UMONS
Université de Mons
Faculté des Sciences
Département d'Informatique

Mémoire :
Database Repairing with Respect to Functional Dependencies

Directeurs : M^r^ Jef Wijsen

Projet réalisé par
Sophie Opsommer

Rapporteur :

Année académique 2018-2019

UMONS
Université de Mons

Faculté des Sciences

Mémoire : Database Repairing with Respect to Functional Dependencies

Mémoire réalisé dans le cadre
du Master en Sciences Informatiques, à finalité spécialisée

Réalisé par

OPSOMMER Sophie SOPHIE.OPSOMMER@student.umons.ac.be

Sous la direction de: *Directeur* : J. WIJSEN

2018-2019

Faculté
des Sciences

ACADÉMIE
UNIVERSITAIRE
WALLONIE-
BRUXELLES

Ce *rapport de projet* est rendu dans le cadre du cursus de « Master en sciences informatiques, à finalité spécialisée ». Le but de ce rapport est de présenter le problème qui est posé, de développer l'analyse du problème et les choix réalisés ainsi que d'exposer la réalisation.

REMERCIEMENTS

Table des matières

Chapitre 1

Introduction

...

Chapitre 2

Contexte

(deviendra peut-être l'introduction)

À l'heure où le monde collecte et enregistre de plus en plus de données et qu'il tente d'en tirer des conclusions et des prédictions, vérifier l'exactitude des informations devient aussi de plus en plus importante. Il existe beaucoup de critères à vérifier et beaucoup de méhodes pour y arriver.

Interessons-nous à l'un d'entre eux : les dépendances fonctionelles.

Dans la pratique, il peut arriver que des dépendances ne sont pas respectées. Les causes sont multiples :

- Une manipulation volontaire (pour une situation d'exception),
- Une erreur manuelle d'encodage ou de manipulation involontaire,
- Une source de données imprécise (par le biais d'internet, de capteurs avec des marges d'erreurs,...),
- Une méthode de collecte imprécise (reconnaissance vocale, analyse de signaux, d'image,...)
- Un rappatriement de données provenant de plusieurs sources avec des données différente ou avec des formats différents,
- Une mauvaise vérification au niveau applicatif,
- L'utilisation de plusieurs applications qui ne respectent pas toutes les mêmes règles et procédures,
- ...

Il est important de corriger la situation (et de trouver la méthode optimale pour y arriver) parce que la qualités des informations joue un rôle important dans la qualités du traitement des données ultérieurement. Comme le dit le jargon/proverbe : « shit in, shit out ». Ce qui revient à exprimer que si les données avec lesquelles on travaille ne sont pas credible, les conclusions que l'on pourra en tirer ne le seront pas non plus. En cas d'audit, ça permet aussi de calculer le degré ?d'inconsistence ? de la table (on calculera la distance entre la base de données initiale et corrigée).

Pour remédier à ces erreurs, il faut d'une part remédier à la cause de ces incohérences mais aussi corriger les données qui sont incorrecte ou qui ont été corrompue. Pour la correction, il y a 2 solutions. Soit modifier les requêtes pour que celles-ci ne renvoient que les lignes non concernées par ces incohérences. Soit il faut trouver une base de données à l'image de la base de données existante qui, après un minimum de changements, respecte les contraintes d'intégrités. Pour cette dernière solution, deux possibllités existent également. On peut supprimer les données qui sont incorrecte ou on peut modifier les données incorrectes.

Dans la suite, nous traiterons d'une méthode possible pour corriger les données existantes par le biais de la suppression des données incohérentes avec les dépendances fonctionnelles.

2.1 Notions de bases

Avant de rentrer dans le vif du sujet, voici un bref rappel de certaines notions de base. Toute personne habituée à travailler dans le domaine des bases de données peut alègrement sauter cette section et passer à la suivante.

2.1.1 Une contrainte d'intégrité :

Les contraintes d'intégrités sont les assertions qui doivent être vérifiées par les données contenues dans la base de données. Elles regroupe différentes catégories comme :

les intégrités de domaine : les valeurs d'un attribut doivent appartenir au domaine de valeurs de l'attribut.

les intégrités de clé : les clés d'une tables sont uniques et non nulle.

les intégrités référentielle : les clés étrangères sont nulles ou référence une clé primaine existante.

2.1.2 Une dépendance fonctionnelle :

Une dépendance fonctioenlle est une contrainte sémantique. Elle est supposée toujours vrai dans le monde réel.

C'est une relation entre plusieurs attributs d'une même table. Elle definit la valeur d'un attribut (ou d'un ensemble d'attributs) en fonction d'un autre ou de plusieurs autres attributs. En effet la dépendance fonctioennelle $X \rightarrow Y$ définit que si plusieurs lignes ont les mêmes valeurs pour un ensemble X d'attributs, elles auront également les mêmes valeurs pour l'ensemble Y d'attributs. En terme de notation, on dira : soit la table de référence T définit par la relation $Ra_1, a_2, a_3, ..., a_n$ avec n le nombre d'attribut et X et Y des sous-ensemble de R. On dit que « Y dépend fonctionnellement de X »ou « X détermine Y »0si à chaque valeur de X correspond une valeur unique de Y. On écrit alors « $X \rightarrow Y$ ».

Par exemble, la table illustrée dans la figure 2.1 pourrait exister dans une école. Elle illustre les différents professeurs, la classe (emplacement physique) où ils enseignent à un jour et une plage horraire précise ainsi que les élèves qui seront présent. Avec les lois spacio-temporrelle que nous connaissons, un professeur ne peut se trouver que dans une classe à la fois. En terme de dépendance, cela signifie qu'à un jour de la semaine, une plage horaire et un professeur ne peut donc correspondre qu'une seule classe. Notons par contre que des élèves d'années différentes peuvent y assister en même temps. Inversément, une classe n'est utilisée que par un professeur à

	Professeur	Classe	Jour	Heure	Eleves
1	Wijsen	A1	Lundi	8h	BA1
2	Wijsen	A1	Lundi	8h	Anneé complémentaire
3	Wijsen	A1	Lundi	10h15	MA1
4	GILLIS	A1	Lundi	13h15	BA2
5	TUYTTENS	A1	Lundi	15h30	BA2

FIGURE 2.1 – Extrait d'une table pour illustrer des dépendances fonctionnelles

la fois. De la même façon, des élèves ne peuvent se trouver que dans une seule classe à la fois Ces dépendances s'écrivent : $\{Eleves, Jour, Heure \rightarrow Classe\}, \{Professeur, Jour, Heure \rightarrow Classe\}, \{Classe, Jour, Heure \rightarrow Professeur\}$.

Une dépendance fonctionnelle élémentaire

Une dépendance fonctionnelle élémentairetel $X \rightarrow A$ est un dépendance fonctionnelle où A est un attribut unique non inclus dans X et où il n'existe pas d'ensemble d'attribut X' inclus dans X tel que $X' \rightarrow A$.

Une clé (primaire)

Une clé est un attribut (ou un groupe minimum d'attributs) qui définit tous les autres attributs de la relation.

2.1.3 Une opération

Une opération est une manipulation qui est efectuée sur un ensemble de données pour obtenir un ou plusieurs nouvel ensemble de données.

L'union

L'union de deux relations R1 et R2 de même schéma est une relation R3 de schéma identique qui a pour n-uplets les n-uplets de R1 et de R2. On notera : $R3 = R1 \bigcup R2$

L'intersection

L'intersection entre deux relations R1 et R2 de même schéma est une relation R3 de schéma identique ayant pour n-uplets les m-uplets communs à R1 et R2. On notera : $R3 = R1 \bigcap R2$

La différence

La différence entre deux relations R1 et R2 de même schéma est une relation R3 de schéma identique ayant pour n-uplets les n-uplets de R1 n'appartenant pas à R2. On notera : $R3 = R1 - R2$

La projection

La projection d'une relation R1 est la relation R2 obtenue en supprimant les attributs de R1 non mentionnés puis en éliminant éventuellement les n-uplets identiques. On notera : $R2 = \pi R1(A_i, A_j, ..., A_m)$ la projection d'une relation R1 sur les attributs $A_i, A_j, ..., A_m$. Î La projection permet d'éliminer des attributs d'une relation. Elle correspond à un découpage vertical.

La restriction

La restriction d'une relation R1 est une relation R2 de même schéma n'ayant que les n-tuplets de R1 répondant à la condition énoncée. On notera : $R3 = \sigma R1(condition)$ la restriction d'une relation R1 suivant le critaire "condition" où "condition" est une relation d'égalité ou d'inégalité entre 2 attributs ou entre un attribut et une valeur. La restriction permet d'extraire les n-tuplets qui satisfont une condition. Elle correspond à un découpage horizontal.

La jointure

La jointure de deux relations R1 et R2 est une relation R3 dont les n-uplets sont obtenus en concaténant les n-uplets de R1 avec ceux de R2 et en ne gardant que ceux qui vérifient la condition de liaison. On notera : $R3 = R1 * R2(condition)$ la jointure de R1 avec R2 suivant le critère "condition". Le schéma de la relation résultat de la jointure est la concaténation des schémas des opérandes. S'il y a des attributs de même nom qui ne font pas partie de la condition, il faut les renommer. Si ils font partie de la condition, il seront fusionné ne n'apparaîtront qu'une seule fois. Les n-uplets de $R1 * R2(condition)$ sont tous les couples (u1,u2) d'un n-uplet de R1 avec un n-uplet de R2 qui satisfont "condition". La jointure de deux relations R1 et R2 est le produit cartésien des deux relations suivi d'une restriction. La condition de liaison doit être du type : <attribut1> : : <attribut2> où : attribut1 $\in$ 1ère relation, attribut2 $\in$ 2ème relation et : : est un opérateur de comparaison (égalité ou inégalité). La jointure permet de composer 2 relations à l'aide d'un critère de liaison.

La division

Soit deux relations $R1(A_1, A_2, ..., A_n, B_1, B_2, ..., B_m)$ et $R2(B_1, B_2, ..., B_m)$. Si le schéma de R2 est un sous-schéma de R1. La division de R1 par R2 est une relation R3 dont : - le schéma est le sous-schéma complémentaire de R2 par rapport à R1 - un n-uplet (a1 , a2, ... , an) appartient à R3 si (a1, a2, ... , an, b1, b2, ... , bm) appartient à R1 pour tous (b1, b2, ... , bm) $\in$ R2. On notera : $R3 = R1 \div R2$ la division de R1 par R2. La division permet de rechercher dans une relation les sous n-uplets qui sont complétés par tous ceux d'une autre relation. Elle permet de répondre à des questions qui sont formulées avec le quantificateur universel : "pour tout ...".

2.1.4 La fermeture transitive

La fermeture transitive permet d'enrichir une relation à 2 attributs de même domaine, avec tous les couples qui se déduisent par transitivité.

Soit par exemple, 2 attributs A et B qui ont pour domaine les chiffres. Soit une table définie par la relation R(A,B) qui contient les données (1, 2) et (2, 3). La fermeture de cet ensemble est composée de (1, 2) et (2, 3) et (1, 3).

2.2 Notions complémentaires

Ici seront détaillés des notions propre à ce travail qui ne seront peut-être pas commun aux standarts de notations habituels. Il estdonc important de les comprendre avant de poursuivre la suite de la lecture.

2.2.1 Un attribut commun à la main gauche :

est un attribut « A » qui se trouve dans toutes les parties gauches des dépendances fonctionnelles de Δ. Dans l'exemple de la figure **??**, l'attribut »Facility « est donc un attribut commun à la main gauche. En présence de ce type d'attribut, trouver les données de la table initiale qui composeront la solution optimale reviendra à traité independamment les lignes en les groupant par la valeur de cet attribut et ensuite en faisant l'union des sous-ensembles. De cette manière, pour chaque sous-ensemble, cet attribut n'a qu'une seule valeur et peut donc être supprimé dans les dépendances fonctionnelles.

Dans l'exemple de la figure 3.1 cela reviendrait a traiter d'une part les 3 premières lignes et d'autre part la dernière. Les dépendances deviennent donc : $\{\emptyset \rightarrow City\} et \{Room \rightarrow Floor\}$.

2.2.2 Un consensus :

est une dépendance fonctionnelle dont la main gauche est vide : $\emptyset \rightarrow Y$. Cette dépendance fonctionnelle est respectée si la valeur du ou des attributs de Y est toujours identique. En présence de ce type d'attribut, trouver les données de la table initiale qui composeront la solution optimale reviendra à grouper les lignes avec la même valeur pour le ou les attributs Y de la main droite de la dépendances. Ensuite plutôt que de faire l'union des sous-ensembles comme précédement, il conviendra de choisir l'un des sous-ensemble. Pour faire ce choix, il faut calculer la distance que la suppression de chacun des groupes impliquerait et prendre le groupe de ligne avec la distance la plus faible. Ensuite, la main droite de la dépendance étant identique pour toutes les lignes du sous-ensemble, cette dépendance peut être supprimée.

Dans l'exemple de la figure 3.1, après avoir traité l'attribut commun, nous avons la dernière ligne qui respecte les contraintes et les trois autes pour qui il y a encore du travail. Traiter le consensus reviendrait à garder la première ligne ou les lignes 2 et 3. Si on garde la première ligne (et donc qu'on supprime les deux autres), la distance entre la table T initiale et la table image T' est $dist_{sub}(T, T') = 1 + 1 = 2$ or si ongarde les deux autres, la distance est de $dist_{sub}(T, T') = 3$. Garder la première ligne est donc la meilleure solution.

Maintenant, $\Delta = \{Room \rightarrow Floor\}$. Et notre sous-ensemble respecte cette dépendance donc le tour est joué.

2.2.3 Une chaîne :

est une dépendance fonctionnelle qui en inclus une autre. ....

2.2.4 Un mariage :

est une situation où la table présente plusieurs identifiants. Chacun d'eux va alors déterminer le ou les autres identifiants. Par exemple $\Delta = \{X \rightarrow Y\}, \{Y \rightarrow X\}$. On dit que leur ""closure"" sont identiques. et il faut également que l'un deux soit présent dans chacune des autres dépendances, par exemple $\Delta = \{X \rightarrow Y\}, \{Y \rightarrow X\}, \{X \rightarrow Z\}$. + complément video 0 :8 :00

2.3 Notations

Pour pouvoir transcrire le problème en notation mathématique, voici les raccourcis utilisés :

D : la base de données source incohérente

D' : image de D qui soit cohérente

T : table unique de la base de données D. Chaque ligne est numérotée par un identifiant qui ne sera jamais modifié. Il permet d'avoir un lien entre la version D et D' de la base. Elles sont aussi caractérisée par un poid qui détermine combien coûte la suppression de cette ligne. La table peut-être avec ou sans doublons et pondérée ou non. Si elle n'est pas pondérée, on fixera le même poids à toutes les lignes.

T' : image de T qui soit cohérente

R ou $R(a_1, ..., a_k)$: schéma relationnel de la table T où chaque a_i est un attribut. Cette relation est un sous-ensemble du produit cartésien de plusieurs domaines de valeurs.

Val : domaine (infini ?) de valeur d'attribut

$T[i]$: la ligne i de la table T

$T[*]$: toutes les lignes de T

$ids(T)$: identifiant de la ligne i de la table T ? ? ?

Val^k : domaine de valeur de l'attribut k. Il s'agit de l'ensemble des valeurs que l'attribut est succeptible de prendre.

$w_T(i)$: poids de la ligne i de la table T

$|T| = |ids(T)|$: le nombre de ligne de T

$t = (a_1, ..., a_k$: est une ligne de la table T

$t.A_j$: référence vers la valeur de a_j

$A, B, C, ..., X, Y, Z$: Les premières lettres de l'alphabet illustrent un seulattribut à la fois alors que les dernières lettres de l'alphabet représententun ensemble d'attribut.

Δ : ensemble des dépendances fonctionelles

$X \to Y$: expression d'une dépendance fonctionnelle où X et Y sont des séquences d'attribut

lhsourhs : »left/right hand side « , on évoque la partie gauche/droite de la dépendance fonctionnelle.

$\Delta| = X \to Y$: la dépendance fonctionnelle est ?implique ? dans Δ. C'est-à-dire que si une table vérifie Δ, elle vérifie aussi $X \to Y$.

$Cl(\Delta)$: la fermeture de delta. Elle regroupe la liste exhaustive detoutes les dépendances fonctionnelle qui sont ?impliquée ? par delta.

$Cl_{\Delta}(X)$: la fermeture d'un attribut (ou d'une séquence d'attribut) X. C'est la liste de tous les attributs A tel que $X \rightarrow A$ est impliquée dans Δ.

$\Delta 1 = \Delta 2$: si leurs fermetures sont identique.

:

:

:

:

:

:

:

:

:

...

Chapitre 3

Sous-ensemble optimal

Pour obtenir une version de la base de données cohérente avec un minimum de suppression de ligne, l'article [2] expose une solution. Les auteurs proposent un algorithme qui, sur base des dépendances fonctionnelles, prédit un problème polynomial ou APX-complet (sous-ensemble de la classe NP-difficile). Si le probème est de complexité polynomiale, alors une solution en un temps polynomial peut être trouver et ce, même si la table est lourde et/ou comporte des doublons. Dans le cas contraire, le problème ne peut qu'être approximé à une constante prêt.

La méthode de correction évaluée tiendra compte du caractère optimal de la solution. Pour se faire elle calculera la distance entre la la base de donnée source et son image. Cette distance sera calculée en additionnant le poid des lignes supprimées. Dans l'exemple de la figure 3.1, supprimer la première ligne, sera plus coûteux que de supprimer les deux suivantes. Pour respecter les dépendances fonctionnelles suivante : $\{Facility \rightarrow City\} et \{Facility, Room \rightarrow Floor\}$, on supprimera donc les lignes 2 et 3 pour obtenir le sous-ensemble optimal.

	id	Facility	Room	Floor	City	Weight
1	1	HQ	322	3	Paris	3
2	2	HQ	322	30	Madrid	1
3	3	HQ	122	1	Madrid	1
4	4	Lab1	835	3	London	4

FIGURE 3.1 – Extrait d'une table pour illustrer des dépendances fonctionnelles particulières

3.1 OSRSucceeds(δ)

Cet algorithme a été écrit pour prédire la complexité de la correction. Il va en effet travailler uniquement sur les dépendances fonctionnelles en ne tenant pas compte des données. Si l'algorithme est un succès, cela signifie qu'il aura réussi à simplifié les dépendances et à les substituée pour ne plus les traiter que une par une. Il est alors possible de lancer l'algorithme de correction des données qui mettra un temps polynomial à s'exécuter. Par contre si l'algorithme est un échec, alors il ne sera pas possible de corriger les données avec l'algorithme proposé dans un temps polynomial.

Voici le pseudo code de ce premier algorithme :

Algorithm 1 OSRSucceeds

1: $while \Delta is non trivial do$:
2: $remove trivial FDs from \Delta$
3: **if then** $\Delta has a common lhs Athen$
4: $\Delta := \Delta - A$ else if $\Delta has a consensus FD \emptyset \rightarrow Athen$
5: $\Delta := \Delta - A$ else if $\Delta has a nlhs marriage(X1, X2) then$
6: $\Delta := \Delta - X1X2$ else
7: **return** $false$
8: **return** $true$

Algorithm 2 OSRSucceeds

```
1: procedure SIMPLIFY(Δ)
2:     stringlen ← length of string
3:     i ← patlen
4: top :
5:     if i > stringlen then return false
6:     j ← patlen
7: loop :
8:     if string(i) = path(j) then
9:         j ← j − 1.
10:        i ← i − 1.
11:        goto loop.
12:        close ;
13:    i ← i + max(delta_1(string(i)), delta_2(j)).
14:    goto top.
```

3.2 OptSRepair(Δ, T)

... $1 : if \Delta istrivial then \% successfultermination 2 : return T 3 : removetrivialFDsfrom \Delta 4 : if \Delta hasacommonlhsthen 5 : returnCommonLHSRep(\Delta, T) 6 : if \Delta hasaconsensusFDthen 7 : returnConsensusRep(\Delta, T) 8 : if \Delta hasanlhsmarriagethen 9 : returnMarriageRep(\Delta, T) 10 : fail \% cannotfindanoptimalS - repair$

3.3 CommonLHSRep(Δ, T)

... $1 : A := acommonlhsof \Delta 2 : return \cup_{(a) \in \pi_A T[*]} OptSRepair(\sigma_{A=a} T, \Delta - A)$

3.4 ConsensusRep(Δ, T)

... $1 : selectaconsensusFD \emptyset \rightarrow Ain \Delta 2 : foralla \in \pi_A T[*] do 3 : S_a := OptSRepair(\sigma_{A=a} T, \Delta - A) 4 : a_{max} := argmax_a \{w_T(S_a) | (a) \in \sigma_A T[*]\} 5 : returnS_{a_{max}}$

3.5 MarriageRep(Δ, T)

... $1 : selectanlhsmarriage(X_1, X_2) of \Delta 2 : forall (a_1, a_2) \in \pi X_1 X_2 T[*] do 3 : S_{a_1,a_2} := OptSRepair(\sigma_{X_1=a_1, X_2=a_2} T, \Delta - X_1 X_2) 4 : w(a_1, a_2) := w_T(S_{a_1,a_2}) 5 : V_i := \pi_{X_i} T[*] fori = 1, 2 6 : E := \{(a_1, a_2) | (a_1, a_2) \in \pi_{X_1 X_2} T[*]\} 7 : G := weightedbipartitegraph(V_1, V_2, E, w) 8 : E_{max} := amaximummatchingofG 9 : return \cup_{(a_1,a_2) \in E_{max}} S_{a_1,a_2}$

Bibliographie

[1] https://www.youtube.com/watch?v=dXoLmFDbq7E, 30/11/2018.

[2] Ester Livshits, Benny Kimelfeld, Sudeepa Roy, *"Computing Optimal Repairs for Functional Dependencies"*, PODS. ACM, 2018, pp. 225–237.

[3] Josef Wijsen, Syllabus

[4] https://www.i3s.unice.fr/~nlt/cours/licence/sgbd1/sgbd1_cours.pdf, 15/12/2018.