Visualisation par l'exemple des dépendances dans les bases de données relationnelles

Fabien De Marchi*, Jean-Marc Petit**

*Laboratoire LIRIS, UMR CNRS 5205
Université Claude Bernard - Lyon 1
8, boulevard Niels Bohr, 69 622 Villeurbanne cedex France
fabien.demarchi@liris.cnrs.fr
**Laboratoire LIMOS, UMR CNRS 6158
Université Blaise Pascal - Clermont-Ferrand II
24 avenue des Landais, 63 177 Aubière cedex, France
jmpetit@math.univ-bpclermont.fr

Résumé. Comprendre la sémantique des bases de données relationnelles existantes est important pour de nombreuses applications. Cette sémantique est principalement véhiculée par les dépendances fonctionnelles (DF) et les dépendances d'inclusion (DI) ; elles généralisent respectivement les notions de clé et de clé étrangère. Toutefois, il est fréquent que les bases de données opérationnelles deviennent désordonnées dans le temps ; dans ce cas, les contraintes d'intégrité doivent être retrouvées à partir des données. Plusieurs méthodes ont été proposées pour la découverte des DF ou des DI. Ces algorithmes fournissent à l'administrateur un ensemble de dépendances satisfaites dans les données.

Se pose alors le problème de la compréhension des dépendances extraites, incluant des aspects liés à la visualisation des connaissances. Cette étape doit permettre, par exemple, d'assister l'utilisateur final à sélectionner les règles intéressantes, ou à comprendre pourquoi une dépendance attendue n'est pas satisfaite dans les données. Nous proposons de fournir à l'administrateur ou l'analyste, en complément de la liste des règles, un échantillon de la base de données, vérifiant exactement les même DF et DI, appelé *base de données d'Armstrong informative* (BDAI). Ces exemples nous semblent particulièrement adaptés pour faciliter les échanges entre l'administrateur et les experts du domaine. Nous donnons certaines propriétés sur l'existence et la taille des BDAI, ainsi que des algorithmes pour les construire. Des expérimentations sur une base réelle issue du web montrent l'intérêt pratique de cette proposition.

1 Introduction

Comprendre la sémantique des bases de données relationnelles existantes est important pour de nombreuses applications. Parmi elles, citons des travaux de rétro-conceptions [Casanova et de Sa, 1983, Markowitz et Makowsky, 1990, Petit *et al.*, 1996, Comyn-Wattiau et Akoka, 1999], dont le but est de retrouver le schéma conceptuel des données à partir de leur forme relationnelle, des travaux sur l'intégration de données

issues de sources différentes [Miller *et al.*, 2001], ou sur la maintenance des vues dans les entrepôts de données [Quass *et al.*, 1996, Laurent *et al.*, 1999].

Cette sémantique est principalement véhiculée par les contraintes d'intégrité [Abiteboul *et al.*, 1995, Levene et Loizou, 1999a], dont les plus importantes sont les dépendances fonctionnelles (DF) et les dépendances d'inclusion (DI) ; elles généralisent respectivement les notions de clé d'une relation et de clé étrangère entre deux relations. Dans le meilleur des cas, ces contraintes ont été spécifiées durant la phase de conception de la base, et sont ainsi à la disposition de l'administrateur. Toutefois, il est fréquent que les bases de données opérationnelles deviennent désordonnées dans le temps, à la suite par exemple de saisies erronées, ou d'utilisations incorrectes de la base. Dans ce cas, les contraintes d'intégrité doivent être retrouvées à partir des données.

Récemment, plusieurs méthodes ont été proposées pour la découverte des DF ([Huhtala *et al.*, 1999, Novelli et Cicchetti, 2001, Lopes *et al.*, 2002] ou des DI ([De Marchi *et al.*, 2002a, De Marchi et Petit, 2003]). Ces algorithmes fournissent à l'administrateur un ensemble de dépendances satisfaites dans les données. Pour assurer le passage à l'échelle de ces méthodes, une attention particulière a été portée sur l'élaboration d'algorithmes efficaces, de structure de données adaptées et d'accès aux données optimisés. En revanche, la question de l'interactivité et de la présentation des résultats à l'utilisateur n'a été pratiquement pas abordée. A notre connaissance, l'ensemble des algorithmes fournissent en sortie une liste d'expressions de dépendances satisfaites dans les données.

De nombreux résultats permettent de manipuler de telles listes, de les transformer de façon qu'elles soient, par exemple, minimales dans le nombre de dépendances ou d'attributs, ou encore qu'elles ne soient pas redondantes. Il n'en reste pas moins qu'il s'agit d'expressions spécialisées, uniquement lisibles par l'administrateur des données. Et même ce dernier aura bien du mal à comprendre, d'un simple coup d'œil, *pourquoi* telle ou telle dépendance est satisfaite ou non dans ses données ; ou encore s'il s'agit d'une propriété "locale" à sa base ou si une dépendance est réellement une propriété sémantique des données.

Dans cet article, nous nous intéressons au problème de la visualisation des DF et des DI extraites au cours d'un processus d'extraction de connaissances dans les données (ECD). L'objectif est de rendre plus compréhensible les dépendances pour un utilisateur familier du domaine (expert) mais non familier du modèle relationnel. Si nos buts sont clairement proches de la finalité habituelle des travaux sur la visualisation [Fayyad *et al.*, 2002], notre démarche diffère légèrement ; plutôt qu'une visualisation basée sur le mode graphique, nous proposons une *visualisation par l'exemple* des dépendances satisfaites.

Nous définissons la notion de *bases de données d'Armstrong informatives* (BDAI). Il s'agit de petits échantillons des bases de données réelles, *mais possédant exactement les mêmes DF et DI satisfaites*. Ces échantillons sont donc une représentation alternative pour représenter les dépendances, parfaitement équivalentes à des listes de DF et de DI. Le résultat central de cet article est la définition de bornes pour la taille de ces échantillons ; elle assure, dans la plupart des cas une taille de la base exemple bien inférieure à la taille de départ. Nous donnons également un algorithme détaillé pour construire des BDAI de taille bornée ; cet algorithme est implémenté et testé sur une

base de données réelle, démontrant un excellent potentiel de réduction (division par 150 de la base originale) en pratique.

2 Préliminaires

Quelques concepts nécessaires à la compréhension de l'article sont rappelés ici; pour plus de détails, le lecteur pourra se référer par exemple à [Abiteboul *et al.*, 1995, Levene et Loizou, 1999a]. Certaines notations pouvant différer selon les auteurs, nous utiliserons ici celles proposées par [Levene et Loizou, 1999a]. Les notions d'attributs, tuples, relations et bases de données relationnelles sont supposées familières au lecteur.

Dépendances fonctionnelles Une *dépendance fonctionnelle* (DF) est la généralisation de la notion de clé primaire. Soit R un schéma de relation, il s'agit d'une expression de la forme $X \rightarrow A$ où $X \subseteq R$ et $A \in R$. Une dépendance fonctionnelle $X \rightarrow A$ est *satisfaite* dans une relation r (noté $r \models X \rightarrow A$) si et seulement si $\forall t_i, t_j \in r, t_i[X] = t_j[X] \Rightarrow t_i[A] = t_j[A]$. On appellera F_r l'ensemble de toutes les DF satisfaites dans une relation r, et F_d l'ensemble des DF satisfaites dans une base de données $\mathbf{d}$.

Soit F un ensemble de DF sur un schéma R. Une DF f sur R est dite *logiquement impliquée* par F, noté $F \models f$, si et seulement si $\forall r$ sur R, si $r \models F$ alors $r \models X \rightarrow A$. La fermeture d'un ensemble d'attributs X relativement à F est l'ensemble : $X_F^+ = \{A \in R \mid F \models X \rightarrow A\}$. Tout ensemble de DF sur R induit donc une fermeture sur $P(R)$, l'ensemble des parties de R. On note $CL(F)$ l'ensemble des fermés induits par F. On appelle générateurs de F, noté $GEN(F)$, la famille unique de générateurs par intersection de $CL(F)$ [Armstrong et Delobel, 1980].

Soit t_i et t_j deux tuples d'une relation r sur un schéma R et $X \subseteq R$ un ensemble d'attributs. On note $ag(t_i, t_j)$ l'ensemble défini par $ag(t_i, t_j) = \{A \in R \mid t_i[A] = t_j[A]\}$. Les *ensembles en accord de* r, notés $ag(r)$ sont définis par $ag(r) = \{ag(t_i, t_j) \mid t_i, t_j \in r, t_i \neq t_j\}$.

Dépendances d'inclusion Les dépendances d'inclusion généralisent la notion de clé étrangère. Une *dépendance d'inclusion* (DI) sur un schéma de base de données $\mathbf{R} = \{R_1, ..., R_n\}$ est une expression de la forme $R_i[X] \subseteq R_j[Y]$, où $R_i, R_j \in \mathbf{R}$, X et Y sont des séquences d'attributs respectivement de R_i et R_j. De plus, X et Y sont des séquences de même taille et de même type. $R_i[X] \subseteq R_j[Y]$ *est satisfaite* dans une base de données $\mathbf{d} = \{r_1, ..., r_n\}$ sur $\mathbf{R}$ (noté $\mathbf{d} \models R_i[X] \subseteq R_j[Y]$) si et seulement si $\forall t \in r_i, \exists s \in r_j$ tel que $t[X] = s[Y]$ (ou de manière équivalente $\pi_X(r_i) \subseteq \pi_Y(r_j)$). On note $I_{\mathbf{d}}$ l'ensemble des DI satisfaites dans une base de données $\mathbf{d}$.

Un ensemble I de DI est dit *circulaire* s'il contient m ($m \geq 1$) DI de la forme : $R_1[X_1] \subseteq R_2[Y_2]$,$R_2[X_2] \subseteq R_3[Y_3]$,...,$R_m[X_m] \subseteq R_1[Y_1]$. Si $X_1 = Y_1$, alors on dit que I est *circulaire propre*.

Soit I un ensemble de DI sur $\mathbf{R}$. Une DI $R_i[X] \subseteq R_j[Y]$ est logiquement impliquée par I, noté $I \models R_i[X] \subseteq R_j[Y]$, si pour toute base de données $\mathbf{d}$ sur $\mathbf{R}$, si $\mathbf{d} \models I$ alors $\mathbf{d} \models R_i[X] \subseteq R_j[Y]$. Soit I et J deux ensembles de DI. I est *une couverture* de J ssi $I^+ = J^+$, avec $I^+ = \{R_i[X] \subseteq R_j[Y]$ tel que $I \models R_i[X] \subseteq R_j[Y]\}$.

Une relation d'ordre partiel (ou de spécialisation), peut être définie sur l'ensemble des DI définies sur un schéma de relation (voir [Mannila et Toivonen, 1997], [De Marchi et Petit, 2003] pour plus de détails sur cette relation d'ordre). La propriété de satisfaction des DI est alors anti-monotone [Mannila et Toivonen, 1997] relativement à cet ordre. Ainsi, tout ensemble I de DI peut-être représenté par sa bordure positive notée $\mathcal{B}d^+(I)$, ses éléments les plus spécialisés. De façon duale, les DI n'appartenant pas à I peuvent être représentées par leur bordure négative $\mathcal{B}d^-(I)$, l'ensemble des éléments les plus généraux qui ne sont pas dans I.

Bases de données d'Armstrong pour les DF et les DI Les *relations d'Armstrong* ont tout d'abord été définies pour les DF [Armstrong, 1974], puis étendues aux *bases de données d'Armstrong* pour les DF et DI [Fagin et Vardi, 1983].

Soient un schéma de base de données $\mathbf{R} = \{R_1, ..., R_n\}$, F un ensemble de DF et I un ensemble de DI définis sur $\mathbf{R}$, et une base de données $\mathbf{d} = \{r_1, ..., r_n\}$ sur $\mathbf{R}$. $\mathbf{d}$ est une *base de données d'Armstrong* pour $F \cup I$ si, pour toute DF ou DI α :

$$\mathbf{d} \models \alpha \Longleftrightarrow F \cup I \models \alpha$$

Dans le cas des DF seules, une caractérisation des relations d'Armstrong est donnée dans [Beeri *et al.*, 1984] : si r et F sont respectivement une relation et un ensemble de DF définis sur un schéma de relation R, alors r est une relation d'Armstrong pour F si et seulement si $GEN(F) \subseteq ag(r) \subseteq CL(F)$.

Il est montré dans [Fagin et Vardi, 1983] qu'une base de données d'Armstrong existe pour tout ensemble de DF et de DI[1]. Des algorithmes ont été proposés pour les construire dans le cas où l'ensemble des DI est circulaire propre [Levene et Loizou, 1999b].

3 Bases de données d'Armstrong pour des bases de données

3.1 Définition

La définition d'une base de données d'Armstrong pour une base de données étend naturellement celle d'une base de données d'Armstrong pour un ensemble de DF et de DI. Les DF et les DI sont implicitement celles satisfaites dans la base de données de départ.

Définition 1 (Base de données d'Armstrong pour une base de données) Soit $\mathbf{d}$ une base de données sur un schéma $\mathbf{R}$. Une *base de données d'Armstrong* $\overline{\mathbf{d}}$ sur $\mathbf{R}$ est telle que $\overline{\mathbf{d}}$ est une base de données d'Armstrong pour $F_{\mathbf{d}} \cup I_{\mathbf{d}}$.

Il s'agit donc d'une représentation alternative des dépendances satisfaites dans une base de données. En soit, cette représentation permet de visualiser autrement les dé-

[1] En fait, il faut restreindre les DF à la classe des DF standards, c.a.d. avec une partie gauche non vide [Fagin et Vardi, 1983].

pendances d'une base de données ; nous allons voir dans la suite quelles sont les inconvénients d'une simple application de cette définition pour la compréhension des bases existantes.

3.2 Limite des méthodes existantes

L'une des principales application des bases de données d'Armstrong a été l'aide à la conception de bases de données [Silva et Melkanoff, 1979, Mannila et Räihä, 1986]. Le principe est, à partir d'un schéma et d'un ensemble de contraintes spécifiées par le concepteur, de générer des bases de données d'Armstrong, aussi appelées *bases de données exemples* ; le concepteur peut alors visualiser ses dépendances de façon moins abstraite grâce à l'exemple. S'il décide d'apporter des modification sur les exemples obtenus, l'ensemble des DF est alors recalculé. Ce processus est itéré jusqu'à convergence, c'est à dire à la fois les DF et l'exemple conviennent au concepteur.

Notre contexte est légèrement différent, puisque nous nous intéressons à la maintenance des bases existantes et non à leur conception. Une extension directe des travaux existants consisterait à : 1) découvrir les DF et DI satisfaites dans la base de départ et 2) construire des bases de données d'Armstrong à partir de ces DF et DI, pour fournir ainsi une petite base exemple à l'administrateur. Soulignons que la taille de la base obtenue serait en général très petite, d'après les bornes exhibées (dans le cas des DF seules) dans [Beeri *et al.*, 1984].

L'exemple suivant illustre cette procédure, et nous permet de souligner toutefois ses limites.

Exemple 1 La base de données **movies**, représentée dans la figure 1 nous servira d'exemple tout au long de l'article. Il s'agit d'un échantillon jouet d'une base réelle existante, présentée et exploitée dans la section 5 [Bay, 1999].
Pour l'instant, nous allons nous limiter à la seule relation *remakes* et à la compréhension des DF qui y sont satisfaites. Supposons que l'on connaisse la couverture canonique de ces DF : $F = \{A \longrightarrow B, B \longrightarrow AC, CE \longrightarrow B, DE \longrightarrow C, AE \longrightarrow D, AD \longrightarrow E\}$. Plusieurs méthodes existent pour construire une relation d'Armstrong pour F dont [Fagin et Vardi, 1983, Beeri *et al.*, 1984, Demetrovics et Thi, 1995] ; elles ont en commun qu'elles construisent des tuples à partir de valeurs soit entières, soit puisées dans les domaines de définition des attributs, soit enfin dans le domaine actif des attributs. La Figure 2 donne une relation d'Armstrong pour les DF de la relation *remakes*, telle qu'elle pourrait être construite par l'application des méthodes existantes à partir des domaines actifs.
On constate dans cet exemple que les tuples de la relation d'Armstrong construite sont complètement fictifs. Supposons qu'un administrateur ait pour tâche de comprendre la sémantique de la relation *remakes*. Si l'administrateur désire savoir pourquoi l'attribut $A(ID)$ n'est pas clé, la lecture des deux premiers tuples de la relation exemple ne l'aide pas beaucoup puisque ces deux tuples n'appartiennent pas à la relation initiale.

Une autre limitation d'une telle approche concerne l'existence même de la relation d'Armstrong; en effet, il faut qu'un nombre suffisant de valeurs distinctes soient dis-

remakes

ID (A)	*TITLE* (B)	*YEAR* (C)	*FRAC* (D)	*PRID* (E)	*PRTITLE* (F)
16	Le voyage	1959	0.9	11	Mademoiselle Fifi
29	Cléopâtre	1962	0.3	23	Cléopâtre
28	Des soeurs courageuses	1938	0.9	17	Quatre soeurs
15	Les amours de Carmen	1948	0.2	22	Gilda
21	Une heure avec toi	1932	0.9	18	Le cercle du mariage
16	Le voyage	1959	0.3	17	Quatres soeurs
40	L'Oeuvre au noir	1990	0.8	42	Le nom de la rose
12	Young at heart	1938	0.9	30	Casablanca
40	L'Oeuvre au noir	1990	0.1	14	Thérèse

movies

FILMID (G)	*TITLE* (H)	*YEAR* (I)
11	Mademoiselle Fifi	1944
12	Young at heart	1938
13	L'exorciste	1971
43	Thérèse	1980
15	Les amours de Carmen	1948
16	Le voyage	1959
17	Quatre soeurs	1938
18	Le cercle du mariage	1924
21	Une heure avec toi	1932
22	Gilda	1946
23	Cléopâtre	1934
28	Des soeurs courageuses	1938
29	Cléopâtre	1962
30	Casablanca	1944
40	L'Oeuvre au noir	1990
41	Le manitou	1978
42	Le nom de la rose	1985

FIG. 1 – Base de données **movies**.

ID (A)	*TITLE* (B)	*YEAR* (C)	*FRAC* (D)	*PRID* (E)	*PRTITLE* (F)
16	Le voyage	1959	0.9	11	Mademoiselle Fifi
16	Le voyage	1959	0.3	23	Cléopâtre
29	Cléopâtre	1959	0.9	17	Quatre soeurs
28	Des soeurs courageuses	1962	0.9	22	Gilda
15	Les amours de Carmen	1938	0.2	11	Le cercle du mariage

FIG. 2 – Une relation d'Armstrong pour *remakes*

ponibles dans les domaines actifs des attributs. Dans le cas contraire, les algorithmes existants ne peuvent s'exécuter.

Pour pallier ces problèmes, la section suivante introduit les *bases de données d'Armstrong informatives*

4 Les bases de données d'Armstrong informatives

4.1 Définition

Une base de données d'Armstrong informative (BDAI) est une base de données d'Armstrong pour une base de données, mais avec une contrainte supplémentaire : tous les tuples sont issus de la base de données de départ, plutôt que d'être construits arbitrairement avec les valeurs des domaines actifs. Dans la suite, nous étendons la définition classique de l'opérateur $\subseteq$ aux bases de données : Soit $\mathbf{d} = \{r_1, ..., r_n\}$ et $\overline{\mathbf{d}} = \{\overline{r}_1, ..., \overline{r}_n\}$ deux bases de données sur le même schéma. Nous supposons, sans perte de généralité, que les schémas de relation sont classés dans un certain ordre (ici, ils seront indicés). On dit que $\overline{\mathbf{d}}$ est incluse dans $\mathbf{d}$, noté $\overline{\mathbf{d}} \subseteq \mathbf{d}$, si pour chaque $\overline{r}_i \in \overline{\mathbf{d}}, \overline{r}_i \subseteq r_i$.

Définition 2 (Bases de données d'Armstrong informatives) Soit $\mathbf{d} = \{r_1, ..., r_n\}$ une base de données sur un schéma $\mathbf{R} = \{R_1, ..., R_n\}$, une base de données d'Armstrong informative (BDAI) $\overline{\mathbf{d}}$ pour $\mathbf{d}$ est définie sur R et est telle que :
– $\overline{\mathbf{d}}$ est une base de données d'Armstrong pour $\mathbf{d}$ et
– $\overline{\mathbf{d}} \subseteq \mathbf{d}$.

L'idée est de mettre en évidence un sous-ensemble, pour chaque relation dans $\mathbf{d}$, satisfaisant exactement les mêmes DF et DI que la base de départ.

4.2 Vers de petites BDAI

En premier lieux, remarquons qu'une BDAI existe pour toute base de données en entrée. En effet, toute base de données est une BDAI pour elle-même. L'intérêt est alors de construire des BDAI dont le nombre de tuples est inférieur à celui de la base de départ. La propriété suivante assure l'existence d'une BDAI avec un nombre de tuples borné, pour toute base de données $\mathbf{d}$ en entrée dans laquelle les DI satisfaites ne sont pas circulaires. On note $\mathcal{B}d^-(I_d)_{R_i \rightarrow R_j}$ la restriction de l'ensemble $\mathcal{B}d^-(I_d)$ à ses éléments définis du schéma R_i vers le schéma R_j. La même notation est utilisée pour $\mathcal{B}d^+(I_d)$. La cardinalité d'une relation r ou d'une base de données $\mathbf{d}$, notée $|r|$ ou $|\mathbf{d}|$, est le nombre de tuples qu'elle contient.

Théorème 1 *Soit* $\mathbf{d} = \{r_1, ..., r_n\}$ *une base de données sur un schéma* $\boldsymbol{R} = \{R_1, ..., R_n\}$. *Il existe une BDAI* $\overline{\mathbf{d}} = \{\overline{r_1}, ..., \overline{r_n}\}$ *pour* $\mathbf{d}$ *telle que :*
$|\overline{\mathbf{d}}| = \sum_{i=1}^{n} |\overline{r}_i|$ *avec,* $\forall i = 1, \ldots, n$ *:*

$$\lceil \frac{1 + \sqrt{1 + 8 \times |GEN(F_{r_i})|}}{2} \rceil \leq |\overline{r}_i|$$

et

$$|\overline{r}_i| \leq Min(|r_i|, 2 \times |GEN(F_{r_i})| + \sum_{j=1}^{n} |\mathcal{B}d^-(I_d)_{R_i \to R_j}| + \sum_{j=1}^{n} |\mathcal{B}d^+(I_d)_{R_j \to R_i}| \times |\overline{r}_j|)$$

On constate que la borne supérieure est définie de façon récursive, et donc que les relations doivent être considérées dans un ordre particulier pour son évaluation. Cet ordre existe et est facilement déterminable à la condition que l'ensemble $\mathcal{B}d^+(I_d)$ soit non circulaire.

La suite de cette section est consacrée à la démonstration du principal résultat de cet article donné dans le théorème 1. Pour ce faire, nous exhibons une preuve constructive dont les principales étapes sont données par l'algorithme 1.

Algorithme 1 Base de données d'Armstrong informative

Entrée: $\mathbf{d}$, $GEN(F_d)$, $\mathcal{B}d^+(I_d)$ et $\mathcal{B}d^-(I_d)$;
Sortie: une BDAI $\overline{\mathbf{d}}$ pour $\mathbf{d}$;
1: $\dot{\mathbf{d}} = \emptyset$;
2: **for all** $r \in \mathbf{d}$ **do**
3: Calculer $GEN(F_r)$;
4: $\dot{\mathbf{d}} = \dot{\mathbf{d}} \cup \{GenRAI(r, GEN(F_r))\}$ – *prise en compte des DF satisfaites et non satisfaites*
5: **end for**
6: $\hat{\mathbf{d}} = InvalidIND(\dot{\mathbf{d}}, \mathcal{B}d^-(I_d), \mathbf{d})$; – *prise en compte des DI non satisfaites*
7: $\overline{\mathbf{d}} = ValidIND(\hat{\mathbf{d}}, \mathcal{B}d^+(I_d), \mathbf{d})$; – *prise en compte des DI satisfaites*
8: **Retourner** $\overline{\mathbf{d}}$.

Celui-ci prend en entrée une base de données $\mathbf{d}$, les générateurs des DF satisfaites dans $\mathbf{d}$, ainsi que les bordures positives et négatives des DI satisfaites dans $\mathbf{d}$. Il construit alors une BDAI pour $\mathbf{d}$ en trois étapes ; chacune des étapes consiste à rajouter des tuples de $\mathbf{d}$ dans la base en construction de façon à obtenir finalement la base $\overline{\mathbf{d}}$ telle que $F_{\overline{\mathbf{d}}} = F_{\mathbf{d}}$ et $I_{\overline{\mathbf{d}}} = I_{\mathbf{d}}$.

Chaque étape de l'algorithme est détaillée dans la suite ; un certain nombre de propositions assurent l'exactitude de l'algorithme, tandis que deux lemmes correspondant à des cas particuliers du théorème 1 sont donnés. Finalement, la preuve du théorème 1 est exhibée à la fin de cette section.

4.3 Prise en compte des DF

L'objectif de cette première étape est de construire une base de données $\dot{\mathbf{d}}$ telle que :

- $\dot{\mathbf{d}} \subseteq \mathbf{d}$ et
- $F_{\dot{\mathbf{d}}} = F_{\mathbf{d}}$

Cet objectif peut-être atteint en construisant, pour chaque relation r_i de la base de données $\mathbf{d}$ de départ, une *relation d'Armstrong informative* (RAI) $\dot{r}_i$ telle que ([De Marchi *et al.*, 2002b]) :

- $\dot{r}_i \subseteq r_i$ et
- $F_{\dot{r}_i} = F_{r_i}$

La base $\dot{\mathbf{d}} = \{\dot{r_1}, ..., \dot{r_n}\}$ possédera alors les propriétés requises.

La proposition 1 donne une caractérisation des IAR ([De Marchi *et al.*, 2002b]) :

Proposition 1 *Soit r et $\dot{r}$ deux relations sur un schéma $\boldsymbol{R}$, telles que $\dot{r} \subseteq r$. $\dot{r}$ est une relation d'Armstrong informative (RAI) pour r si et seulement si : $GEN(F_r) \subseteq ag(\dot{r})$.*

Preuve

($\Leftarrow$) *Puisque $GEN(F_r) \subseteq ag(\dot{r})$, il reste à s'assurer que $ag(\dot{r}) \subseteq CL(F_r)$ pour que $\dot{r}$ soit une relation d'Armstrong pour F_r (cf. préliminaires). Or, $\dot{r} \subseteq r$ donc $ag(\dot{r}) \subseteq ag(r)$. Puisque r est une relation d'Armstrong pour F_r, $ag(r) \subseteq CL(F_r)$ et donc $ag(\dot{r}) \subseteq CL(F_r)$.*

De plus, on a $\dot{r} \subseteq r$, donc $\dot{r}$ est une RAI pour r.

($\Rightarrow$) *$\dot{r}$ est une relation d'Armstrong pour F_r, donc $GEN(F_r) \subseteq ag(\dot{r}) \subseteq CL(F_r)$.* □

L'algorithme 2 est déduit de cette caractérisation. Il choisi au hasard dans r deux tuples en accord sur chaque générateur de F_r, pour les insérer dans la RAI en construction. Sa complexité est linéaire dans le nombre de générateurs de F_r.

Algorithme 2 GenRAI

Entrée: une relation r, $GEN(F_r)$;
Sortie: une RAI $\dot{r}$ pour r ;

1: $\dot{r} = \emptyset$
2: **for all** $X \in GEN(F_r)$ **do**
3: Soit $t_1, t_2 \in r$ tels que $ag(t_1, t_2) = X$;
4: $\dot{r} = \dot{r} \cup \{t_1\} \cup \{t_2\}$
5: **end for**
6: **Retourner** $\dot{r}$.

Exemple 2 La figure 3 représente la base de données **movies**, composée de relations d'Armstrong informatives pour chacune des relations de la base de données **movies** (figure 1 page 6). Ces relations sont construites à partir des ensembles de générateurs suivants : $\{ABC, CD, D, E\}$ pour *remakes* et $\{H, I\}$ pour *movies*.

$\dot{remakes}$

ID (A)	*TITLE* (B)	*YEAR* (C)	*FRAC* (D)	*PRID* (E)	*PRTITLE* (F)
16	Le voyage	1959	0.9	11	Mademoiselle Fifi
28	Des soeurs courageuses	1938	0.9	17	Quatre soeurs
16	Le voyage	1959	0.3	17	Quatres soeurs
12	Young at heart	1938	0.9	30	Casablanca

$\dot{movies}$

FILMID (G)	*TITLE* (H)	*YEAR* (I)
12	Young at heart	1938
23	Cléopâtre	1934
28	des soeurs courageuses	1938
29	Cléopâtre	1962

FIG. 3 – Base de données $\dot{\mathbf{movies}}$, telle que $F_{\dot{\mathbf{movies}}} = F_{\mathbf{movies}}$.

Cette base de données vérifie la propriété $F_{\dot{\mathbf{movies}}} = F_{\mathbf{movies}}$.

De l'algorithme 2, on peut déduire de façon naturelle des bornes pour chaque relation de la RAI construite, et donc des bornes pour la relation $\dot{\mathbf{d}}$ qui est l'union de ces relations.

Lemme 1 *Soit* $\boldsymbol{d}$ *une base de données. Il existe une base de données* $\dot{\boldsymbol{d}} \subseteq \boldsymbol{d}$ *telle que* $F_{\dot{\boldsymbol{d}}} = F_{\boldsymbol{d}}$ *et :* $|\dot{\boldsymbol{d}}| = \sum_{i=1}^{n} |\dot{r}_i|$ *avec,* $\forall i = 1, \ldots, n$ *:*

$$\lceil \frac{1 + \sqrt{1 + 8 \times |GEN(F_{r_i})|}}{2} \rceil \leq |\dot{r}_i| \leq Min(|r_i|, 2 \times |GEN(F_{r_i})|$$

Intuitivement, la borne inférieure de chaque RAI $\dot{r}_i$ correspond au nombre minimum de tuples qu'elle doit contenir pour avoir au moins un couple en accord sur chaque générateur de F_{r_i}. La borne supérieure correspond au cas où chacun de ces couples de tuples sont distincts.

La proposition suivante assure que l'ensemble des DF satisfaites par $\dot{\mathbf{d}}$ reste inchangé quelque soient les nouveaux tuples de $\mathbf{d}$ que l'on insère dans $\dot{\mathbf{d}}$:

Proposition 2 *Pour toute base de données* $\hat{\boldsymbol{d}}$ *telle que* $\dot{\boldsymbol{d}} \subseteq \hat{\boldsymbol{d}} \subseteq \boldsymbol{d}$*, on a* $F_{\hat{\boldsymbol{d}}} = F_{\boldsymbol{d}}$*.*

Preuve *Soit* $\dot{\boldsymbol{d}} = \{\dot{r}_1, ..., \dot{r}_n\} \subseteq \boldsymbol{d} = \{r_1, ..., r_n\}$ *telle que* $\forall i \in \{1, ..., n\}$*,* $\dot{r}_i$ *est une RAI pour* r_i*. D'après la propriété 1,* $GEN(F_{r_i}) \subseteq ag(\dot{r}_i)$*.*
Soit $\dot{r}_i \subseteq \hat{r}_i \subseteq r_i$*. On a forcément* $ag(\dot{r}_i) \subseteq ag(\hat{r}_i)$*, donc* $GEN(F_{r_i}) \subseteq ag(\hat{r}_i)$*.*
Puisque $\hat{r}_i \subseteq r_i$*,* $\hat{r}_i$ *est une RAI pour* r_i*, toujours d'après la propriété 1.* □

4.4 Prise en compte des DI non satisfaites

Cette phase a pour but de contredire les DI non satisfaites dans $\mathbf{d}$ afin de construire $\hat{\mathbf{d}}$ telle que $\dot{\mathbf{d}} \subseteq \hat{\mathbf{d}} \subseteq \mathbf{d}$ et $I_{\hat{\mathbf{d}}} \subseteq I_{\mathbf{d}}$. Pour contredire les DI, nous utiliserons la notion de

tuple disqualifiant définie par :

Définition 3 (tuple disqualifiant) Soit $R_i[X] \subseteq R_j[Y]$ une DI sur $\mathbf{R}$. Un tuple $t \in r_i$ est un *tuple disqualifiant* pour $R_i[X] \subseteq R_j[Y]$ si $t[X] \notin \pi_Y(r_j)$.

Ainsi, pour construire $\hat{\mathbf{d}}$, il faut insérer dans $\dot{\mathbf{d}}$ un seul tuple disqualifiant pour chaque DI non satisfaite. En fait, grâce à la propriété d'anti-monotonie des DI, il suffit de contredire la couverture négative $\mathcal{B}d^-(I)$ des DI satisfaites dans $\mathbf{d}$. Cette démarche conduit à l'algorithme 3.

Algorithme 3 *InvalidIND*

Entrée: $\mathbf{d}$, $\mathcal{B}d^-(I)$, $\dot{\mathbf{d}}$;
Sortie: $\hat{\mathbf{d}}$ telle que $\hat{\mathbf{d}} \subseteq \mathbf{d}$, $F_{\hat{\mathbf{d}}} = F_{\mathbf{d}}$, $I_{\hat{\mathbf{d}}} \subseteq I_{\mathbf{d}}$;
1: $\hat{\mathbf{d}} = \dot{\mathbf{d}}$;
2: **for all** $i = R_i[X] \subseteq R_j[Y] \in \mathcal{B}d^-(I)$ **do**
3: **if** $(\pi_X(\hat{r}_i) - \pi_Y(r_j)) = \emptyset$ **then**
4: Trouver un tuple disqualifiant t pour i ;
5: $\hat{r}_i = \hat{r}_i \cup \{t\}$;
6: **end if**
7: **end for**
8: **Retourner** $\hat{\mathbf{d}}$.

Pour chaque DI de $\mathcal{B}d^-(I)$, l'algorithme 3 cherche si un tuple disqualifiant appartient à la base de données en construction (ligne 3). Si ce n'est pas le cas, un tel tuple est choisi pour cette DI puis inséré dans $\hat{\mathbf{d}}$ (lignes 4 et 5). La complexité de cet algorithme est linéaire dans la taille de $\mathcal{B}d^-(I)$.

Exemple 3 Sur l'exemple courant, la base de données $\hat{\mathbf{movies}}$ est d'abord initialisée à $\dot{\mathbf{movies}}$ représentée dans la figure 3.
$\mathcal{B}d^-(I_{\mathbf{movies}})$ est composée de 12 DI, que nous ne détaillons pas (10 DI unaires et 2 DI de taille 2). Considérons plus particulièrement deux cas significatifs :
- pour la DI $A \subseteq E \in \mathcal{B}d^-(I_{\mathbf{movies}})$, l'ensemble $\pi_A(\hat{remakes}) - \pi_E(remakes)$ n'est pas vide, ce qui signifie qu'un tuple disqualifiant pour cette DI est déjà dans $\hat{remakes}$.
- pour la DI $H \subseteq B \in \mathcal{B}d^-(I_{\mathbf{movies}})$, $\pi_H(\hat{movies}) - \pi_B(remakes)$ est vide, et un tuple disqualifiant pour cette DI doit être choisi dans $movies$, par exemple $< 11, MademoiselleFifi, 1944 >$, et est inséré dans $\hat{movies}$.

Cette dernière DI est la seule restant à contredire dans la base en construction, et ainsi l'algorithme 3 pourrait construire la base de données $\hat{\mathbf{movies}}$ représentée dans la figure 4, telle que $\hat{\mathbf{movies}} \subseteq \mathbf{movies}$, $F_{\hat{\mathbf{movies}}} = F_{\mathbf{movies}}$ et $I_{\hat{\mathbf{movies}}} \subseteq I_{\mathbf{movies}}$.

$\hat{remakes}$

ID (A)	*TITLE* (B)	*YEAR* (C)	*FRAC* (D)	*PRID* (E)	*PRTITLE* (F)
16	Le voyage	1959	0.9	11	Mademoiselle Fifi
28	Des soeurs courageuses	1938	0.9	17	Quatre soeurs
16	Le voyage	1959	0.3	17	Quatres soeurs
12	Young at heart	1938	0.9	30	Casablanca

$\hat{movies}$

FILMID (G)	*TITLE* (H)	*YEAR* (I)
12	Young at heart	1938
23	Cléopâtre	1934
28	des soeurs courageuses	1938
29	Cléopâtre	1962
11	Mademoiselle Fifi	1944

FIG. 4 – Base de données $\mathbf{\hat{movies}}$, telle que $F_{\mathbf{\hat{movies}}} = F_{\mathbf{movies}}$ et $I_{\mathbf{\hat{movies}}} \subseteq I_{\mathbf{movies}}$.

On peut alors établir une borne pour la taille de $\hat{\mathbf{d}}$:

Lemme 2 *Soit* $\boldsymbol{d}$ *une base de données. Il existe une base de données* $\hat{\boldsymbol{d}} \subseteq \boldsymbol{d}$*, telle que* $F_{\hat{\boldsymbol{d}}} = F_{\boldsymbol{d}}$ *et* $I_{\hat{\boldsymbol{d}}} \subseteq I_{\boldsymbol{d}}$ *et :*
$|\hat{\boldsymbol{d}}| = \sum_{i=1}^{n} |\hat{r}_i|$ *avec,* $\forall i = 1, \ldots, n$ *:*

$$\lceil \frac{1 + \sqrt{1 + 8 \times |GEN(F_{r_i})|}}{2} \rceil \leq |\hat{r}_i|$$

et

$$|\hat{r}_i| \leq Min(|r_i|, 2 \times |GEN(F_{r_i})| + \sum_{j=1}^{n} |\mathcal{B}d^-(I)_{R_i \to R_j}|$$

Preuve *La borne inférieure provient du lemme 1, et correspond au cas où, dans la phase de prise en compte des DF, les tuples insérés étaient disqualifiant pour l'ensemble des DI de* $\mathcal{B}d^-(I)$*.*
La borne supérieure est celle donnée dans le lemme 1, augmentée du cas où un tuple disqualifiant pour chaque DI de $\mathcal{B}d^-(I)$ *a du être ajouté.* □

La proposition suivante assure que les propriétés de $\hat{\mathbf{d}}$ seront conservées dans la suite de l'algorithme, quelque soient les tuples de $\mathbf{d}$ que l'on insère dans $\hat{\mathbf{d}}$:

Proposition 3 *Pour toute base de données* $\overline{\boldsymbol{d}}$ *telle que* $\hat{\boldsymbol{d}} \subseteq \overline{\boldsymbol{d}} \subseteq \boldsymbol{d}$*, on a* $F_{\overline{\boldsymbol{d}}} = F_{\boldsymbol{d}}$ *et* $I_{\overline{\boldsymbol{d}}} \subseteq I_{\boldsymbol{d}}$*.*

Preuve $F_{\overline{\boldsymbol{d}}} = F_{\boldsymbol{d}}$ *provient de la proposition 2. Puisque* $\hat{\boldsymbol{d}} \subseteq \overline{\boldsymbol{d}}$*, il y a dans* $\overline{\boldsymbol{d}}$ *au moins un tuple disqualifiant pour chaque DI de* $\mathcal{B}d^-(I)$*, et ainsi* $I_{\overline{\boldsymbol{d}}} \subseteq I_{\boldsymbol{d}}$*.* □

4.5 Prise en compte des DI satisfaites

La ligne 3 de l'algorithme 1 a pour but de forcer les DI satisfaites dans $\mathbf{d}$ afin de construire une base de données $\overline{\mathbf{d}}$ telle que $\hat{\mathbf{d}} \subseteq \overline{\mathbf{d}} \subseteq \mathbf{d}$, $\overline{\mathbf{d}}$ étant une BDAI pour $\mathbf{d}$, i.e. $F_{\overline{\mathbf{d}}} = F_{\mathbf{d}}$ et $I_{\overline{\mathbf{d}}} = I_{\mathbf{d}}$.

Le principe est de considérer une par une chaque DI de $\mathcal{B}d^+(I)$, et de tester si elle est satisfaite ou non dans la base en construction. Si ce n'est pas le cas, elle doit être "forcée" dans la base en construction, à l'aide de la fonction $ForceDI$ (algorithme 4), qui insère les tuples nécessaires de $\mathbf{d}$ dans $\overline{\mathbf{d}}$.

Algorithme 4 $ForceDI$: force la satisfaction d'une DI

Entrée: $\mathbf{d}$, $\overline{\mathbf{d}} \subseteq \mathbf{d}$ et $i = R_i[X] \subseteq R_j[Y] \in \mathcal{B}d^+(I)$;
Sortie: $\overline{\mathbf{d}}$ telle que $\overline{\mathbf{d}} \subseteq \mathbf{d}$ et $\overline{\mathbf{d}} \models i$;
1: **for all** $v \in \{\pi_X(\overline{r}_i) - \pi_Y(\overline{r}_j)\}$ **do**
2: Soit $t \in r_j$ tel que $t[Y] = v$;
3: $\overline{r}_j = \overline{r}_j \cup \{t\}$;
4: **end for**
5: **Retourner** $\overline{\mathbf{d}}$

On doit maintenant se poser la question suivante : existe-t-il un ordre pour forcer les DI, de telle sorte qu'elles ne soient considérées qu'une fois chacune ? Dans la suite, nous considérons que l'ensemble $\mathcal{B}d^+(I)$ est non circulaire, avant d'aborder le cas de DI circulaires.

Supposons qu'une DI $i = R_i[X] \subseteq R_j[Y]$ ait été forcée dans $\overline{\mathbf{d}}$. Puisque l'on ne supprime jamais de tuple, la seule façon de contredire i est d'ajouter des tuples dans la partie gauche, soit dans $\overline{r}_i$. Cette action n'aura lieu que si une DI entrante dans R_i a été forcée. Ainsi, si un schéma de relation R_i n'a pas de DI entrante dans $\mathcal{B}d^+(I)$, chaque DI allant de R_i vers un autre schéma de relation peut être définitivement forcée.

L'algorithme 5 est basé sur ce principe. Si R est un schéma de relation, on note $\mathcal{B}d^+(I)_{\rightarrow R}$ l'ensemble des DI de $\mathcal{B}d^+(I)$ qui entrent dans R, et $\mathcal{B}d^+(I)_{R\rightarrow}$ l'ensemble des DI de $\mathcal{B}d^+(I)$ qui sortent de R.

Algorithme 5 $ValidIND$

Entrée: $\mathbf{d}$,$\hat{\mathbf{d}}$ et $\mathcal{B}d^+(I)$.
Sortie: $\overline{\mathbf{d}}$, BDAI pour $\mathbf{d}$;
1: $\overline{\mathbf{d}} = \hat{\mathbf{d}}$;
2: **for all** R tel que $\mathcal{B}d^+(I)_{\rightarrow R} = \emptyset$ **do**
3: **for all** $i \in \mathcal{B}d^+(I)_{R\rightarrow}$ **do**
4: $\overline{\mathbf{d}} = ForceDI(\mathbf{d}, \overline{\mathbf{d}}, i)$;
5: $\mathcal{B}d^+(I) = \mathcal{B}d^+(I) - \{i\}$;
6: **end for**
7: **end for**
8: **Retourner** $\overline{\mathbf{d}}$.

Les schémas de relation sans DI entrantes doivent être considérés en premier (ligne 2) : les DI qui en sortent sont forcées (ligne 4) et peuvent être retirées de la couverture des

DI (ligne 5). L'action se répète jusqu'à ce qu'aucune DI ne reste dans $\mathcal{B}d^+(I)$, ce qui est assuré puisque $\mathcal{B}d^+(I)$ est non circulaire. L'algorithme est globalement linéaire dans la taille de $\mathcal{B}d^+(I)$, au calcul près de $\mathcal{B}d^+(I)_{\rightarrow R}$ et $\mathcal{B}d^+(I)_{R\rightarrow}$ qu'on peut raisonnablement négliger.

Exemple 4 Dans notre exemple, la base de données $\overline{\textbf{movies}}$ est d'abord initialisée à $\hat{\textbf{movies}}$, représentée dans la figure 4. $\mathcal{B}d^+(I_{\textbf{movies}})$ est composée de 2 DI : $\mathcal{B}d^+(I_{\textbf{movies}}) = \{ABC \subseteq GHI, E \subseteq G\}$. La première relation considérée est donc la relation *remakes* puisqu'elle n'a pas de DI entrante. On force d'abord $ABC \subseteq GHI$ en insérant le tuple $< 16, \text{Le voyage}, 1959 >$ dans $\overline{movies}$, puis la DI $E \subseteq G$ en insérant les tuples $< 17, \text{Quatre soeurs}, 1938 >$ et $< 30, \text{casablanca}, 1944 >$ dans $\overline{movies}$.
Les deux DI sont alors supprimées, et l'algorithme se termine. On obtient finalement la base de données $\overline{\textbf{movies}}$, BDAI pour **d**, représentée dans la figure 5.

$\overline{remakes}$

ID (A)	*TITLE* (B)	*YEAR* (C)	*FRAC* (D)	*PRID* (E)	*PRTITLE* (F)
16	Le voyage	1959	0.9	11	Mademoiselle Fifi
28	Des soeurs courageuses	1938	0.9	17	Quatre soeurs
16	Le voyage	1959	0.3	17	Quatres soeurs
12	Young at heart	1938	0.9	30	Casablanca

$\overline{movies}$

FILMID (G)	*TITLE* (H)	*YEAR* (I)
12	Young at heart	1938
23	Cléopâtre	1934
28	Des soeurs courageuses	1938
29	Cléopâtre	1962
11	Mademoiselle Fifi	1944
16	Le voyage	1959
17	Quatre soeurs	1938
30	Casablanca	1944

FIG. 5 – Base de données $\overline{\textbf{movies}}$, BDAI de **movies**.

Preuve du théorème 1

Nous démontrons tout d'abord que la base de données $\overline{\boldsymbol{d}}$ *est bien une IADB pour* **d**. *D'après la proposition 3 page 12, et après application de l'algorithme 5, on a bien* $F_{\overline{\boldsymbol{d}}} = F_{\boldsymbol{d}}$ *et* $I_{\overline{\boldsymbol{d}}} = I_{\boldsymbol{d}}$. *Il suffit alors de montrer que* $F_{\boldsymbol{d}}$ *et* $I_{\boldsymbol{d}}$ *n'ont pas d'interaction, c'est à dire qu'aucune dépendance ne peut être inférée par* $F_d \cup I_d$ *sans appartenir ni à* F_d, *ni à* I_d.

Soit f *une dépendance (DI ou DF) telle que* $F_{\boldsymbol{d}} \cup I_{\boldsymbol{d}} \models f$. *Puisque* $d \models F_{\boldsymbol{d}}$ *et* $d \models I_{\boldsymbol{d}}$, *on a* $d \models f$. *Puisque* $F_{\boldsymbol{d}}$ *et* $I_{\boldsymbol{d}}$ *sont les ensembles de* toutes *les DF et DI satisfaites dans* d, *on a* $f \in F_{\boldsymbol{d}}$ *ou* $f \in I_{\boldsymbol{d}}$. *Donc* $F_{\boldsymbol{d}}$ *et* $I_{\boldsymbol{d}}$ *n'ont pas d'interaction.*

	movies	$\dot{movies}$	$\hat{movies}$	$\overline{movies}$
cardinalité	68262	374	374	434

TAB. 1 – Résulats expérimentaux

La démonstration du théorème 1 est alors naturelle à partir du lemme 2 et de l'algorithme 5. La borne inférieure correspond au cas où toutes les DI de $\mathcal{B}d^+(I)$ étaient satisfaites dans $\hat{\mathbf{d}}$. La borne supérieure est atteinte si, pour forcer chaque DI i de $\mathcal{B}d^+(I)$, il a fallu rajouter autant de valeurs dans la partie droite de i qu'il y a de lignes dans la partie gauche.

4.6 Cas d'un ensemble de DI circulaire

La méthode est élaborée de telle façon que chaque étape soit définitive, chaque élément de $\mathcal{B}d^-(I)$ et $\mathcal{B}d^+(I)$ n'étant considéré qu'une seule fois. Toutefois, dans le cas où $\mathcal{B}d^+(I)$ est circulaire, cette démarche ne semble plus possible. Forcer une DI entrante dans une relation R peut amener à contredire une autre DI sortante de R, elle même forcée auparavant. Dans ce cas, nous n'avons d'autre méthode que de traiter en boucle les cycles jusqu'à obtenir un état stable. Il est alors impossible de prévoir le nombre de fois que les DI seront parcourues. Rappelons toutefois que l'algorithme se termine à coup sûr, au pire en obtenant la base de départ. Ce qui n'est pas le cas pour le problème moins contraint de la construction de bases de données d'Armstrong "classiques", pour lequel, à notre connaissance, aucun algorithme n'existe dans le cas de DI circulaires en entrée.

5 Résultats expérimentaux

Le but de cette section est double : montrer la faisabilité de notre approche, et illustrer le pouvoir de réduction des BDAI sur une base de données réelle de cardinalité moyenne. La base de données **movies** provient du *UCI KDD Archive* [Bay, 1999]. La plus grande relation possède 40000 tuples et la cardinalité totale de la base est de 68262 tuples. Pour chaque étape de l'algorithme 1, la cardinalité de la BDAI en construction est reportée dans le tableau 1. Première étape : Nous avons calculé une relation d'Amrstrong informative pour chaque relation de **movies**, et ainsi calculé $\dot{movies}$ telle que $F_{\dot{movies}} = F_{\mathbf{movies}}$. La cardinalité de $\dot{movies}$ est reportée dans la colonne 3 du tableau 1.

Deuxième étape : dans la base de données **movies** 896 DI constituent la bordure négative des DI satisfaites. Pour chacune, l'algorithme 3 cherche d'abord si un tuple disqualifiant est déjà présent dans la base de données en construction. Dans cette expérimentation, aucun tuple disqualifiant n'a dû être inséré : pour chaque DI de la bordure négative, cela était déjà fait (colonne 4 du tableau 1).

Troisième étape : comme nous l'avons vu, il suffit de forcer la bordure positive des DI satisfaites dans **movies** ; cet ensemble est composé de 10 DI. Forcer ces DI a conduit à l'ajout de 60 tuples. Ainsi, on obtient une DBAI pour **movies** avec 434 tuples (colonne 5 du tableau 1) ; cela correspond à une réduction de **movies** par un facteur 150,

gardant exactement les même DF et DI. Précisons que, dans nos conditions expérimentales, le temps d'exécution pour l'ensemble de l'algorithme 1 est de quelques secondes seulement ; ce temps n'excède pas 30 minutes si l'on prend en compte la découverte des générateurs induits par les DF de **movies** ainsi que les bordures positives et négatives des DI satisfaites dans **movies**.

Étude de cas Nous avons cherché à observer le comportement d'utilisateurs des bases de données d'Armstrong informatives, afin de recueillir leurs impressions. Nous avons proposé à des étudiants de troisième année de master informatique de l'université Clermont-Ferrand II le cas d'études suivant : "Réaliser la ré-ingénierie de la base de données **movies** en utilisant le logiciel DBA Companion [Lopes *et al.*, 2004]". Il s'agit d'un prototype universitaire réalisé en C++ et dans lequel sont intégrés divers algorithmes permettant notamment de découvrir les DF et DI satisfaites dans une base donnée en entrée, ainsi que de construire une BDAI pour cette base. Leur premier réflexe fut de découvrir les clés et les clés étrangères satisfaites dans les données. Lorsque leur intuition initiale était contredite par le résultat, ils n'ont eu d'autre solution que d'observer une DBAI pour *movies* pour pouvoir aller un peu plus loin. Nous avons remarqué que, grâce à la petite taille de la BDAI générée, ils étaient capables d'effectuer instantanément des tris, des sélections, des projections et des jointures pour manipuler et visualiser les exemples et les contre-exemples. Les BDAI furent réellement un élément clé de leur travail, qui les a conduit à comprendre la conception logique de la base, compréhension nécessaire en amont de toute tâche de ré-ingénierie du schéma de la base de données.

6 Conclusion

Nous avons défini dans cet article un nouvel outil de visualisation des DF et DI satisfaites dans une base de données existante, les bases de données d'Armstrong informatives (BDAI). Vérifiant exactement les mêmes DF et DI que la base initiale, elles sont donc une représentation complémentaire à une simple liste spécialisée de dépendances. En outre, leur caractère informatif provient du fait que tous les tuples qui les peuplent sont réels, et proviennent des données originales. On peut voir ces échantillons comme un ensemble d'illustrations de toutes les dépendances satisfaites et non satisfaites dans la base. Elles peuvent aussi servir à tester différents processus de restructuration des données (par exemple de normalisation), les changements définitifs étant appliqués plus tard à la base réelle.

Nous avons démontré de façon constructive que, pour toute base de données en entrée, on pouvait construire une BDAI avec une taille bornée. Si l'ensemble des DI satisfaites dans la base départ est non-circulaire, alors la borne exhibée est linéaire dans le nombre de générateurs des DF et dans la taille des bordures positives et négatives des DI. Bien plus petite dans de nombreux cas pratiques que la taille de la base initiale, nous pensons en outre que cette borne n'est souvent pas atteinte ; en effet, l'étape de prise en compte des DI non satisfaites devrait souvent ne conduire qu'à l'ajout de très peu de tuples. Cette intuition est renforcée par les résultats obtenus sur la base de données

réelle **movies** issue du web, pour laquelle notre programme construit rapidement une BDAI 150 fois plus petite.

Cet article est, à notre connaissance, le premier à aborder réellement le problème de la visualisation des dépendances dans les bases de données relationnelles existantes. Certes, la visualisation de s'effectue pas de façon classique par le biais de représentations graphiques ; elle ne permet pas non plus de visualiser de façon rapide l'ensemble des contraintes découvertes. Toutefois, l'objectif est bien de rendre possible l'évaluation du résultat par l'œil humain et non expert des motif découverts, rendant possible l'exploitation de l'expertise humaine au bénéfice du résultat final. La mise en place d'approches plus graphiques pour la visualisation des dépendances reste une piste intéressante à explorer, pouvant s'inspirer naturellement des nombreux travaux sur la visualisation dans des processus de découverte de "clusters" ou de règles d'associations. A l'inverse, il semble aussi potentiellement intéressant de mettre en œuvre tes techniques de visualisation par l'exemple dans d'autres processus plus classiques de fouille de données.

Références

[Abiteboul *et al.*, 1995] S. Abiteboul, R. Hull, et V. Vianu. *Foundations of Databases.* Addison-Wesley, Reading, Mass., 1995.

[Armstrong et Delobel, 1980] W. W. Armstrong et C. Delobel. Decomposition of functional dependencies in relations. *ACM Transactions on Database Systems*, 5(4) :404–430, 1980.

[Armstrong, 1974] W. W. Armstrong. Dependency structures of database relationships. In J. L. Rosenfeld, editor, *International Federation for Information Processing (IFIP'74), Stockholm, Sweden*, pages 580–583, 1974.

[Bay, 1999] S. D. Bay. The UCI KDD Archive [`http ://kdd.ics.uci.edu`]. Technical report, Irvine, CA : University of California, Department of Information and Computer Science, 1999.

[Beeri *et al.*, 1984] C. Beeri, M. Dowd, R. Fagin, et R. Statman. On the structure of Armstrong relations for functional dependencies. *Journal of the ACM*, 31(1) :30–56, 1984.

[Casanova et de Sa, 1983] M. A. Casanova et J. E. Amaral de Sa. Designing entity-relationship schemes for conventional information systems. In C. G. Davis, S. Jajodia, P. A. Ng, et R. T. Yeh, editors, *International Conference on Entity-Relationship Approach (ER'83)*, pages 265–277. North Holland, 1983.

[Comyn-Wattiau et Akoka, 1999] I. Comyn-Wattiau et J. Akoka. Relational database reverse engineering. *Networking and Information Systems*, 2(3) :345–364, 1999.

[De Marchi *et al.*, 2002a] F. De Marchi, S. Lopes, et J.-M. Petit. Efficient algorithms for mining inclusion dependencies. In C. S. Jensen, K. G. Jeffery, J. Pokorný, S. Saltenis, E. Bertino, K. Böhm, et M. Jarke, editors, *International Conference on Extending Database Technology (EDBT'02), Prague, Czech Republic*, volume 2287 of *Lecture Notes in Computer Science*, pages 464–476. Springer, 2002.

[De Marchi *et al.*, 2002b] F. De Marchi, S. Lopes, et J.-M. Petit. Samples for understanding data-semantics in relations. In M.-S. Hacid, Z. W. Ras, D. A. Zighed, et Y. Kodratoff, editors, *International Symposium on Methodologies for Intelligent Systems (ISMIS'02), Lyon, France*, volume 2366 of *Lecture Notes in Artificial Intelligence*, pages 565–573. Springer, 2002.

[De Marchi et Petit, 2003] F. De Marchi et J-M. Petit. Zigzag : a new algorithm for discovering large inclusion dependencies in relational databases. In *International Conference on Data Mining (ICDM'03), Melbourne, Florida, USA*, pages 27–34. IEEE Computer Society, 2003.

[Demetrovics et Thi, 1995] J. Demetrovics et V.D. Thi. Some remarks on generating armstrong and inferring functional dependencies relation. *Acta Cybernetica*, 12(2) :167–180, 1995.

[Fagin et Vardi, 1983] R. Fagin et M.Y. Vardi. Armstrong databases for functional and inclusion dependencies. *Information Processing Letters*, 16 :13–19, 1983.

[Fayyad *et al.*, 2002] U. Fayyad, G. G. Grinstein, et A. Wierse, editors. *Information visualisation in data mining and knowledge discovery*. Morgan Kaufmann, 2002.

[Huhtala *et al.*, 1999] Y. Huhtala, J. Karkkainen, P. Porkka, et H. Toivonen. TANE : An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(2) :100–111, 1999.

[Laurent *et al.*, 1999] D. Laurent, J. Lechtenbörger, N. Spyratos, et G. Vossen. Complements for data warehouses. In *International Conference on Data Engineering (ICDE'99), 23-26 March 1999, Sydney, Austrialia*, pages 490–499. IEEE Computer Society, 1999.

[Levene et Loizou, 1999a] M. Levene et G. Loizou. *A Guided Tour of Relational Databases and Beyond*. Springer, 1999.

[Levene et Loizou, 1999b] M. Levene et G. Loizou. How to prevent interaction of functional and inclusion dependencies. *Information Processing Letters*, 71, 1999.

[Lopes *et al.*, 2002] S. Lopes, J.-M. Petit, et L. Lakhal. Functional and approximate dependencies mining : Databases and FCA point of view. *Special issue of JETAI*, 14(2/3) :93–114, 2002.

[Lopes *et al.*, 2004] S. Lopes, F. De Marchi, et J.-M. Petit. DBA Companion : A tool for logical database tuning. In *Demo session of International Conference on Data Engineering (ICDE'04)*, `http ://www.isima.fr/~demarchi/dbacomp/`, 2004. IEEE Computer Society.

[Mannila et Räihä, 1986] H. Mannila et K.-J. Räihä. Design by example : An application of armstrong relations. *Journal of Computer and System Sciences*, 63(2) :126–141, 1986.

[Mannila et Toivonen, 1997] H. Mannila et H. Toivonen. Levelwise Search and Borders of Theories in Knowledge Discovery. *Data Mining and Knowledge Discovery*, 1(1) :241–258, 1997.

[Markowitz et Makowsky, 1990] V.M. Markowitz et J.A. Makowsky. Identifying Extended Entity-Relationship Object Structures in Relational Schemas. *Transactions on Software Engineering*, 16(1) :777–790, 1990.

[Miller *et al.*, 2001] R. J. Miller, M. A. Hernández, L. M. Haas, L. Yan, C. T. H. Ho, R. Fagin, et L. Popa. The clio project : Managing heterogeneity. *ACM SIGMOD Record*, 30(1) :78–83, 2001.

[Novelli et Cicchetti, 2001] N. Novelli et R. Cicchetti. Functional and embedded dependency inference : a data mining point of view. *Information System*, 26(7) :477–506, 2001.

[Petit *et al.*, 1996] J.-M. Petit, F. Toumani, J.-F. Boulicaut, et J. Kouloumdjian. Towards the reverse engineering of denormalized relational databases. In S. Y. W. Su, editor, *International Conference on Data Engineering (ICDE'96), New Orleans, Louisiana*, pages 218–227. IEEE Computer Society, 1996.

[Quass *et al.*, 1996] D. Quass, A. Gupta, I. S. Mumick, et J. Widom. Making views self-maintainable for data warehousing. In *International Conference on Parallel and Distributed Information Systems (PDIS'96), Miami Beach, Florida, USA*, pages 158–169. IEEE Computer Society, 1996.

[Silva et Melkanoff, 1979] A. M. Silva et M. A. Melkanoff. A method for helping discover the dependencies of a relation. In H. Gallaire, J.-M. Nicolas, et J. Minker, editors, *Advances in Data Base Theory (ADBT'81)*, pages 115–133, Toulouse, France, 1979.

Summary

The understanding of relational databases semantic is an important task for many applications. This semantic is mainly conveyed by functional dependencies (FD) and inclusion dependencies (IND) ; they generalize respectively the notions of keys and foreign keys. However, it is a well known observation that operational databases become disordered over time ; in that case, integrity constraints have to be discovered from data. Several methods have been proposed for FD and IND discovery. These algorithms return to the administrator a set of satisfied dependencies.

Then, the administrator needs to understand extracted dependencies, and knowledge visualization issues have to be considered. For example, he must be able to select interesting rules, or to understand why an expected dependency is not satisfied in data. We propose to compute, as a complement to the listing of dependencies, a sample of the database verifying exactly the same FD and IND, called *informative Armstrong database* (IADB). We think such examples very adapted to facilitate discussions between data experts and database administrator. We give some properties concerning existence and size of IADB, and algorithms to compute them. Experiments on a real-life database from the web show the practical interest of our proposal.