UMONS
Université de Mons
Faculté des Sciences
Département d'Informatique

Mémoire :
Base de donnée

Directeurs : M^r Jef Wijsen

Projet réalisé par
Sophie Opsommer

Rapporteur :

Année académique 2018-2019

UMONS
Université de Mons

Faculté des Sciences

Mémoire :

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

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.

Une dépendance fonctioenlle 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 $X -> Y$ 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.

	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

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 professeurne ne peut se trouver que dans une classe à la fois. En terme de dépendance, 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. De la même façon, des élèves ne peuvent se trouver que dans une seule classe à la fois. Inversément, une classe n'est utilisée que par un professeur à la fois. Ces dépendances s'écrivent : {Eleves, Jour, Heure -> Classe}, {Professeur, Jour, Heure -> Classe}, {Classe, Jour, Heure -> Professeur}.

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

- Une erreur manuelle d'encodage ou de manipulation volontaire,
- 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 prevenant 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 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

$X -> Y$: 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.

$\Delta| = X -> Y$: la dépendance fonctionnelle est ?implique ? dans Δ. C'est-à-dire que si une table vérifie Δ, elle vérifie aussi X -> 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 -> 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 -> City } et { Facility, Room -> 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.

Avant de présenterle pseudo code, voici quelques notion supplémentaires.

3.1.1 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 : $\{\emptyset -> City\} et \{Room -> Floor\}$.

3.1.2 Un consensus :

est une dépendance fonctionnelle dont la main gauche est vide : $\emptyset -> Y$. 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, $\Delta = \{Room- > Floor\}$. Et notre sous-ensemble respecte cette dépendance donc le tour est joué.

3.1.3 Une chaîne :

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

3.1.4 Un mariage :

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

Voici le pseudo code de ce premier algorithme :

Algorithm 1 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.
```

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] test