Inclusion Dependency 検出の効率化手法の検討

宇野 洋明[1]　　齋藤 和広[1,2]　　川島 英之[1]

1. はじめに

リレーション R, S があるとき,「R のあるカラム（の組）の値はいずれのタプルでも，S のあるカラム（の組）の値の一部である」という関係がみられる場合がある。このような関係を，Inclusion Dependency (IND)という。この IND が「いくつのカラムどうしの関係か」の指標を arity という。R の n 個からなるカラムの組の値が，S の n 個からなるカラムの値に含まれるとき，n-ary の IND の関係がある。

IND の検出は，データプロファイリングのひとつとして位置づけられ[1], データ統合の前段階の処理として活用できる。

この IND を検出するアルゴリズムのひとつとして，FAIDA[3]がある。FAIDA には効率化手法のひとつとして，低次の arity の結果に基づいて IND の候補を生成する仕組みがある。今回は，FAIDA にみられる全体的な枠組みを援用しつつ，IND 検出プログラムの実装を行った。

2. IND 検出手法

IND を検出するプログラムの全体的な流れは，次のようになる。（図 1 にも示す。）以下，リレーション R について，カラム数 $|R|$，i番目のカラム（射影）を$R[i]$と表す。また，タプル数を$|r|$，i番目のタプル（選択）をr_iで表す。S についても同様である。

(1) 指定された 2 つのリレーション R, S を読みこみ, arity の上限を求める。arity の上限は，$min\{|R|, |S|\}$である。

(2) n (arity)を 1 から上限まで増やしながら，以下の候補生成と IND チェックを繰り返す。

- 候補生成：R，S から，n 個ずつカラムを選択し，IND になりうる候補を生成する。
- IND チェック：生成された候補について，データの値に基づき IND かどうかチェックする。この結果を保存，出力する。

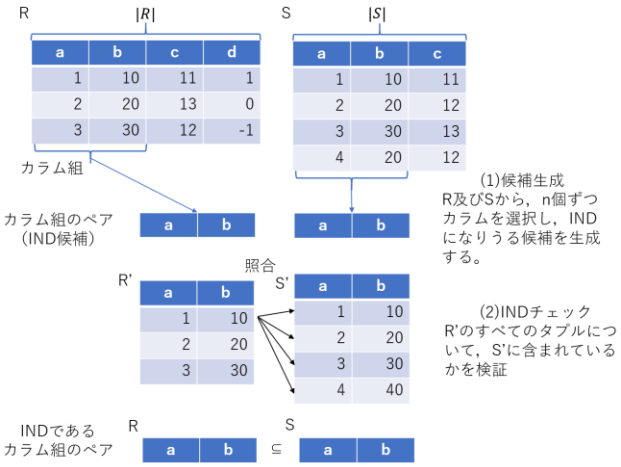

図 1 IND 検出の流れ

今回, 候補生成と IND チェックを繰り返す基本的な手続きを採用しつつ，候補生成において，「直前の arity 及び 1-ary の IND の結果を組み合わせることで，n-ary の候補を生成する」手法をとりいれ，また，IND チェックのために，Position List Index(PLI)[2]を用いたハッシュ化を採用する。以上を組み合わせて，IND 検出アルゴリズムを実装する。実装は，C による。

2.1. 候補生成

n-ary における候補生成とは，R および S からそれぞれ n 個のカラムを選択し，(IND 関係となる可能性のある) カラム組のペアを生成する手続きである。

[1] 慶應義塾大学
[2] 株式会社 KDDI 総合研究所

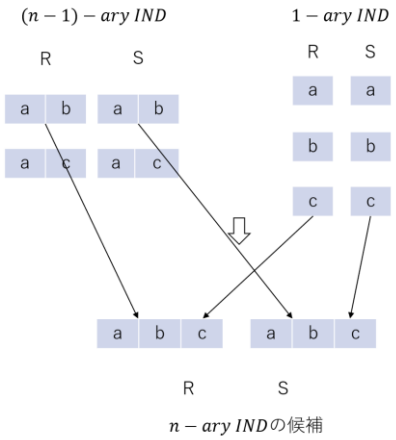

図 2 直前の arity と 1-ary の IND の結果の組み合わせ

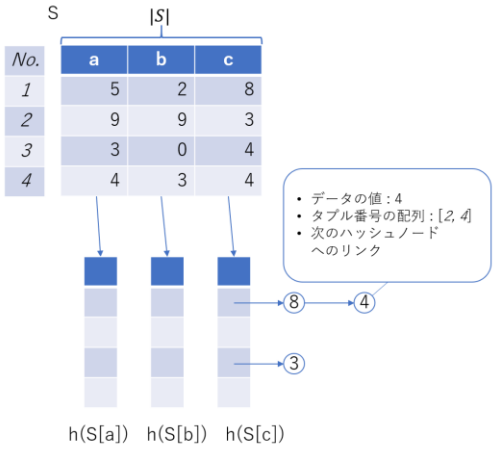

図 3 PLI を組み入れたハッシュ化

本手法では,「(n-1)-arity で IND であると示されたカラム組」に,「1-ary で IND であると示されたカラムのペア」を足し合わせることで，n-ary の候補を生成する。（図 2）この手続きは (n-1)-ary の IND の一覧が必要なため，$n = 1$では手法を用いたのち，$n \leq 2$ではじめてこの方法を採用する。この手法により, IND となりえない候補は生成されず，続く IND チェックを省力化できる。

2.2. IND チェック

IND チェックの段階は, 生成された候補が IND の関係にあるか，すなわち候補のペアの一方が他方に含まれているかを，タプルを走査することによって検証する手続きである。

この IND チェックの処理は, 探索対象のリレーションをあらかじめハッシュ化しておくことで効率化することができる。

ハッシュテーブル作成には，「リレーションを読み込んだ直後に，カラム 1 つずつに対して生成する」方法を採用し，チェイン法で実装した。このとき，ハッシュテーブルは$|S|$個作成される。まず，あらかじめ S のハッシュテーブルをカラムごとに作成しておく。このとき，図 3 に示すように, ハッシュテーブルのデータ構造として PLI を採用し，当該カラムに含まれる値だけでなく，その値が含まれる「タプル番号」も保存する。これは，2-ary 以上のとき，あるカラムの値と別のカラムの値が，同じタプルのものかを確認するためである。

n -ary の IND チェックを実行する際には，R のセルの値で，S の対応するハッシュテーブルを検索する。カラム組のペアは n カラムからなるので，これを繰り返す。セルの照合回数は，$|r| \times n$まで削減できる。

3. おわりに

今回, 直前の arity と 1-ary の IND の結果を組み合わせる候補生成と，ハッシュ化の際のデータ構造として PLI を採用した IND チェックを組み合わせて, IND 検出プログラムを実装した。今後，実データを通してこの効率化の程度を検証する予定である。

参考文献

[1] Abedjan, Z., Golab, L. and Naumann, F., Profiling Relational Data: A Survey, *The VLDB Journal*, Vol. 24, No. 4, pp. 557-581 (2015).

[2] Heise, A., Quiané-Ruiz J., Abedjan, Z., Jentzsch, A., Naumann, F.: Scalable Discovery of Unique Column Combinations, *Proc. VLDB Endow.*, Vol. 7, No. 4, pp. 301-312 (2013).

[3] Kruse, S., Papenbrock, T., Dullweber, C., Finke, M., Hegner, M., Zabel, M., Zöllner, C. and Naumann, F.: Fast Approximate Discovery of Inclusion Dependencies, *Datenbanksysteme für Business, Technologie und Web (BTW 2017),* pp. 207-226 (2017).