Extraction de Dépendances Fonctionnelles Approximatives: une Approche Incrémentale

Ekaterina Simonenko* et Noël Novelli**

* LRI-CNRS UMR 8623
Université Paris-Sud XI; F-91405 Orsay Cedex
ekaterina.simonenko@lri.fr
** LIF-CNRS UMR 6166 – Case 901
Université d'Aix-Marseille; Faculté des Sciences de Luminy;
F-13288 Marseille Cedex 9
noel.novelli@lif.univ-mrs.fr

Résumé. La découverte de dépendances fonctionnelles (DF) à partir d'une relation existante est une technique importante pour l'analyse de Bases de Données. L'ensemble des DF exactes ou approximatives extraites par les algorithmes existants est valide tant que la relation n'est pas modifiée. Ceci est insuffisant pour des situations réelles où les relations sont constamment mises à jour.
Nous proposons une approche incrémentale qui maintiens à jour l'ensemble des DF valides, exactes ou approximatives selon une erreur donnée, quand des tuples sont insérés et supprimés. Les résultats expérimentaux indiquent que lors de l'extraction de DF à partir d'une relation continuellement modifiée, les algorithmes existants sont sensiblement dépassés par notre stratégie incrémentale.

1 Contexte

Les Dépendances Fonctionnelles (DF) représentent les contraintes d'intégrité les plus courantes et les plus importantes en Bases de Données (Mannila et Räihä (1994)). Une DF entre 2 attributs (X, Y) notée $X \rightarrow Y$ est vraie dans une relation si les valeurs de Y sont totalement déterminées par les valeurs de X (Codd (1970)). Le problème de l'extraction de DF est le suivant : "Étant donnée une relation r, trouver toutes les DF qui sont valides dans r". Les Dépendances Fonctionnelles Approximatives (DFA) généralisent les DF et sont définies comme "les DF qui sont presque valides dans r, i.e. quelques tuples doivent être retirés de la relation r pour que la DF $X \rightarrow Y$ soit vraie dans r" (Kivinen et Mannila (1995)). Des DFA apparaissent dans les relations s'il existe une dépendance naturelle entre les attributs mais certains tuples contiennent des erreurs ou représentent une exception. Comme des erreurs peuvent être présentes dans les BD, les DF approximatives sont particulièrement intéressantes.
Récemment, la taille des bases de données a augmenté significativement voire de façon infinie pour les flux de données, rendant les algorithmes existants inefficaces. Ces approches ne peuvent considérer que des relations figées. Quand un tuple est ajouté ou supprimé, l'ensemble des DF valides doit être recalculé.
Les algorithmes les plus efficaces pour l'inférence de DF sont TANE (Huhtala et al. (1998)),

DEPMINER (Lopes et al. (2000)), FASTFDS (Wyss et al. (2001)) et FUN (Novelli et Cicchetti (2001)). Ces approches extraient l'ensemble des DF minimales et non triviales, la couverture canonique de DF. Cette dernière est équivalente à l'ensemble des DF (Codd (1970)). D'un point de vue formel, DEPMINER et FASTFDS sont basés sur la caractérisation des parties gauches des dépendances minimales non triviales comme étant l'ensemble des transversaux minimaux d'un hypergraphe (Mannila et Räihä (1994)). En revanche, TANE et FUN énumèrent toutes les sources de DF possibles et déterminent si ces sources induisent des DF minimales non triviales. Concernant les DF Approximatives (DFA), leur extraction est faite de façon efficace par DEPMINER, TANE et FUN. TANE et FUN reprennent l'une des mesures d'erreur définie dans Mannila et Räihä (1994); Kivinen et Mannila (1995) (appelé $g3$) comme la proportion minimale de tuples qu'il suffit de supprimer de la relation pour que la DF soit satisfaite par tous les tuples restants. L'erreur est égale à 0 si la DF est exacte, et proche de 1 si la DF n'est vérifiée que par un petit nombre de tuples.
Ces travaux permettent l'extraction des DF d'une relation figée. A notre connaissance seule l'approche INCFDS (Gasmi (2010)) prend en compte les modifications (insertion de tuples uniquement) de la relation pour maintenir à jour l'ensemble des DF exactes valides. L'approche est basée sur la détection des modifications induites lors de l'insertion de tuples dans la relation sur les ensembles en accord (puis bien sûr sur les maximaux et leurs complémentaires).

2 Approche Incrémentale pour la découverte de DFA

Nous désignons par R un schéma de relation et r une relation sur R. Un *sous-ensemble maximal* S de $X \subseteq R$ est un sous-ensemble de X tel que $S \subset X$ et $|S| = |X| - 1$. La cardinalité d'une combinaison d'attributs X dans r, noté $|X|_r$ représente le nombre de valeurs distinctes de X dans r. $|r|$ représente la cardinalité de r c'est-à-dire le nombre de tuples de r. Pour chaque combinaison d'attributs X, l'ensemble de ses valeurs réelles, le domaine actif, dans la relation r est noté $ADom_r(X)$.

2.1 Définitions

Définition 1 *Dépendances Fonctionnelles (DF)*
Soit $X, A \subset R$ une combinaison d'attributs. La dépendance fonctionnelle entre X et A, notée $X \rightarrow A$, est valide dans r si est seulement si $\forall\, t_1, t_2$ deux tuples de r, si $t_1[X] = t_2[X]$, alors $t_1[A] = t_2[A]$.
$X \rightarrow A$ est une DF minimale si et seulement si : $\forall\, X' \subset X$, $X' \not\rightarrow A$.
$X \rightarrow A$ est une DF non-triviale si et seulement si : $A \not\subseteq X$.

Définition 2 *(Kivinen et Mannila (1995)) Dépendances Fonctionnelles Approximatives (DFA)*
Une DF entre X et A est dite approximative suivant l'erreur ε, notée $X \rightarrow^{\varepsilon} A$, si $g3(X \rightarrow A) \leq \varepsilon$, où la fonction $g3$ calcule la proportion, par rapport à $|r|$, des tuples qu'il faut retirer à r pour que la DF $X \rightarrow A$ soit valide.

2.2 Extraction Incrémentale de DFA

Soit $Remove_r(X \rightarrow^{\varepsilon} A)$, où X désigne une combinaison d'attributs et A un attribut, un ensemble minimal de tuples à retirer de la relation r pour que la DF exacte $(X \rightarrow A)$ soit

valide. Soit $e_r(X \rightarrow^{\varepsilon} A) = |Remove_r(X \rightarrow^{\varepsilon} A)|$, le nombre de tuples à enlever de r pour rendre la DF $(X \rightarrow A)$ valide. Dans la suite, $e_r(X \rightarrow^{\varepsilon} A)$ est souvent appelé "erreur".
Le Théorème suivant décrit l'influence de l'insertion d'un tuple t sur $e_r(X \rightarrow^{\varepsilon} A)$.

Théorème 1 *Étant donnée une DFA $X \rightarrow^{\varepsilon} A$ valide sur r, $e_{r \cup t}(X \rightarrow^{\varepsilon} A) \geq e_r(X \rightarrow^{\varepsilon} A)$*

Preuve : Soit t le tuple qui va être inséré. Considérons $t[X]$ la valeur de t sur la combinaison d'attributs X.
Si $t[X] \notin ADom_r(X)$: le tuple t ne peut pas violer la DF $X \rightarrow A$, puisque la valeur de t sur X est nouvelle. Donc, $e_r(X \rightarrow^{\varepsilon} A)$ reste le même : $e_{r \cup t}(X \rightarrow^{\varepsilon} A) = e_r(X \rightarrow^{\varepsilon} A)$.
Sinon $t[X] \in ADom_r(X)$:

1. Si $t[XA] \notin ADom_r(XA)$, on peut en déduire que soit $t[A] \notin ADom_r(A)$, soit $t[A]$ n'a encore jamais été associée à $t[X]$. Dans les 2 cas t viole la DF $X \rightarrow A$. Or, $t[XA] \notin ADom_r(XA) \Rightarrow t[XA] \notin ADom(Remove_r(X \rightarrow^{\varepsilon} A)[XA])$, et donc $Remove_{r \cup t}(X \rightarrow^{\varepsilon} A) = Remove_r(X \rightarrow^{\varepsilon} A) \cup t$ et $e_{r \cup t}(X \rightarrow^{\varepsilon} A) = e_r(X \rightarrow^{\varepsilon} A) + 1$. Il n'est donc pas nécessaire de recalculer $e_r(X \rightarrow^{\varepsilon} A)$.
2. Si $t[XA] \in ADom_r(XA)$:
   (a) Si $t[XA] \notin ADom(Remove_{r \cup t}(X \rightarrow^{\varepsilon} A)[XA])$,
   alors $t[XA] \in ADom(\ Remove_{r \cup t}(X \rightarrow^{\varepsilon} A)[XA])$ puisque
   $t[XA] \in ADom_r(XA)$, donc t satisfait la DF $X \rightarrow A$ et $e_r(X \rightarrow^{\varepsilon} A)$ ne change pas.
   (b) Si $t[XA] \in ADom(Remove_{r \cup t}(X \rightarrow^{\varepsilon} A)[XA])$, ce qui signifie que les tuples avec la même valeur sur XA que t, sont considérés comme non-vérifiant la DF $X \rightarrow A$. Malheureusement, ajouter t dans $Remove_r(X \rightarrow^{\varepsilon} A)$ ne suffit pas, car après avoir incrémenté la cardinalité de $Remove_r(\ X \rightarrow^{\varepsilon} A)$, il est nécessaire de vérifier sa minimalité ce qui implique le recalcule complet de $Remove_{r \cup t}(X \rightarrow^{\varepsilon} A)[XA]$. □

Il est clair que lors de l'insertion d'un tuple, l'erreur ne peut qu'augmenter de 1. De plus, la preuve de ce théorème énumère tous les cas à considérer et permet donc de ne recalculer $e_{r \cup t}(X \rightarrow^{\varepsilon} A)$ que lorsque c'est strictement nécessaire.

Le théorème suivant donne le résultat analogue pour la suppression d'un tuple.

Théorème 2 *Étant donnée une DFA $X \rightarrow^{\varepsilon} A$ valide sur r, $e_{r \setminus t}(X \rightarrow^{\varepsilon} A) \leq e_r(X \rightarrow^{\varepsilon} A)$*

Preuve : La preuve de ce théorème n'est pas explicitée ici par manque de place mais elle suit le même cheminement que celle du théorème 1. □
Pour maintenir l'ensemble des DFA valides dans une relation selon une erreur ε donnée, une représentation interne basée sur la caractérisation des DFA introduite dans Novelli (2000) est utilisée. Celle-ci est une table contenant pour chaque combinaison d'attributs X, sa quasi-fermeture approximative X_r° et sa fermeture approximative $X_r^{\oplus}$. Quand un tuple est ajouté ou supprimé d'une relation, pour certaines DFA, l'erreur $e_r(X \rightarrow^{\varepsilon} A)$ change, et donc la représentation interne doit être mise à jour pour découvrir le nouvel ensemble de DFA minimales valides.

2.3 Algorithme AFD-DYNAMICUPDATE

L'algorithme, AFD-DYNAMICUPDATE, que nous décrivons ci-après maintient à jour l'ensemble des DFA valides dans une relation suivant une erreur donnée. Il s'appuie sur les théorèmes 1 et 2 ainsi que sur leur preuve pour minimiser le nombre de recalcul d'erreur de chaque DFA. Le recalcule de l'erreur n'est appelé que si c'est strictement nécessaire. Lorsqu'un tuple est ajouté, la représentation correspondante doit être mise à jour. Cela consiste à vérifier quelles DFA doivent être ajoutées ou supprimées de l'ensemble des DFA valides. Une approche par niveau est utilisée pour le parcours des candidats.
Pour la suppression, seule la fonction `RecalculateError` est modifiée suivant le Théorème 2.

Algorithm 1 AFD-DYNAMICUPDATE (R, $TupleInQuestion$, ε, $g3[]$ $Levels$)

Input :
R : L'ensemble d'attributs de la relation considérée
$TupleInQuestion$: Le tuple inséré
ε : L'erreur maximale admissible
Input/Output :
$g3[]$: Les erreurs (par rapport à $|r|$) de DF $X \rightarrow^{\varepsilon} A$
$Levels$: La représentation

for all $i \in [1..Levels.NbLevel]$ **do**
  for all candidate $l \in L_i$ **do**
    for all subset $s \subset l.candidate$ / $|s| = |l.candidate| - 1$ **do**
      $X := s.candidate$
      $A := \{l.candidate - s.candidate\})$
      $g3[X \rightarrow^{\varepsilon} A] := RecalculateError(l, Remove_r(X \rightarrow^{\varepsilon} A), ErrorChanges)$
      if $ErrorChanges$ **then**
        if $g3[X \rightarrow^{\varepsilon} A] \leq \varepsilon$ **then**
          if $A \notin s.closure$ **then**
            $s.closure := s.closure \cup A$
            for all superset $S \supset s.candidate$ / $|s| = |l.candidate| + 1$ **do**
              $UpdateQuasiClosureAdd(S, A)$
        else if $g3[X \rightarrow^{\varepsilon} A] > \varepsilon$ **then**
          if $A \in s.closure$ **then**
            $s.closure := s.closure - A$
            for all superset $S \supset s.candidate$ / $|s| = |l.candidate| + 1$ **do**
              $UpdateQuasiClosureDelete(S, A)$
  DisplayFDs($L_i - 1$)

2.4 Expérimentations pour AFD-DYNAMICUPDATE

L'algorithme a été implémenté en C++ avec QT4. Plusieurs expérimentations ont été réalisées sur un ordinateur équipé d'un processeur Pentium 4 cadencé à 3GHz avec 1Go de RAM. Pour les comparaisons entre AFD-DYNAMICUPDATE et FUN, nous commençons à partir d'une relation vide et ajoutons un par un chaque tuple (donc #tuples fois) [1]. Pour chaque

1. Les résultats obtenus pour les suppressions sont similaires à ceux obtenus pour l'ajout.

tuple inséré, l'approche FUN est exécutée. Les temps d'exécution de FUN donnés dans les comparaisons correspondent à la somme des temps de calcul de FUN pour chaque relation modifiée. Les figures 1(A) et 1(B) détaillent pour différentes valeurs des paramètres des relations synthétiques, les temps d'exécution de AFD-DYNAMICUPDATE. La figure 1(A) montre que AFD-DYNAMICUPDATE est indépendant de la corrélation des données (30%, 50%, 70%), et bien sûr qu'il est exponentiel au nombre d'attributs. La figure 1(B) illustre que l'algorithme est linéaire au nombre de tuples ajoutés dans la relation.

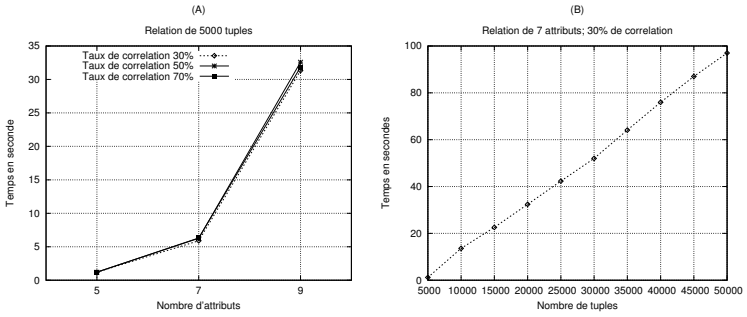

FIG. 1 – *Temps d'exécution pour différents nombres d'attributs, différentes corrélations, et différents nombres de tuples*

La table suivante décrit les caractéristiques des données réelles et les temps d'exécutions avec AFD-DYNAMICUPDATE et FUN.

Relation names	#attributes	#tuples	AFD-DYNAMICUPDATE	FUN
TombNecropolis	7	1 846	5,700s	82,770s
someCompanies	6	40 316	11m15s	1h 27m

Le tableau ci-dessus est très clair : notre approche est beaucoup plus efficace que FUN pour le calcul incrémental de Dépendances Fonctionnelles.

3 Conclusion et perspectives

Nous avons proposé une nouvelle approche pour l'extraction de DFA et un algorithme associé nommé AFD-DYNAMICUPDATE. Contrairement à d'autres algorithmes, celui-ci est une approche incrémentale qui permet la mise à jour de l'ensemble des DFA valides quand la relation évolue. Nous caractérisons l'influence de l'évolution d'une relation sur l'ensemble des DFA valides. Pour cela, nous n'utilisons pas la relation, mais une représentation de celle-ci, ce qui rend l'algorithme efficace et exploitable.
Pour montrer les avantages et la faisabilité de notre approche, nous avons comparé AFD-DYNAMICUPDATE avec FUN. Les résultats des expérimentations obtenus sous les mêmes conditions (cf. Section 2.4) montrent de bonnes propriétés de passage à échelle de l'approche lorsque la relation change (ajout ou suppression de tuples).
Par ailleurs, ce travail sera utilisé pour la détection visuelle de changements d'habitude ou de

détections d'erreurs. Un outil de visualisation *on line* et *off line* (à l'aide de scénarii, séquence d'instructions d'ajout et de suppression de tuples) est en cours de réalisation pour visualiser l'évolution de l'ensemble des DF valides au cours du temps. Cet outil montrera les différents ensembles de DF (exactes ou approximatives) valides pour une relation à plusieurs instants ce qui permettra de mieux comprendre les changements d'habitudes ou les apparitions d'erreurs.

Références

Codd, E. (1970). A Relational Model of Data for Large Shared Data Banks. *Communication of the ACM 13*(6), 377–387.

Gasmi, G. (2010). Incfds : un nouvel algorithme d'inférence incrémentale des dépendances fonctionnelles. In *EGC'10*, pp. 303–314.

Huhtala, Y., J. Karkkainen, P. Porkka, et H. Toivonen (1998). Efficient Discovery of Functional and Approximate Dependencies. In *ICDE*, pp. 392–401.

Kivinen, J. et H. Mannila (1995). Approximate Dependency Inference from Relations. *Theoretical Computer Science 149*(1), 129–149.

Lopes, S., J. Petit, et L. Lakhal (2000). Efficient Discovery of Functional Dependencies and Armstrong Relations. In *EDBT*, pp. 350–364.

Mannila, H. et K. Räihä (1994). Algorithms for Inferring Functional Dependencies from Relations. *Data and Knowledge Engineering 12*(1), 83–99.

Novelli, N. (2000). *Extraction de Dépendances Fonctionnelles dans les Bases de Données : une Approche Data Mining*. Ph. D. thesis, Université Aix-Marseille II.

Novelli, N. et R. Cicchetti (2001). Functional and Embedded Dependency Inference : A Data Mining Point of View. *Information Systems 26*(7), 477–506.

Wyss, C. M., C. Giannella, et E. L. Robertson (2001). Fastfds : A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances - extended abstract. In *DaWaK*, pp. 101–110.

Summary

Functional Dependency (FD) inference from existing relations is an important data base analysis technique. The problem has been treated by using data mining approach, by the algorithms TANE, DEPMINER, FASTFDS et FUN. Since the FD set, inferred by these algorithms is valid as long as the relation remains unchanged, they become efficient in the real life situations when the relation is constantly updated. It is particularly important to be able to extract approximate FDs, allowing to take into account errors and exceptions in the given data. We propose an incremental approach, allowing to update the valid set of approximate FDs, while tuples are inserted or deleted. The algorithm employs a level-wise strategy to keep the internal representation of the relation up-to-date. Experimental results show that during FDs extraction from a constantly updated relation, existing algorithms are significantly outperformed by our incremental approach.