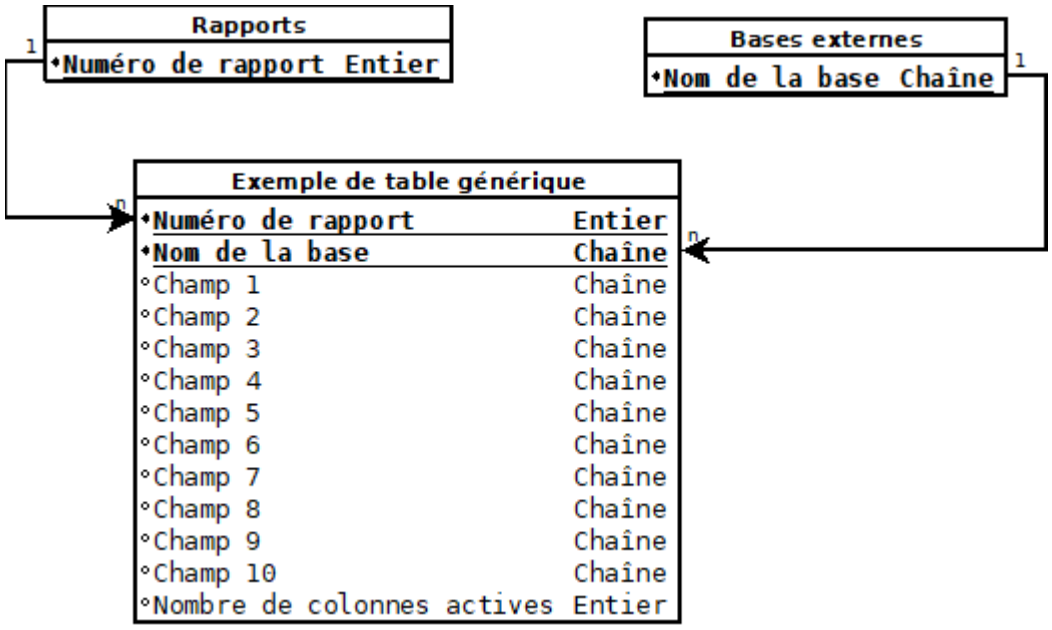


FIGURE 3.3 – Schéma de la base embarquée

les noms de bases déjà analysées, ceci afin de faciliter la modification. Les champs contenant le nom de la base et le numéro du rapport sont tous deux des clés étrangères faisant respectivement référence à ces deux tables. (modèle présenté figure 3.3).

De plus, toute base référencée dans les fichiers de propriétés mais qui ne serait pas présente dans la table contenant les noms des bases de données y est ajoutée automatiquement, et un numéro de rapport égal au numéro du rapport précédent (assumé être le numéro le plus grand de la table des numéros de rapport) auquel on ajoute un est inséré dans la table des numéros de rapports afin de représenter le rapport courant.

SELECT * FROM USER_TABLE_GENERIQUE;

REPORTNUMBER	DATABASENAME	FIELD1	FIELD2	FIELD3	FIELD4	FIELD5	FIELD6	FIELD7	FIELD8	FIELD9	FIELD10	NUMBEROFACTIVEFIELDS
1	Nom de la base	champ générique capable de contenir toute information	autre champ générique contenant une information quelconque	*null*	*null*	*null*	*null*	*null*	*null*	*null*	*null*	4

FIGURE 3.4 – Structure d'une table générique

Pour chaque requête (soit pour chaque table de la base embarquée, puisqu'à chaque requête des fichiers de configuration correspond une table de la base embarquée) et pour chaque base, on récupère ensuite tous les résultats liés à un numéro de rapport inférieur à celui du rapport courant (les anciens résultats), et les résultats liés au numéro de rapport courant auxquels on soustrait les anciens

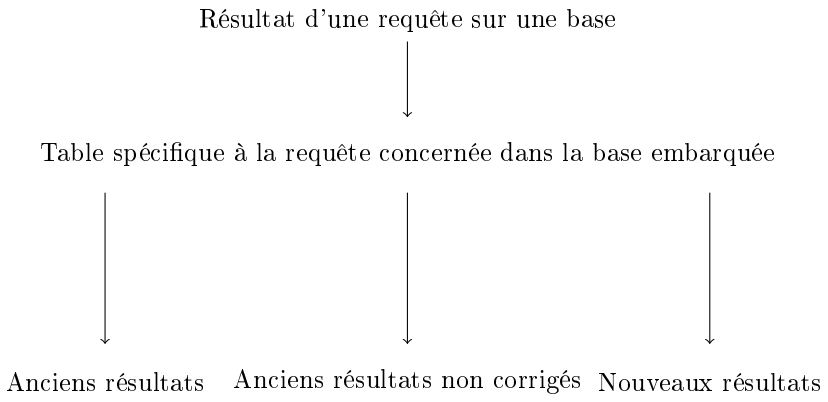


FIGURE 3.5 – Schéma du fonctionnement de la base embarquée

résultats (ce sont les nouveaux résultats).

Enfin, on récupère, également pour chaque requête et pour chaque base les résultats qui n'ont pas été corrigés depuis le dernier rapport. Pour cela, on cherche l'intersection de tous les résultats ayant pour numéro de rapport le numéro de rapport courant et des anciens résultats. Le résultat de cette requête contient tous les résultats ayant été déjà contenus dans un rapport précédent mais qui sont toujours présents au moment de la création du rapport courant, c'est-à-dire les résultats déjà évoqués lors des précédents rapports mais n'ayant pas été corrigés depuis.

Ce processus entier est schématisé dans la figure 3.5.

Dans le rapport en HTML proprement dit, la distinction entre ces trois types de résultats est représentée par une différence de classe, ce qui est traduit par une différence dans le style CSS lorsque les deux résultats sont affichés. D'autre part, grâce à un script javascript simple, il est possible de masquer (figure 3.6)ou d'afficher (figure 3.7) les anciens résultats ou les résultats non corrigés (les nouveaux résultats étant toujours visibles, bien que changeant de style lorsqu'un autre type de résultat apparaît), ces deux types étant mutuellement exclusifs ; tout affichage de l'un masque automatiquement l'autre.

Un autre modèle de base embarquée ayant le nombre exact de colonnes nécessaire pour chaque requête avait d'abord été envisagé, mais il nécessitait la création manuelle de ses tables et a donc été abandonné.

3.4 Synthèse technique

Le projet final est constitué de plusieurs classes :

— une classe utilitaire permettant la lecture des fichiers de propriétés

— une classe utilitaire permettant le chiffrement et le déchiffrement des mots de passe de ceux-ci

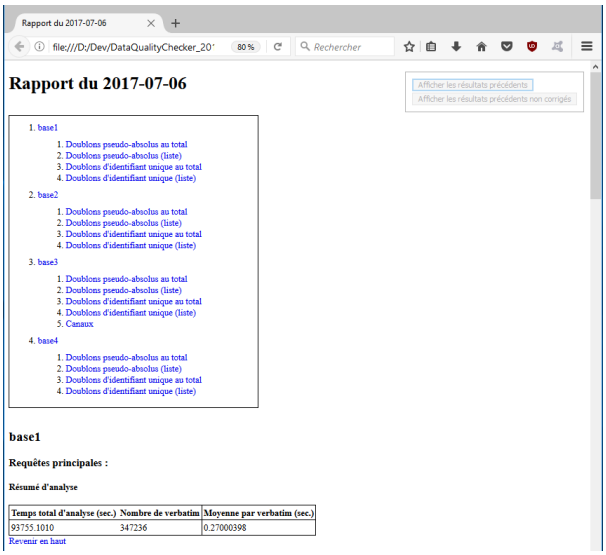


FIGURE 3.6 – Exemple de rapport dans lequel ne sont affichés que les nouveaux résultats

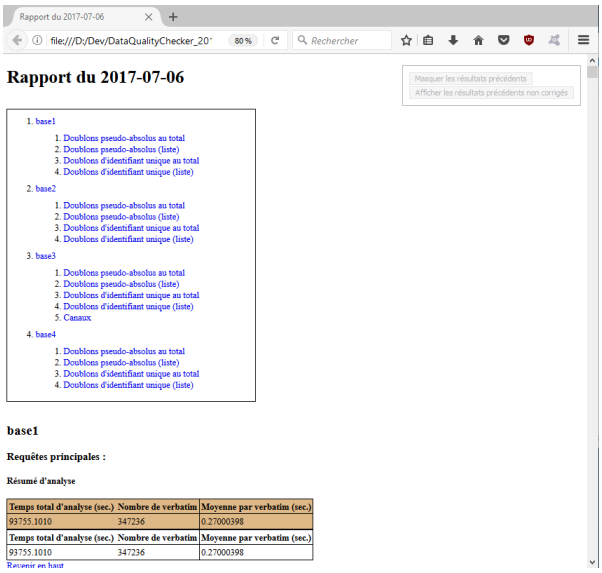


FIGURE 3.7 – Exemple de rapport dans lequel les nouveaux et les anciens résultats sont tous deux affichés

— une classe représentant les bases de données et la connexion à celles-ci en général
— une classe dérivée de la classe représentant les bases de données pour les bases externes clients MySQL
— une classe dérivée de la classe représentant les bases de données pour la base embarquée H2
— une classe représentant les résultats des requêtes
— une classe représentant des tableaux de ces résultats pour permettre des appels multiples aux méthodes de la classe représentant les résultats
— la classe principale, qui permet la production d'un rapport.

Le poste sur lequel le développement a été effectué est un terminal Windows 10. Le projet DQC a été développé en Java. Il nous a été permis d'utiliser une IDE pour ce fait. Notre choix s'est porté sur Eclipse. De plus, afin d'"habiller" quelque peu le rapport en sortie, le rapport comprend également une feuille de style CSS, réalisée à l'aide des outils de développeur du navigateur Mozilla Firefox, ainsi qu'un script Javascript simple, codé à l'aide de l'éditeur de texte Notepad++. Les bases de données de l'entreprise sont des bases MySQL, et la base embarquée utilisée dans l'optique de comparaison du rapport avec lui même utilise le système de gestion de bases de données H2.

Ce développement considérant les trois problématiques précédentes, a pu répondre aux objectifs demandés.

Chapitre 4

Les résultats du projet

Nous allons ici étudier dans quelle mesure l'utilitaire développé répond aux trois contraintes décrites dans la section 2.1.

4.1 Maintenabilité

Le résultat du projet permet bien, grâce aux fichiers de propriétés externes au code proprement dit, de modifier et d'ajouter les requêtes demandées et d'ajouter des bases, tout ceci sans recompilation, puisque leur lecture est effectuée à chaque début d'exécution du programme, et que celui-ci se base sur les résultats de cette lecture tout au long de son exécution.

De plus, toutes les valeurs littérales utilisées dans le code sont situées dans des constantes (statiques) placées au sommet de toutes les classes constituant le projet. Ainsi, en cas de besoin d'une éventuelle modification du code source, il est aisé de repérer la partie qu'on souhaite modifier.

Enfin, dans un souci de portabilité, tous les chemins décrivant les destinations des fichiers sont des chemins relatifs et indépendants de la plateforme. En effet, EDQC étant amené à être utilisé sur des machines sur Debian, un chemin absolu dans des répertoires Windows était exclu.

4.2 Sécurité

Les mots de passe des bases de données ne sont pas en clair dans les fichiers de propriétés, mais dans une forme chiffrée qui ne peut être déchiffrée que grâce à une clé maître qui n'est contenue nulle part dans les fichiers externes à l'application. De plus, ces mots de passe associés aux logins contenus dans les fichiers ne permettent d'accéder qu'à une session en lecture seule, ce qui empêche toute modification accidentelle ou mal intentionnée des données clients au travers de ce programme.

Pour augmenter encore la sécurité, toutes les entrées de l'utilisateur par les fichiers de propriétés sont vérifiées par des méthodes internes aux classes

concernées, ce qui limite tout ce que l'utilisateur peut faire au travers de DQC, et donc les "dégâts" qu'il peut causer.

4.3 Reporting

Le rapport produit contient trois types de résultats : les résultats qui viennent d'apparaître lors de la création du rapport courant, les résultats qui étaient présents lors des anciens rapports en général, et ceux qui sont présents lors de la création du rapport courant et qui étaient déjà présents lors du rapport précédents, c'est à dire en substance les résultats qui n'ont pas été corrigés depuis leur apparition. On peut choisir de n'afficher que les résultats récents (figure 3.6), ou d'afficher les résultats récents avec un des autres types de résultats, le style permettant alors de les différencier (figure 3.7).

Ainsi, on obtient une séparation entre les différents types de résultats et une vision claire de l'état de la base et des nouveaux problèmes qui y sont détectés par DQC.

4.4 Perspectives d'améliorations

Il serait possible dans le futur de ne pas demander la saisie du nom des tables de la base embarquée à l'utilisateur mais d'attribuer directement à chaque requête et pour chaque fichier de propriété un identifiant unique dépendant soit uniquement de l'identifiant de la requête si elle provient du fichier principal ou du nom de la base et de l'identifiant de la requête sinon.

Il serait également possible de ne pas fixer dans le code la position du fichier principal mais de le passer en paramètre à l'appel du programme. Ainsi, il serait possible d'avoir plusieurs fichiers principaux contenant chacun une liste de fichiers spécifiques à des bases qui leur serait propre.

Bilans

4.5 Bilan technique

Le sujet de ce stage a été le développement d'un utilitaire de vérification et de reporting de la qualité des données en bases répondant à un réel besoin de l'entreprise. Cet objet a été développé en Java, interagit avec des bases de donnes relationnelles (MySQL) et produit un rapport détaillé en HTML.

Trois objectifs principaux sont remplis par le projet : sa maintenance et son exécution doivent être aisées, il doit garder sécurisées les données avec lesquelles il interagit et permettre d'avoir une vision claire des problèmes dans les données dans le temps en étant actualisé.

4.6 Bilan personnel

Ce stage a pu apporter des connaissances sur l'utilisation de différentes API Java, sur l'utilisation de l'IDE Eclipse ainsi que sur la programmation orientée objet et plus particulièrement sur la modélisation orientée objet, dont je n'avais qu'une connaissance très superficielle et théorique, car il s'agissait d'un premier véritable projet utilisant cette technique. Il a aussi pu attirer l'attention sur des contraintes comme la sécurité et la maintenabilité d'un produit, souvent présentes dans le développement professionnel. Il a également permis d'observer la vie d'une entreprise et notamment la participation à des évènements importants tels que le dixième anniversaire d'ERDIL. Enfin, il a permis d'entrevoir la réalité du monde du travail (suivi des directives des supérieurs, contraintes à respecter...)

De manière générale, les connaissances acquises lors de ce stage pourront certainement être utiles dans le futur, que ce soit au cours de mes études ou au cours de mes expériences professionnelles.

Résumé

Ce stage a été effectué dans l'entreprise ERDIL, ayant pour secteur d'activité l'informatique et la linguistique, en ce qui concerne le domaine de l'analyse de verbatims.

Le stage a eu pour but de développer en Java un producteur de rapport HTML sur la qualité des données clients contenues dans les bases de type MySQL de l'entreprise en listant les différents problèmes de cohérence qui y auront été trouvés dans le rapport. Celui-ci a pu permettre l'automatisation des requêtes qui étaient auparavant exécutées manuellement sur les bases de l'entreprise.

Mots-clé : Java, SQL, HTML, bases de données MySQL, qualité des données, production de rapport

Abstract

This internship took place in the company ERDIL, whose main lines of business are computer development and linguistics, as part of verbatim analysis.

This internship's goal was to develop in Java an HTML report producer about data quality in the company's client MySQL databases, by listing all the different coherency problems that would have been found there in the produced report. This software allowed the queries, which were before that manually executed on the company's bases, to be automatized.

Keywords: Java, SQL, HTML, MySQL databases, data quality, report production