IncFDs: un nouvel algorithme d'inférence incrémentale des dépendances fonctionnelles

Ghada Gasmi*

* Département des Sciences de l'Informatique
Faculté des sciences de Tunis
Campus universitaire, 1060 Tunis, Tunisie
ghada.gasmi@gmail.com

Résumé. L'inférence des dépendances fonctionnelles est l'une des problématiques les plus étudiées en bases de données. Elle a fait l'objet de plusieurs travaux qui ont proposé des algorithmes afin d'inférer, efficacement, les dépendances fonctionnelles pour les utiliser dans différents domaines : administration de bases de données, ré-ingénierie, optimisation des requêtes,etc. Toutefois, pour les application réelles, les bases de données sont évolutives et les relations sont fréquemment augmentées ou diminuées de tuples. Par conséquent, afin de s'adapter à ce cadre dynamique, une solution consiste à appliquer l'un des algorithmes, disponibles dans la littérature, pour inférer les dépendances fonctionnelles, après chaque mise à jour. Cette solution étant coûteuse, nous proposons, dans cet article, d'inférer les dépendances fonctionnelles d'une manière incrémentale. À cet effet, nous introduisons un nouvel algorithme, appelé INCFDS, et nous évaluons ses performances par rapport à l'approche classique d'inférence des dépendances fonctionnelles à partir d'une relation dynamique.

1 Introduction

Les dépendances fonctionnelles (DF) ont été largement étudiées dans le cadre des bases de données relationnelles. À l'origine, l'étude des DF a été motivée par le fait qu'elles peuvent être utilisées pour exprimer des contraintes vérifiées pour un schéma de relation indépendamment d'une instance particulière du schéma (Abiteboul et al. (1995)).

Un peu plus tard, l'étude des DF été motivée par une autre problématique. En effet, lors de la conception des systèmes d'information ou de bases de données, la connaissance de dépendances fonctionnelles était supposée connue. Toutefois, durant la vie d'une base de données, le problème se pose dans d'autres termes : quelles sont, à partir des ensembles des données stockées, les dépendances qui sont effectivement vérifiées ? Ce problème est connu comme celui de *l'inférence de dépendances fonctionnelles*, et s'il a fait l'objet de nombreux travaux (Wyss et al. (2001); Lopes et al. (2000); Yao et al. (2002); Novelli et Cicchetti (2001); Flach et Savnik (1999); Huhtala et al. (1999); Mannila et Raiha (1987)) [1], c'est parce que la réponse apportée à

1. Une étude détaillée portant sur ces algorithmes est présentée dans (Lopes et Novelli)

la question précédente est capitale dans plusieurs domaines d'application. Tout d'abord, dans la pratique, de nombreuses bases de données ne sont pas normalisées pour plusieurs raisons : (i) originellement des erreurs ont pu être commises ; (ii) des évolutions de schéma, non maîtrisées au cours du temps, ont pu avoir lieu ; (iii) des choix d'organisation privilégiant l'optimisation des requêtes ont pu être faits. Dans ces différents cas, l'administrateur de bases de données a besoin d'outils pour lui permettre d'effectuer la ré-organisation logique des bases ou de contrôler la satisfaction des dépendances fonctionnelles. Ensuite la ré-ingénierie des bases de données s'appuie sur la connaissance des dépendances fonctionnelles valides. Il s'agit, à partir d'un schéma logique, de reconstituer un schéma conceptuel.

1.1 Motivations

Pour les applications réelles, les bases de données sont évolutives et les relations sont fréquemment augmentées par de nouveaux tuples (ou diminuées de tuples). Par conséquent, afin de s'adapter à ce cadre dynamique, une solution consiste à appliquer l'un des algorithmes, disponibles dans la littérature, après chaque mise à jour, pour inférer les dépendances fonctionnelles. Toutefois, cette solution pourrait être désavantageuse notamment lorsque la taille et la fréquence de mise à jour de la base augmentent. Ainsi, dans un cadre dynamique, le problème d'inférence des dépendances fonctionnelles se pose dans d'autres termes : comment inférer les dépendances fonctionnelles à moindre coût ?

Dans cet article, nous apportons un élément de réponse à cette question. En effet, nous nous basons sur quelques résultats de la théorie des hypergraphes pour proposer un algorithme incrémental, appelé INCFDS, permettant d'inférer les dépendances fonctionnelles, à partir de celles qui ont été antérieurement inférées (*i.e.,* dépendances fonctionnelles vérifiées avant la mise à jour de la relation). En outre, nous présentons les résultats d'une étude empirique, menée sur des bases benchmark, afin de montrer le gain qu'apporte notre algorithme par rapport aux algorithmes classiques d'inférence des dépendances fonctionnelles.

La suite de ce papier est organisée comme suit. La section 2 présente les notions de base liées à la théorie des bases de données et à la théorie des hypergraphes. La section 3 donne un bref état de l'art sur les algorithmes dédiés à l'inférence des dépendances fonctionnelles. La section 4 introduit l'algorithme incrémental, dénommé INCFDS, que nous proposons pour inférer les dépendances fonctionnelles. La section 5 analyse quelques résultats d'expérimentations que nous avons menées sur des bases benchmark en vue d'évaluer les performances de l'algorithme INCFDS. La conclusion et les perspectives, qui font l'objet de la section 6, terminent ce papier.

2 Préliminaires

Cette section présente les éléments nécessaires à la compréhension de la suite du papier.

2.1 Base de données relationnelle

Dans ce qui suit, nous proposons un survol d'un ensemble de notions utilisées en théorie des bases de données relationnelles. Pour plus de détails, le lecteur pourra se référer à divers ouvrages traitant de ces notions (Mark et George (1999); Abiteboul et al. (1995)).

Soit $\mathcal{R}$ un ensemble fini d'attributs. Pour chaque attribut $A \in \mathcal{R}$, l'ensemble de toutes ses valeurs possibles se nomme domaine de A et se note $Dom(A)$. Un tuple sur $\mathcal{R}$ est une application $t : \mathcal{R} \rightarrow \bigcup_{A \in \mathcal{R}} Dom(A)$, où $t(A) \in Dom(A) \forall A \in \mathcal{R}$. Une relation est un ensemble de tuples. On dit que r est une relation sur $\mathcal{R}$ et que $\mathcal{R}$ est le schéma de r. Si $X \subseteq \mathcal{R}$ et t est un tuple, on note $t[X]$ la restriction de t à X.

Exemple 1 *Considérons l'exemple de la relation illustrée par le tableau 1. Par souci de concision, les attributs IdHot, NumCh, TypeCh, CatHot, Prix seront renommés, dans le reste de l'article, respectivement par A, B, C, D et E.*

IDHot (A)	NumCh (B)	TypeCh (C)	CatHot (D)	Prix (E)
1	100	1	2	50
4	101	1	2	50
1	102	2	2	70
1	200	1	2	50
2	101	3	3	100
2	200	1	3	70
1	100	3	2	50

TAB. 1 – *Relation exemple*

Une dépendance fonctionnelle sur $\mathcal{R}$ est une expression $X \rightarrow Y$, où $X, Y \subseteq \mathcal{R}$. La DF $X \rightarrow Y$ est satisfaite par une relation r (noté $r \models X \rightarrow Y$) si et seulement si $\forall t_i, t_j \in r$, $t_i[X] = t_j[X] \Rightarrow t_i[Y] = t_j[Y]$. La DF $X \rightarrow Y$ est triviale si $Y \subseteq X$. Elle est dite standard si $X \neq \emptyset$. Elle est minimale si Y ne dépend d'aucun sous-ensemble propre de X. On note $\mathcal{F}_r^+$ l'ensemble de toutes les DF satisfaites par la relation r : $\mathcal{F}_r^+ = \{X \rightarrow Y | X \cup Y \subseteq \mathcal{R}, r \models X \rightarrow Y\}$.

Soit $\mathcal{F}$ un ensemble de DF sur $\mathcal{R}$. On dit qu'une DF $X \rightarrow Y$ dérive de $\mathcal{F}$, noté $\mathcal{F} \models X \rightarrow Y$, si $X \rightarrow Y$ est satisfaite par toute relation satisfaisant toutes les DF de $\mathcal{F}$, *i.e,.* si $r \models \mathcal{F}$ alors $r \models X \rightarrow Y$. Nous notons par $lhs(\mathcal{F}, X)$ la collection de tous les ensembles d'attributs déterminant X. Si $\mathcal{G}$ est un ensemble de DF, la notation $\mathcal{F} \models \mathcal{G}$ signifie que $\mathcal{F} \models g, \forall g \in \mathcal{G}$. Les trois règles suivantes, nommées axiomes d'Armstrong (Armstrong (1974)), forment un système d'inférence juste et complet pour les DF : soient $\mathcal{R}$ un schéma relationnel et X, Y, $Z \subseteq \mathcal{R}$; (1) Si $Y \subseteq X$ alors $X \rightarrow Y$ (pseudo-réflexivité) ; (2) Si $X \rightarrow Y$, alors $XZ \rightarrow YZ$ (augmentation) ; (3) Si $X \rightarrow Y$ et $Y \rightarrow Z$, alors $X \rightarrow Z$ (transitivité). La fermeture de $\mathcal{F}$, notée $\mathcal{F}^+$, est l'ensemble de toutes les DF qui peuvent être inférées à partir de $\mathcal{F}$ à l'aide des axiomes d'Armstrong, *i.e.,* $\mathcal{F}^+ = \{X \rightarrow Y | \mathcal{F} \models X \rightarrow Y\}$.

L'ensemble $\mathcal{F}$ est une couverture de l'ensemble $\mathcal{G}$ si $\mathcal{F}^+ = \mathcal{G}^+$, *i.e.,* $\mathcal{F} \models \mathcal{G}$ et $\mathcal{G} \models \mathcal{F}$. Une couverture $\mathcal{F}$ est canonique si $\forall X \rightarrow Y \in F$, Y ne comporte qu'un attribut et si $X \rightarrow Y$ est minimale. Nous désignons par $\mathcal{MIN}\mathcal{F}_r$ la couverture canonique de $\mathcal{F}_r^+$.

2.2 Théorie des hypergraphes

Dans ce qui suit, nous rappelons quelques définitions et propriétés relatives aux hypergraphes (Berge (1976)).

Définition 1 ***(Hypergraphe)***
Un hypergraphe $\mathcal{H}$ est un couple $\mathcal{H}$ =($\mathcal{V}$,$\mathcal{E}$) tel que $\mathcal{V} = \{v_1, v_2..., v_n\}$ un ensemble fini d'éléments et $\mathcal{E} = \{e_1, e_2..., e_m\}$ est une famille de sous-ensembles de $\mathcal{V}$ tel que : $e_i \neq \emptyset$ et $\bigcup_{i=1}^{m} e_i = \mathcal{V}$. Les éléments de $\mathcal{V}$ sont appelés sommets tandis que les éléments de $\mathcal{E}$ sont appelés hyperarêtes de l'hypergraphe $\mathcal{H}$ [2].
Un hypergraphe partiel $\mathcal{H}_i$ de $\mathcal{H}$ est un hypergraphe, sur $\mathcal{V}$, qui contient les i premières hyperarêtes de $\mathcal{H}$. Un hypergraphe $\mathcal{H}$ =($\mathcal{V}$,$\mathcal{E}$) est simple si et seulement si pour chaque couple e_i,$e_j \in \mathcal{E}$ $e_i \subseteq e_j \Rightarrow i = j$. On note par Min($\mathcal{H}$) l'ensemble des hyperarêtes minimales de $\mathcal{H}$ tel que Min($\mathcal{H}$)=$\{e_i \in \mathcal{E} | \nexists e_j \in \mathcal{E}, i \neq j, e_j \subset e_i\}$. On note par Max($\mathcal{H}$) l'ensemble des hyperarêtes maximales de $\mathcal{H}$ tel que Max($\mathcal{H}$)=$\{e_i \in \mathcal{E} | \nexists e_j \in \mathcal{E}, i \neq j, e_i \subset e_j\}$.
Soient $\mathcal{H}$ =$\{e_1, ..., e_n\}$ et $\mathcal{G}$ =$\{e'_1, ..., e'_m\}$ deux hypergraphes. L'union des deux hypergraphes $\mathcal{H} \cup \mathcal{G} = \{e_1, ..., e_n, e'_1, ..., e'_m\}$. Le produit cartésien des deux hypergraphes $\mathcal{H} \vee \mathcal{G} = \{e_i \cup e'_j | i \in [1..n], j \in [1..m]\}$.

Définition 2 ***(Transversal)***
Soit un hypergraphe $\mathcal{H}$ =($\mathcal{V}$,$\mathcal{E}$). Un ensemble $\mathcal{T} \subseteq \mathcal{V}$ est appelé transversal de $\mathcal{H}$ si et seulement si son intersection avec toutes les hyperarêtes de $\mathcal{H}$ n'est pas vide. Un transversal $\mathcal{T}$ est dit minimal si et seulement si $\nexists \mathcal{T}' \subset \mathcal{T}$ tel que $\mathcal{T}'$ est un transversal. La collection de tous les transversaux minimaux de $\mathcal{H}$, notée par $Tr(\mathcal{H})$, est appelée hypergraphe transversal. L'hypergraphe transversal $Tr(\mathcal{H})$ est un hypergraphe simple et $Tr(\mathcal{H})$=$Tr(Min(\mathcal{H}))$.

Propriété 1 *(Berge (1976)) Soient $\mathcal{H}$ et $\mathcal{G}$ deux hypergraphes simples.*
$Tr(\mathcal{H} \cup \mathcal{G}) = Min(Tr(\mathcal{H}) \vee Tr(\mathcal{G}))$.

Propriété 2 *(Berge (1976)) Soient $\mathcal{H} = \{e_1, ..., e_n\}$ un hypergraphe et $\mathcal{H}_i = \{e_1, ..., e_i\}$ un hypergraphe partiel de $\mathcal{H}$. Alors, nous avons : $Tr(\mathcal{H}_i) = Min(Tr(\mathcal{H}_{i-1}) \vee \{\{v\}, v \in e_i\}))$.*

3 État de l'art sur l'inférence des DF

La première étude détaillée du problème d'inférence des DF a été présentée dans (Mannila et Raiha (1987)) et elle peut être formulée de la façon suivante : "Étant donnée une relation r, trouver une couverture de l'ensemble des DF satisfaites par r", *i.e.*, trouver une couverture de l'ensemble $\mathcal{F}_r^+$ (*c.f.*, voir sous-section 2.1). Comme les couvertures de $\mathcal{F}_r^+$ peuvent être de tailles diverses, on recherche plus précisément une "petite" couverture. En effet, la plupart des algorithmes proposés dans la littérature extraient la couverture canonique de r $\mathcal{MIN}\mathcal{F}_r$.

2. Dans le reste du papier, un hypergraphe sera identifié uniquement par l'ensemble de ses hyperarêtes

Vu les différentes applications de l'inférence des DF, telles que l'analyse de bases de données, la rétro-conception et l'optimisation de requêtes, cette problématique a fait l'objet de plusieurs travaux de recherche qui peuvent être classés en deux catégories.
La première catégorie, appelée *algorithmes orientée tuples*, [3] (Lopes et al. (2000); Wyss et al. (2001); Mannila et Raiha (1987); Flach et Savnik (1999)), regroupe les algorithmes analysant les tuples de la relation afin d'en extraire ensuite les DF. En effet, cette catégorie d'algorithmes formalise l'information découverte à partir d'un couple de tuples en introduisant les notions *d'ensembles en accord* (*c.f.*, voir section 4.1). Les parties gauches des DF minimales de la couverture canonique de r sont ensuite extraites à partir des ensembles en accord sans accéder à la relation. C'est l'idée principale des algorithmes de cette catégorie : les accès à la base de données se font lors du calcul des ensembles en accord (ou, de façon équivalente, des *ensembles en désaccord*) puis le calcul des dépendances se fait sans accéder aux données. Le calcul des dépendances fonctionnelles est alors obtenu à partir des ensembles en accord et ce en empruntant quelques résultats de la théorie des hypergraphes.

La seconde catégorie, dénommée *algorithmes orientée attributs* , [4] (Novelli et Cicchetti (2001); Huhtala et al. (1999); Yao et al. (2002)), regroupe les algorithmes qui génèrent un ensemble de DF candidates puis vérifient leur validité par la relation r. Tous les algorithmes orientés attributs ont le même souci : "réduire l'espace de recherche, c'est-à-dire le nombre de DF à vérifier". À cet effet, ils établissent des règles d'élagage de l'ensemble des parties de $\mathcal{R}$ de façon théorique et algorithmique. Par conséquent, les différences qui existent entre ces algorithmes résident dans la caractérisation des DF et aussi dans les propriétés induites permettant un élagage efficace.

4 Inférence incrémentale des dépendances fonctionnelles

4.1 Problématique

Pour les application réelles, les bases de données sont évolutives et les relations sont fréquemment augmentées par de nouveaux tuples (ou diminuées de tuples). Pour s'adapter à cette dynamicité, une solution intuitive consisterait à appliquer l'un des algorithmes, disponibles dans la littérature, afin d'inférer les dépendances fonctionnelles vérifiées par la nouvelle relation après chaque mise à jour [5]. Néanmoins, cette solution pourrait être inefficace notamment lorsque la taille de la relation et la fréquence de sa mise à jour augmentent. Par conséquent, pour de telles relations, le problème de l'inférence des dépendances fonctionnelles se pose dans d'autres termes. Comment inférer les DF vérifiées dans une relation mise à jour à moindre coût ?

Cependant, nous voyons que le problème d'inférence des dépendances fonctionnelles, vérifiées par une relation après chaque mise à jour, peut être ramené à un problème de *maintenance des dépendances fonctionnelles*. Ainsi, contrairement à l'approche classique d'inférence

3. Dans un contexte plus général, ce type d'algorithmes est à rapprocher de la classe des algorithmes ascendants (bottom-up) qui partent des données pour générer des hypothèses générales.

4. Cette catégorie d'algorithmes est à rapprocher de la classe des algorithmes descendants (top-down) qui utilisent une approche de type générer-tester.

5. Dans la suite du papier, l'opération de mise à jour est restreinte à l'opération d'ajout de tuple.

des DF, qui a pour entrée la relation mise à jour, nous proposons de nous baser sur les DF antérieures (*i.e.,* vérifiées par la relation avant sa mise à jour) pour déterminer le nouvel ensemble de DF vérifiées par la relation après sa mise à jour.
En d'autres termes, nous pensons qu'une approche incrémentale d'inférence des DF pourrait être avantageuse pour le cas d'une relation fréquemment mise à jour.

4.2 Présentation de l'algorithme INCFDS

Dans le cadre de notre travail, nous avons restreint la mise à jour à l'ajout d'un tuple, puisqu'elle constitue l'opération la plus fréquemment utilisée. À cet effet, nous adoptons le principe des approches orientés tuples (*c.f.*, voir section 3) pour proposer un nouvel algorithme, appelé INCFDS, permettant d'inférer les DF d'une manière incrémentale. Le pseudo-code de INCFDS est illustré par l'algorithme 1.

Données :
– une relation r
– un nouveau tuple t
– les dépendances fonctionnelles vérifiées par r : $\mathcal{MINF}_r$.
Résultat : les dépendances fonctionnelles vérifiées par $r \cup t$: $\mathcal{MINF}_{r \cup t}$.
début
2 Calcul des ensembles en accord induits par t;
3 Détermination des ensembles maximaux;
4 Inférence des nouvelles dépendances fonctionnelles
fin

Algorithme 1 : Présentation de l'algorithme `IncFDs`.

4.2.1 Calcul des ensembles en accord induits par le tuple t

Tout d'abord, nous commençons par présenter quelques définitions nécessaires à la compréhension de cette section.

Définition 3 ***(Ensemble en accord)***
Pour un couple de tuples d'une relation, l'ensemble en accord est l'ensemble de tous les attributs partageant la même valeur pour les deux tuples considérés. Formellement un ensemble en accord est défini comme suit :
Soient t et t' deux tuples de r et $X \subseteq \mathcal{R}$ un ensemble d'attributs. Les tuples t et t' sont en accord sur X si et seulement si t[X] = t'[X]. L'ensemble en accord de t et t' est défini de la façon suivante : ag(t,t') = $\{A \in \mathcal{R} | t[A] = t'[A]\}$. *Nous notons par ag(r) =* $\{ag(t,t') | t, t' \in r, t \neq t'\}$.

Définition 4 ***(Ensembles en accord induits par un tuple*** t***)***
Soient une relation r et un tuple t. Les ensembles en accord induits par le tuple t par rapport à la relation r sont définis comme suit : $Ag(r)_t = \{ag(t,t') | t' \in r\}$.

Définition 5 ***(Classe d'équivalence d'un tuple*** t***)***
La classe d'équivalence d'un tuple t par rapport à l'ensemble $X \subseteq \mathcal{R}$ est définie comme suit : $X_t = \{t' \in r | t[A] = t'[A] \forall A \in X\}$.

En effet, les classes d'équivalence d'un tuple t servent de base pour déterminer les ensembles en accord induits par t d'une manière efficace. Ainsi, $Ag(r)_t$ n'est pas calculé à partir des tuples de la relation r mais il est calculé à partir des classes d'équivalence de t. Par exemple, les classes d'équivalence du tuple $t7$ par rapport aux attributs de la relation r, illustrée par la table 1, sont :
$A_{t7} = \{1, 3, 4\}$; $B_{t7} = \{1\}$; $C_{t7} = \{5\}$; $D_{t7} = \{1, 2, 3, 4\}$; $E_{t7} = \{1, 2, 4\}$.
$Ag(r)_{t7} = \{ABDE, AD, ADE, C, DE\}$.

Le calcul des ensembles en accord induits par t constitue l'étape clef de l'algorithme IncFds. En effet, suite à l'ajout d'un tuple, certaines des DF qui ont été satisfaites par r peuvent ne plus l'être par la relation mise à jour. La question qui se pose alors est "Comment déterminer les ensembles des DF à maintenir (*i.e.,* qui ne sont plus satisfaites par la relation mise à jour) ?
Dire qu'une DF $X \rightarrow A$ est violée suite à l'ajout du tuple t, signifie qu'il existe au moins un tuple t' de r qui partage la même valeur de X avec t ($t[X] = t'[X]$) mais il a une valeur différente sur A ($t[A] \neq t'[A]$). C'est justement cette information qui est véhiculée par l'ensemble en accord induit par t.
Par conséquent, toutes les DF dont la partie droite n'est pas incluse dans les ensembles en accord induits par t sont candidates à être modifiées. Nous désignons par $\mathcal{CANDIDF}_r$ l'ensemble des DF à maintenir.

4.2.2 Détermination des ensembles maximaux

Comme nous l'avons vu précédemment, une DF $X \rightarrow A$ satisfaite dans r, dont la partie droite n'est pas incluse dans aucun ensemble en accord induit par le tuple t, sera violée suite à l'ajout de t. Afin de déterminer, à moindre coût, les DF candidates à être modifiées, soit $\mathcal{CANDIDF}_r$, il suffit donc de ne conserver que les plus grands ensembles en accord induits par t (i.e., *les ensembles maximaux*), notés par $MaxAg(r)_t$.

4.2.3 Inférence des nouvelles dépendances fonctionnelles

a. Fondements théoriques

Il a été montré dans (Kantola et al. (1992)) que les compléments des ensembles en accord maximaux servent de base pour le calcul des parties gauches des DF. En effet, il a suffit de remarquer qu'un ensemble d'attributs détermine A s'il n'est inclus dans aucun ensemble maximal pour A (*i.e.*, la collection des ensembles maximaux ne contenant pas A, notés $(MaxAg(r), A)$. Ceci peut être exprimé de façon équivalente en disant qu'un ensemble X doit avoir une intersection non vide avec les compléments de tous les ensembles maximaux pour A $(CMaxAg(r), A)$. Ce résultat a été formalisé en utilisant quelques résultats de la théorie des hypergraphes. En effet, les parties gauches des DF ayant A pour partie droite ne sont autre que les transversaux minimaux de l'hypergraphe simple formé par les compléments des ensembles maximaux pour A.

Cependant, dans notre approche, nous ne disposons que des ensembles maximaux induits par le nouveau tuple t. À ce niveau, la question suivante se pose : "Comment déterminer les

nouvelles parties gauches des DF $\mathcal{CANDIDF}_r$ sachant que nous avons $MaxAg(r)_t$?"
Pour tenter de répondre à cette question, nous nous sommes basés sur les résultats de la théorie des hypergraphes pour écrire la proposition 1 qui vise à caractériser, formellement, les DF à maintenir.

Proposition 1 *Soient ($CMaxAg(r)_t$,A) = $\{\varepsilon_1, ..., \varepsilon_i, ..., \varepsilon_n\}$ la collection des compléments des ensembles maximaux induits par le tuple t pour l'attribut A et $\sigma_1, ..., \sigma_i, ..., \sigma_n$ la collection des ensembles d'attributs calculés suite à la considération de $\varepsilon_1, ..., \varepsilon_i, ..., \varepsilon_n$, respectivement.*
Les dépendances fonctionnelles satisfaites par $r \cup t$ sont données par :

($\mathcal{MINF}_{r\cup t}$,A) = $\{X \rightarrow A | X \in \sigma_n\}$
Sachant que :

$$\begin{cases} \sigma_i = Min_{\subseteq}\{Y \cup \{B\} | Y \in \sigma_{i-1}, \{B\} \in \varepsilon_i\} \\ \sigma_0 = \{Y | Y \rightarrow A \in (\mathcal{MINF}_r, A)\} \end{cases}$$

Preuve 1 *Dans ce qui suit, nous prouvons qu'en appliquant notre approche, nous obtenons les mêmes DF générées par l'approche classique.*
Il est trivial que la collection des compléments des ensembles maximaux pour l'attribut A, qui sont calculés à partir de $r \cup t$, est donnée par ($CMaxAg(r \cup t)$,A) = $CMaxAg(r) \cup \{\varepsilon_1, ..., \varepsilon_i, ..., \varepsilon_n\}$. Nous savons aussi que cette collection forme un hypergraphe simple.
Alors, nous désignons par $\mathcal{H}_1, ..., \mathcal{H}_i, ..., \mathcal{H}_n$ les hypergraphes partiels donnés par, respectivement, ($MaxAg(r) \cup \varepsilon_1$),...,($MaxAg(r) \cup \varepsilon_1 \cup ... \cup \varepsilon_i$),..., ($MaxAg(r) \cup \varepsilon_1, ..., \varepsilon_i, ..., \varepsilon_n$).
Nous désignons par $\sigma_1, ..., \sigma_i, ..., \sigma_n$ leurs transversaux minimaux respectifs.
Pour n = 0, nous avons ($\mathcal{MINF}_{r\cup t}$,A) = $\{X \rightarrow A | X \in \sigma_0\}$ = ($\mathcal{MINF}_r$,A). Ceci est trivial puisque nous n'avons considéré aucun complément.
Cependant, pour le cas général, nous avons d'après la propriété 2 (c.f., voir section 2.2),
σ_i = $Min(\sigma_{i-1} \vee \{\{B\}, B \in \varepsilon_i\})$). Par conséquent, σ_n = $Min(\sigma_{n-1} \vee \{\{B\}, B \in \varepsilon_n\})$.
Ainsi, nous avons ($\mathcal{MINF}_{r\cup t}$,A) = $\{X \rightarrow A | X \in \sigma_n\}$.

A ce stade, nous avons montré qu'en utilisant, seulement, $CMaxAg(r)_t$ nous pouvons inférer les nouvelles dépendances fonctionnelles suite à l'ajout du tuple t. Ceci est très avantageux parce que la complexité du calcul de $CMaxAg(r \cup t)$ est en $O(nm^2)$, sachant que $n = |\mathcal{R}|$ et $m = |r|$, alors que celle de $CMaxAg(r)_t$ est en $O(nm)$.

b. Mise en oeuvre

Afin de maintenir les DF, ayant comme partie droite un attribut A, à partir des compléments $CMaxAg(r)_t$, nous proposons une fonction récursive qui prend en entrée $CMaxAg(r)_t$ et ($CMaxAg(r)$,A) et construit un arbre de recherche dont chaque noeud est une partie gauche candidate pour la nouvelle DF.
La fonction commence par initialiser le premier niveau de l'arbre par les parties gauches de ($CMaxAg(r)$,A). Pour passer au niveau i de l'arbre, nous considérons le i^{eme} complément de $CMaxAg(r)_t$. Deux cas peuvent se présenter :

1. Si le noeud X en question est un transversal du complément à considérer alors nous ignorons le complément i et nous passons au complément $i + 1$.

2. Si le noeud X en question n'est pas un transversal du complément à considérer, alors il faut générer des noeuds fils dont chacun est l'union de X et $\{B\}$, sachant que $\{B\} \in$ complément i. Il est à noter qu'un noeud fils n'est retenu qu'après avoir vérifié sa minimalité par rapport aux feuilles de l'arbre.

Exemple 2 *Supposons que nous disposons d'une relation r composée des cinq premiers tuples de la relation de la table 1 et que nous ajoutons le tuple T6. La figure 1 illustre le principe de détermination des nouvelles DF ayant comme partie droite l'attribut "C".*
Les DF satisfaites par les cinq premiers tuples de la relation, et dont la partie droite est l'attribut "C", sont $E \rightarrow C$, $AB \rightarrow C$ et $BD \rightarrow C$. Par ailleurs, les compléments des ensembles en accord maximaux induits par le tuple $T6$ sont $CMaxAg(r)_t6 = \{BCE, ABCD\}$. Nous initialisons le premier niveau de l'arbre par les parties gauches des anciennes DF qui déterminent "C". Alors, nous avons AB et BD deux transversaux minimaux de l'hypergraphe $\{BCE, ABCD\}$. Par conséquent, les deux DF $AB \rightarrow C$ et $BD \rightarrow C$ ne seront pas violées suite à l'ajout du tuple $T6$. Néanmoins, E est un transversal minimal de BCE. Ainsi nous considérons seulement le complément $ABCD$. Alors, nous obtenons AE, BE, CE et DE comme étant les nouvelles parties gauches des Df qui déterminent "C".

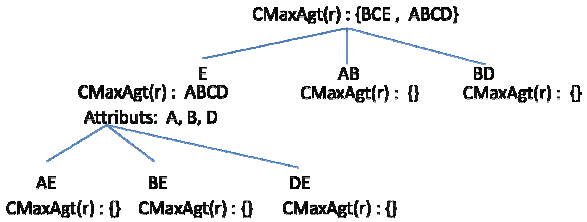

FIG. 1 – *Les parties gauches des nouvelles DF déterminant l'attribut "C".*

5 Évaluations expérimentales

Afin d'évaluer les performances de l'approche incrémentale proposée par rapport à l'approche classique d'inférence des DF, nous avons implémenté l'algorithme INCFDS et FASTFDS (Wyss et al. (2001)) en Java sur un Centrino 2 Duo à 2 Ghz et 3 Go de RAM sous Windows Vista. Par ailleurs, nous avons mené une série de tests sur des bases benchmark qui sont fréquemment utilisées pour évaluer les algorithmes de fouille de données. Les caractéristiques de ces bases sont présentées dans la table 2.

Les courbes de la figure 2 montrent l'évolution du temps requis pour l'inférence des DF suite à la variation du nombre de tuples à considérer.

À la lecture des courbes de la figure 2, nous pouvons constater que :

Base	$\lvert\mathcal{R}\rvert$	$\lvert r\rvert$
Flare	10	323
Nursery	8	12960
Mushrooms	23	8124
Credit	20	1000

TAB. 2 – *Caractéristiques des bases benchmark.*

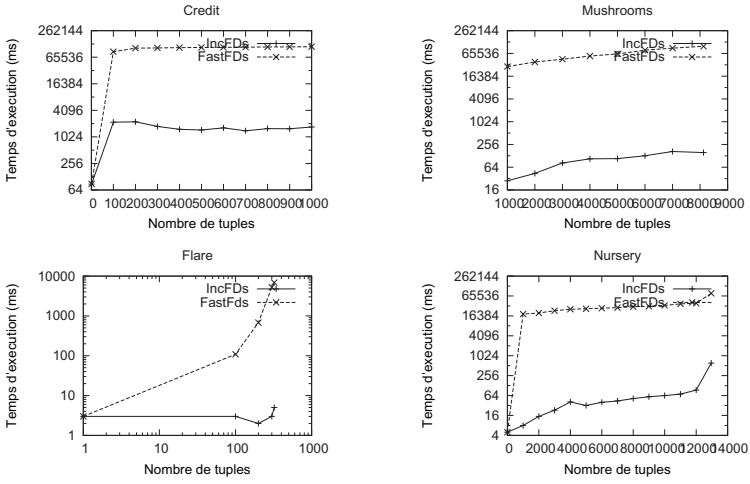

FIG. 2 – *Temps d'infrérence des DF vs. variation du nombre de tuples à considérer.*

1. Plus le nombre de tuples, à considérer, augmente plus le temps d'inférence des DF augmente. En effet, ceci peut être expliqué par le fait que l'étape de calcul des ensembles en accord induits par le nouveau tuple à insérer devient plus coûteuse.
2. Le temps d'inférence des DF peut dans certains cas diminuer. Ceci est dû au nombre de DF à maintenir. En effet, certaines des DF vérifiées par la relation avant sa mise à jour peuvent ne plus l'être suite à l'ajout d'un nouveau tuple. Par conséquent, plus le nombre de DF à maintenir augmente plus le temps d'exécution de l'algorithme INCFDS augmente. Ainsi, la diminution du temps d'inférence des DF, rapportée dans certains cas, est dû au nombre faible des DF à maintenir.
3. Plus le nombre de tuples à considérer augmente plus l'utilisation de l'algorithme INCFDS est avantageuse. Ceci peut être expliqué par :
   (a) l'algorithme FASTFDS calcule tous les ensembles en accord vérifiés par la relation mise à jour en comparant toutes les paires de tuples de la relation tandis que INCFDS ne calcule que les ensembles en accord induits par le nouveau tuple t.
   (b) l'algorithme FASTFDS se base sur les ensembles en accord qu'il a préalablement calculé, pour générer toutes les DF vérifiées par la relation mise à jour. Par contre, l'algorithme INCFDS ne regénère que les DF qui ont été violées suite à l'ajout du nouveau tuple.

6 Conclusion

Dans cet article, nous avons proposé un nouvel algorithme, appelé INCFDS, permettant de déterminer les DF vérifiées par une relation suite à l'ajout d'un nouveau tuple. Cette proposition est basée sur quelques résultats de la théorie des hypergraphes, qui nous ont permis de définir une nouvelle catégorisation des DF.
Les résultats des expérimentations, menées sur des bases benchmark, ont montré que dans le cadre de relations dynamiques, notre approche incrémentale apporte un gain très appréciable par rapport à l'approche classique d'inférence des DF.

Comme perspectives de prolongement de ce travail, il nous semble intéressant d'étudier d'autres opérations de mises à jour telles que la suppression et la modification de tuples. Par ailleurs, il serait intéressant d'étudier la possibilité de paralléliser l'algorithme INCFDS en vue d'améliorer ses performances.

Références

Abiteboul, S., R. Hull, et V. Vianu (1995). *Foundations of Databases*. Addison-Wesley.

Armstrong, W. W. (1974). Dependency structures of data base relationships. In *IFIP Congress*, pp. 580–583.

Berge, C. (1976). *Graphs and Hypergraphs*. North-Holland Mathematical Library 6, American Elsevier, 2ème édition.

Flach, P. A. et I. Savnik (1999). Database Dependency Discovery : A machine learning approach. AI Communications,volume 12 (3) : 139–160.

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

Kantola, M., H. Mannila, K. J. Raiha, et H. Siirtola (1992). Discovering functional and inclusion dependencies in relational databases. *International Journal of Intelligent Systems Vol 7*, 591–607.

Lopes, S. et N. Novelli. L'inférence des dépendances fonctionnelles. *Ingénierie des Systèmes d'Information Vol 9*.

Lopes, S., J.-M. Petit, et L. Lakhal (2000). Efficient discovery of functional dependencies and armstrong relations. In *Proceedings of the 7th International Conference on Extending Database Technology, EDBT, Konstanz, Germany*, pp. 350–364.

Mannila, H. et K.-J. Raiha (1987). Dependency inference. In *Proceedings of 13th International Conference on Very Large Data Bases VLDB, Brighton, England*, pp. 155–158.

Mark, L. et L. George (1999). *A Guided Tour of Relational Databases and Beyond*. London, UK : Springer-Verlag.

Novelli, N. et R. Cicchetti (January 2001). Fun : An efficient algorithm for mining functional and embedded dependencies. In *Proceedings of the 8th International Conference on Database Theory, London, United Kingdom*, pp. 189–203.

Wyss, C. M., C. Giannella, et E. L. Robertson (2001). FASTFDS : A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances. In *Proceedings of the Third International Conference of Data Warehousing and Knowledge Discovery DaWaK, Munich, Germany, September*, pp. 101–110.

Yao, H., H. J. Hamilton, et C. J. Butz (2002). Fd_{Mine} : Discovering functional dependencies in a database using equivalences. In *Proceedings of the International Conference on Data Mining (ICDM 2002), Maebashi City, Japan*, Volume 0, pp. 729.

Summary

The discovery of functional dependencies in relational databases is an important in the data mining domain. Motivated by many application domains, different approaches were proposed to extract functional dependencies from existing relations. Nevertheless all the proposed approaches suppose that the database is static and recompute functional dependencies whenever the database was updated. To avoid this recomputation, we propose, in this paper, a novel incremental approach allowing to determine the updated set of functional dependencies after the insertion of a new tuple. Results of the experiments carried out on benchmark databases showed that, compared to classical approach of functional dependencies extraction, our approach brings important profits in term of computation time.