データ交換フレームワークにおける
ターゲット側公開ポリシ導出アルゴリズムの実装に向けて

2017SC047 中村俊貴
指導教員：石原靖哲

1　はじめに

近年，公共交通機関の交通データとスマートフォンアプリが連携した会員サービスの普及などに伴い，データ交換・統合 [2] の技術に注目が集まっている．データ交換フレームワークにおける重要なデータの公開には制限を設ける必要がある．通常のデータ交換設定では，参加者はソース側とターゲット側の 2 種類である．しかし，実際の状況では，ソース側とターゲット側が交換されたデータを公開する他の参加者も存在する．そして，ソース側とターゲット側では一般にデータベースの構造が異なるため，ターゲット側の公開ポリシを表す問合せを，ソース側の公開ポリシを表す問合せと同じにすることができない．しかも，仮にソース側とターゲット側のデータベースの構造が同じであったとしても，公開ポリシとして同じ問合せを採用すると，ソース側のデータの秘匿性が失われる場合があることがわかっている [5].

文献 [4, 5] では問合せ解像度という概念を用い，ターゲット側公開ポリシがソース側ポリシを適切に反映しているための安全性要件を定義している．さらに，文献 [3] では，問合せ解像度より限定された概念である CQ-rewriting という概念を用いて，ターゲット側の適切な公開ポリシを見つけるアルゴリズムを提案している．しかし，アルゴリズムの実装は行われておらず，現実的な時間でターゲット側ポリシを出力できるのか不明である．

そこで本研究では，文献 [3] で提案されたアルゴリズムの実装を Prolog を用いて実装した．また，未完成の部分については実装方法の提案をした．

2　諸定義

2.1　データ交換フレームワーク

本研究で対象とする，情報の提供者であるソース側と，ソース側からデータを受け取るターゲット側でのデータのやり取り [5] について図 1 に示す．

図 1 において，ソース側を一次情報提供者として，ターゲット側を二次情報提供者とする．それぞれは，直接のユーザ A とユーザ B を持ち，一次情報提供者と二次情報提供者では一般にデータベースの構造が異なるとする．そのため，マッピング M によりデータを共有する．

2.2　適切なポリシ

Q_S を一次情報提供者でのデータ公開ポリシを表す連言問合せとする．M を，一次情報提供者と二次情報提供者の間のマッピングを表す連言問合せの系列とする．Q_S と

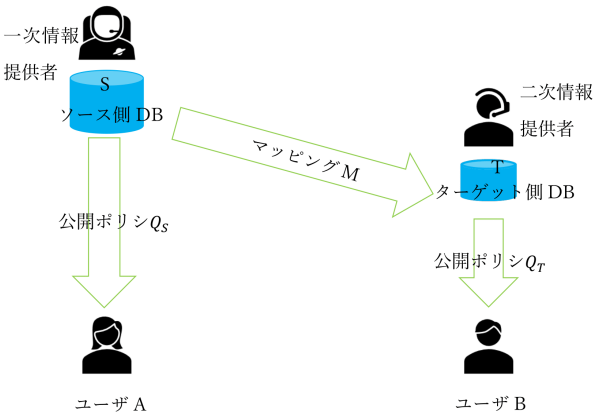

図 1　データ交換フレームワーク

M に対する二次情報提供者での適切なポリシー Q_T は，以下で定義される秘匿性と可用性を満たす連言問合せである [3].

- 秘匿性：$Q \circ Q_S$ と $Q_T \circ M$ が等価となるような連言問合せ Q が存在する．このような Q を「Q_S を使用した $Q_T \circ M$ の CQ-rewriting」と呼ぶ．
- 可用性：$Q_T \circ M$ は，CQ-rewriting の存在に関して極大の情報を提供する．すなわち，秘匿性を満たすすべての Q'_T について，Q'_T を使用した Q_T の CQ-rewriting がある場合，Q_T を使用した Q'_T の CQ-rewriting が存在する．

2.3　canonical rewriting

文献 [3] のアルゴリズムでは canonical rewriting という技法 [1] が用いられている．この技法は CQ-rewriting の存在判定に用いられる．M を用いた Q_S の canonical rewriting R_{can} とは次のような連言問合せである．

- R_{can} の左辺は Q_S の左辺と同じである．
- R_{can} の右辺は Q_S の右辺上の M に対するすべての解から成る．このとき，Q_S に現れる変数は一時的に定数とみなす．

3　適切なポリシの導出アルゴリズムの実装

文献 [3] では，与えられた Q_S と M に対し，秘匿性を満たす Q_T を導出するアルゴリズムが提案されている．このアルゴリズムは以下の 2 ステップからなる．

1. 秘匿性を満たすポリシの有限集合 $\mathcal{Q}$ を求める．
2. $\mathcal{Q}$ の中で極大の情報を提供するポリシを選び，出力

する．

以下では，次のような Q_S と M が与えられた場合を例として，アルゴリズムおよびその実装の説明を行う．

$$Q_S : V_S(A, C, D, E) \ :- \ S_1(A, b), S_2(b, C, D, E)$$
$$M : T_1(A, B) \ :- \ S_1(A, B)$$
$$T_2(C) \ :- \ S_2(b, C, D, e)$$

3.1 ステップ 1

ステップ 1 では，本来無限集合である秘匿性を満たすポリシ集合の有限部分集合 $\mathcal{Q}$ を求める．このステップでは，Q_S の左辺の変数（上の例では A, C, D, E）の一部に，M の右辺の定数（上の例では b, e）を代入した問合せを考え，それぞれの canonical rewriting を $\mathcal{Q}$ の要素の候補とする．そして，候補それぞれについて秘匿性を満たすかどうかを，再び 2.3 節の技法を用いてチェックする．秘匿性を満たした候補の集合が $\mathcal{Q}$ である．

本研究では，ステップ 1 のプログラムを前半部分と後半部分に分けて実装した．前半部分は，Q_S の左辺の変数と M の右辺の定数のペアの総当たりを求める．後半部分では，2.3 節の技法を前半部分で求めた結果に適用するために，M の左辺で前半部分の結果を問合せる．その後，文献 [3] のアルゴリズムに基づき，ポリシの候補を絞り込む処理を行うことで，秘匿性を満たすポリシの有限集合 $\mathcal{Q}$ を求める．このとき，変数を一時的に定数にする処理を，ファイルに結果を書き出すことで実現した．

```
toim(A,B,C,D,E,F,G,Head,Body) :-
%testはs1,s2を一つずつファクトとして登録する
test,
%ファクトとして登録したs1,s2を
適切な順番で呼び出す処理
nth_clause(s1(_,_),N,Reference_1),
clause(s1(_1,_2),_,Reference_1),
nth_clause(s2(_,_,_,_),N,Reference_2),
clause(s2(A,B,C,D),_,Reference_2),
%前半のプログラムの変数と紐づける
variable_names([_,_,_,_],[A,B,C,D]),
Head_1 = vs(A,B,C,D),
Body = (t1(E,F),t2(G)),
%以下ポリシの候補を絞り込む部分
%make_listはt1,t2の中身をリストとして書き出す
make_list(Head_1,Body,P,H,B),
%checkはHの各要素Xについて,
XがBに現れるかどうかをチェックする
check(H,B,L),
Head = [P|L].
%以下make_listやcheckについての定義が存在する
```

上記のプログラムでは，ユーザが toim を実行することを想定している．第 1 引数から第 4 引数までは，Q_S の左辺の変数が入力される．第 5 引数から第 7 引数までは，M の左辺の変数が入力される．toim が実行されると，test が呼び出され，前半部分の結果を書き出したファイルの S_1 と S_2 に当たる部分を一つずつファクトとして登録する．その後，登録したファクト適切な順番で呼び出すために，nth_clause と clause を用いた．そして，呼び出したファクトに対して Head_1 と Body が実行されると，M を用いた Q_S の canonical rewriting が求まる．その後，文献 [3] のアルゴリズムのポリシの候補をさらに絞り込むための処理が make_list や check により行われ，toim の第 8 引数と第 9 引数に格納される．

3.2 ステップ 2

ステップ 2 では，ステップ 1 で求めた $\mathcal{Q}$ の各要素 Q について，それが $\mathcal{Q}$ の中で極大の情報を与えるかをチェックする．具体的には，Q とは異なる任意の $Q' \in \mathcal{Q}$ について，「Q' を使用した Q の CQ-rewriting が存在するならば，Q を使用した Q' の CQ-rewriting が存在する」かどうかを，2.3 節の技法を用いてチェックする．チェックに通ったものが適切なポリシとして出力される．ステップ 2 に関しても Prolog を用いて，ステップ 1 の toim のような M の左辺で問い合わせるプログラムで実装できると予想している．ただし，ステップ 2 でも変数を一時的に定数として扱う場面が 2 回存在する．そのためインタプリタ上での作業を減らすために，ファイルに書き出す以外の方法で実装するのが理想であると考えられる．

4 まとめ

本研究では，文献 [3] で提案されたアルゴリズムを Prolog を用いて実装した．Prolog の性質上，変数を一時的に定数に置き換える処理については，一度テキストファイルに書き出して文字列のように扱うことで実現した．また，現在のプログラムの処理速度を Prolog の述語を用いて計測したが，大幅に時間を要する部分は存在しなかった．ステップ 2 での変数と定数の変換をファイル操作で行った場合に処理時間を要する可能性はあるが，現実的な時間での出力は可能であると予想される．今後の課題としては等価判定のチェックをする部分の実装と，変数と定数の変換処理の洗練化が挙げられる．

参考文献

[1] Foto N. Afrati. Determinacy and query rewriting for conjunctive queries and views. *Theoretical Computer Science*, Vol. 412, pp. 1005–1021, 2011.

[2] Pablo Barceló. Logical foundations of relational data exchange. *SIGMOD Rec.*, Vol. 38, No. 1, pp. 49–58, 2009.

[3] Yasunori Ishihara. Toward appropriate data publishing in relational data exchange framework. In *Fourth Workshop on Software Foundations for Data Interoperability*, 2020.

[4] 山口流星. 公開ポリシを考慮したデータ交換フレームワークにおける問合せクラスの拡張. 南山大学理工学部機械電子制御工学科卒業論文, 2020.

[5] 福嶋啓二, 石原靖哲, 藤原融. データ交換フレームワークにおける問合せ解像度に基づいたデータ公開. 電子情報通信学会技術研究報告, SS2018-44, pp. 103–108, 2019.