双方向変換フレームワーク BIRDS における データのアクセス権限を指定する制約のひな型の提案

M2022SC008 中尾瑠以
指導教員：石原靖哲

1　はじめに

近年，複数の組織が協調して研究開発を行ったり，サービスを運用する活動が盛んになっている．そのような協調活動において，参加組織それぞれが所有するデータの間の整合性を保つことが重要である．それと同時に，共有先の組織に対してアクセス権限（データのどの部分に読み出しや書き込みが可能か）を指定できることが望ましい．実際，PostgreSQL をはじめとして DBMS には通常アクセス権限指定機能がある．

データ共有のための技術として，双方向変換と呼ばれる，2 つの情報源（ソースとターゲット）の間の整合性を保つことが可能な仕組みが注目されている．BIRDS [1] と呼ばれる関係データの双方向変換フレームワークでは，ソースやターゲットに対する制約を指定することができる．

BIRDS では，アクセス権限を指定する機能は陽には考えられておらず，制約として与える必要がある．BIRDS の開発者が提供しているライドシェアリングの例 [2] では，アクセス権限を制約として指定しており，その例の中で与えられた双方向変換が well-behavedness という望ましい性質を持つことを自動で検証できることが述べられている．ここで，アクセス権限を表す制約を取り除いた場合，自動検証に失敗することが容易に確認できる．つまり well-behavedness の自動検証のためには，アクセス権限を制約として与えることが必要不可欠である．しかし，一般に PostgreSQL 等が提供しているアクセス権限をどのように BIRDS の制約で書き表すか，という指針は存在しない．

そこで本研究では，PostgreSQL で利用可能なアクセス権限のうち，「行単位および列単位で INSERT, DELETE, UPDATE それぞれの許可や禁止」について，BIRDS におけるデータのアクセス権限をソースに対して指定する制約のひな型を与える．さらに，PostgreSQL が持つアクセス権限指定機能を用いてアクセス権限を指定した場合と，BIRDS におけるアクセス権限を指定した制約のひな型の比較を用いて評価を行う．

2　関連研究

双方向変換を用いたデータ共有システムとして BIRDS を用いた Dejima アーキテクチャに関する研究 [3] がある．Dejima アーキテクチャとは，双方向変換を利用した更新伝播を用いて，柔軟なデータ統合を可能とするアーキテクチャである．また，BIRDS や Dejima アーキテクチャの両方を用いていない双方向変換に関する研究 [4, 5] がある．これらの研究では，双方向変換を用いてデータの共有を行うが，アクセス権限の指定はしていない．本研究では，BIRDS を用いてアクセス権限を指定した状態でデータの共有が行える機能を実現した．

3　双方向変換

双方向変換とは，2 つの情報源（ソースとターゲット）の間に双方向の変換を与えることにより，情報源の整合性を保つ仕組みである．図 1 のように，ソース S のみ引数にとりターゲット T を返す get 関数と，ソース S とターゲット T を引数にとり対応するソースを返す put 関数で表される．ここで，以下に示す特性が満たされる場合，双方向変換は well-behaved であるという．

$$put(S, get(S)) = S \quad \text{(GETPUT 特性)}$$

$$get(put(S, T')) = T' \quad \text{(PUTGET 特性)}$$

GETPUT 特性は，ターゲットにおいてデータの更新が行われなかった場合，ソースにおいてもデータの更新が行われないことを表している．一方 PUTGET 特性は，ターゲットにある全ての情報を完全にソースに変換できることを表している．

4　BIRDS におけるデータ更新

BIRDS とは関係データの双方向変換フレームワークであり，双方向変換を用いることで 2 つの情報源の整合性を保つことができる．

本研究では BIRDS のケーススタディで用いられているライドシェアリングシステム [2] を用いる．本研究で用いたソーステーブルとターゲットテーブルを一部簡略化したものを図 2 に示す．さらに，簡略化したテーブルに合うように修正した put 関数の定義を下の (1), (2) に示す．put 関数の定義ではデータの挿入を指定する場合は「+」を用い，データの削除を指定する場合は「−」を用いる．

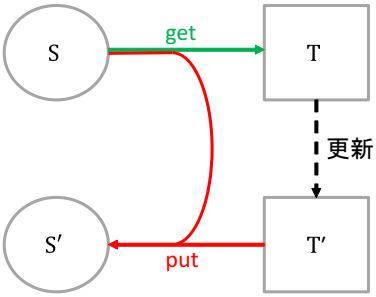

図 1　双方向変換

$$+vehicle(\mathrm{V},\mathrm{L},\mathrm{S}) \text{ :- } public(\mathrm{V},\mathrm{A},\mathrm{S}), area(\mathrm{L},\mathrm{A}), \neg vehicle(\mathrm{V},\mathrm{L},\mathrm{S}). \quad (1)$$

$$-vehicle(\mathrm{V},\mathrm{L},\mathrm{S}) \text{ :- } vehicle(\mathrm{V},\mathrm{L},\mathrm{S}), area(\mathrm{L},\mathrm{A}), \neg public(\mathrm{V},\mathrm{A},\mathrm{S}). \quad (2)$$

(1) は，(V,A,S) という行が *public* テーブルには存在し，かつ (L,A) という行が *area* テーブルには存在するが，(V,L,S) という行は *vehicle* テーブルには存在しない場合，(V,L,S) という行を *vehicle* テーブルに挿入することを表す．(2) は，(V,L,S) という行が *vehicle* テーブルに存在し，かつ (L,A) という行が *area* テーブルに存在するが，(V,A,S) という行は *public* テーブルには存在しない場合，(V,L,S) という行を *vehicle* テーブルから削除することを表す．

また，BIRDS ではソースやターゲットが満たすべき制約を指定することができる．BIRDS で指定できる制約の例を以下に示す．

$$\bot\text{:- } S(\mathrm{X}), \neg T(\mathrm{X}).$$

「⊥:-」は右辺が全て成り立つ場合はないという意味である．つまり，この制約は「ある X という行が S に存在し T に存在しない」という条件を満たす場合はない（すなわち，「S に存在する X という行は T にも存在する」）という意味の制約である．

BIRDS においてターゲットでデータの更新が行われた場合，INSERT 文と DELETE 文を組み合わせてソースに更新内容を反映させる．一方，関係データベースシステムで用いられている SQL 言語では，INSERT 文と DELETE 文に加えてデータの更新を命令する UPDATE 文が存在する．そして通常では，関係データの許される変化の仕方を INSERT, DELETE, UPDATE それぞれの許可，禁止の組み合わせ (アクセス権限) で指定する．しかし，BIRDS におけるアクセス権限を考える場合，UPDATE が INSERT と DELETE の組み合わせで表現されるため，UPDATE を禁止または許可する制約も INSERT と DELETE を組み合わせた制約で表現する必要がある．

5 BIRDS におけるアクセス権限を表す制約のひな型

本節では，PostgreSQL で利用可能なアクセス権限のうち，「行単位および列単位で INSERT，DELETE，UPDATE それぞれの許可や禁止」について，BIRDS におけるソースに対する制約のひな型を与える．

5.1 行単位のアクセス権限

5.1.1 禁止する場合

行単位の UPDATE のみ禁止するアクセス権限は，以下の制約によって表現することができる．

- 「データの挿入と削除の両方を行う」場合はない

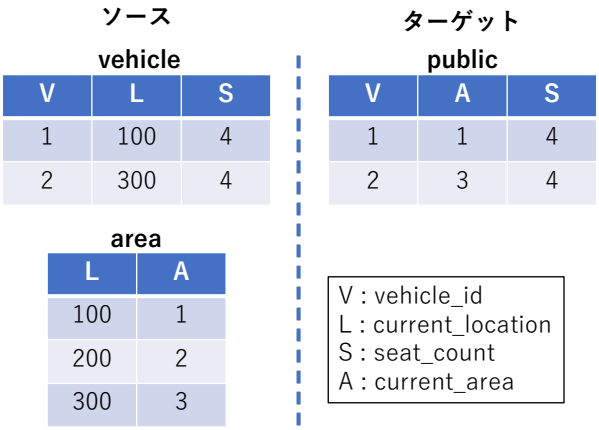

図 2　テーブルの概要

データの挿入と削除の両方を行う場合を禁止することにより UPDATE を禁止し，データの挿入のみ行う INSERT と削除のみ行う DELETE は許すことができる．これにより，UPDATE のみ禁止する制約を表現することができる．この制約を BIRDS で表現すると以下のようになる．なお，「_」は無名変数を表す．

$$\bot\text{:- } +vehicle(\_,\_,\_), -vehicle(\_,\_,\_).$$

同様に，行単位の INSERT のみ禁止するアクセス権限と DELETE のみ禁止するアクセス権限は，それぞれ以下の制約によって表現することができる．

- 「データの挿入のみ行い，削除は行なわない」場合はない
- 「データの削除のみ行い，挿入は行なわない」場合はない

これらの制約を BIRDS で表現するとそれぞれ (3),(4) のようになる．

$$\bot\text{:- } +vehicle(\_,\_,\_), \neg -vehicle(\_,\_,\_). \quad (3)$$

$$\bot\text{:- } \neg +vehicle(\_,\_,\_), -vehicle(\_,\_,\_). \quad (4)$$

5.1.2 許可する場合

行単位の INSERT，DELETE，UPDATE それぞれを許可するアクセス権限は，それぞれを禁止するアクセス権限を用いることで指定することができる．

まず，UPDATE のみ許すアクセス権限を考える．これは INSERT と DELETE を禁止することで指定できる．つまり，INSERT を禁止する場合と DELETE を禁止する場合を用いることで UPDATE のみ許す制約を生成することができる．また，この制約を BIRDS で表現すると以下のようになる．

$$\bot\text{:- } +vehicle(\_,\_,\_), \neg -vehicle(\_,\_,\_).$$
$$\bot\text{:- } \neg +vehicle(\_,\_,\_), -vehicle(\_,\_,\_,).$$

INSERT のみ許す場合は，UPDATE のみ許す場合と同様に以下のようになる．

$$\bot\text{:- } -vehicle(\_, \_, \_).$$

DELETE のみ許す場合は，UPDATE のみ許す場合と同様に以下のようになる．

$$\bot\text{:- } +vehicle(\_, \_, \_).$$

5.2 列単位のアクセス権限

BIRDS で扱うことが可能な列単位のアクセス権限として，INSERT と UPDATE それぞれを禁止する制約のひな型を与える．以下では，アクセス権限を指定したい列名を C とする．

5.2.1 NULL 値とデフォルト値

SQL において，列 C に対する INSERT や UPDATE を禁止した場合，列 C 以外に対して値を与えた（列 C を空のデータにした）INSERT や，列 C 以外の値を更新する UPDATE は許され， それ以外の INSERT や UPDATE は禁止される．また，SQL では空のデータを挿入した場合，NULL 値としてデータが挿入される．しかし，BIRDS では NULL 値のデータを扱うことを想定していないため，空のデータを挿入した場合エラーが発生し INSERT や UPDATE の SQL 文が実行できなくなる．

そこで本研究では，ソースとターゲットそれぞれのテーブル作成時に，そのデータのアプリケーションにおいて使われる可能性がない値をひとつ，NULL 値を表す値として選択する．そして，その値をデフォルト値として定義することで，実質的に空のデータを挿入することを可能にする．

BIRDS では SQL ファイルを用いて PostgreSQL にターゲットテーブルを定義するため，PostgreSQL 上で ALTER VIEW コマンドを用いてビュー定義を変更する必要がある．以下に本研究で用いた ALTER VIEW コマンドの例を示す．座席数を表す列 seat_count のデフォルト値を「−1」と定義している．

```
ALTER VIEW public ALTER seat_count
SET DEFAULT -1;
```

5.2.2 禁止する場合

列単位の UPDATE のみ禁止するアクセス権限は，以下の制約によって表現することができる．

- 「列 C の値が $S1$ である行の挿入と列 C の値が $S2$ である行の削除が同時に行われるが $S1$ と $S2$ は同一ではない」場合はない

列 C の値が $S1$ である行の挿入と列 C の値が $S2$ である行の削除が同時に実行され，$S1$ と $S2$ が同一ではない場合，列 C の値が $S1$ から $S2$ に更新されていることになる．そこで，これを禁止することで列 C に対して UPDATE のみ禁止することができる．この制約を BIRDS で表現すると以下のようになる．

$$\bot\text{:- } +vehicle(\_, \_, S1), -vehicle(\_, \_, S2), \neg S1 = S2. \tag{5}$$

列単位の INSERT のみ禁止する場合の制約は，以下の制約によって表現することができる．

- 「列 C の値が S である行の挿入を行うが S がデフォルト値ではない」場合はない

列 C の値が S である行の挿入が行われたとき，S がソーステーブルで定義されているデフォルト値ではない場合を禁止することで，列 C に対して値を与えた行の挿入が行われることを禁止することができるため，列 C に対して INSERT のみ禁止することができる．デフォルト値が -1 のときにこの制約を BIRDS で表現すると以下のようになる．

$$\bot\text{:- } +vehicle(\_, \_, S), \neg S = -1. \tag{6}$$

6 制約のひな型を用いたアクセス権限指定手法の評価

6.1 BIRDS のケーススタディにおける等価性の証明

ライドシェアリングにおいてアクセス権限指定の目的で与えられている制約を，本研究で提案したひな型に基づく制約で等価に置き換えできることを証明した．行った証明の一部を以下に示す．

ライドシェアリングで用いられている制約を以下に示す．

$$all\_id(\mathrm{V})\text{ :- } vehicle(\mathrm{V}, \mathrm{L}, \_), area(\mathrm{L}, \_). \tag{7}$$

$$\bot\text{:- } all\_id(\mathrm{V}), \neg\ public(\mathrm{V}, \_, \_). \tag{8}$$

$$\bot\text{:- } public(\mathrm{V}, \_, \_), \neg\ all\_id(\mathrm{V}). \tag{9}$$

これは (8) と (9) が制約であり，(7) は (8) と (9) の制約を記述するためのテーブルを別途定義した式である．(8), (9) の制約と等価な，本研究で提案したひな型に基づく制約は，(4), (5), (6) である．等価性を証明するには，以下の 2 点を示せばよい．

- (8) または (9) の制約に違反するならば，(4)，(5)，(6) のいずれかの制約に違反する．
- (4)，(5)，(6) のいずれかの制約に違反するならば，(8) または (9) の制約に違反する．

(8) の制約に違反するならば，(4)，(5)，(6) のいずれかの制約に違反する証明を行う．まず，(8) の右辺を (7) を用いて展開する．

$$\bot\text{:- } vehicle(\mathrm{V}, \mathrm{L}, \_), area(\mathrm{L}, \_), \neg public(\mathrm{V}, \_, \_).$$

さらに (2) を用いて変形する．

$$\bot\text{:- } -vehicle(\mathrm{V}, \mathrm{L}, \_)$$

これは制約 (4) そのものである．したがって，(8) の制約に違反するならば，(4)，(5)，(6) のいずれかの制約に違反する．

同様に，以下に示す 4 通りの証明も行った．

- (9) の制約に違反するならば，(4)，(5)，(6) のいずれかの制約に違反する．
- (4) の制約に違反するならば，(8) か (9) の制約に違反する．
- (5) の制約に違反するならば，(8) か (9) の制約に違反する．
- (6) の制約に違反するならば，(8) か (9) の制約に違反する．

その結果，すべてにおいて証明することができたため，ライドシェアリングの制約はひな型に基づく制約で表現することが可能である．

また，ライドシェアリングでは主キー制約が指定されていることを前提として考えられている．しかし，本研究で与えたひな型では，主キー制約の有無に関わらずアクセス権限を指定する制約を与えることが可能である．さらに，ライドシェアリングでは制約を与えるために別途テーブルを定義する必要があるが，ひな型では必要ないためシンプルに制約を与えることが可能である．

6.2 PostgreSQL が持つアクセス権限指定機能との比較

制約のひな型を用いたアクセス権限指定手法の評価を行うにあたり，PostgreSQL に備わっているアクセス権限を指定する機能である GRANT，REVOKE を用いて比較を行う．本研究では，ひな型と同じ条件にするために，GRANT コマンドの設定可能な権限の一つである ALL PRIVILEGES を用いて，利用可能な権限を全て付与した後，REVOKE コマンドを用いて権限を取り上げることで実現を行った．また，比較は長所と短所を用いて行う．

- ひな型の長所
  - アクセス権限を含めて GETPUT，PUTGET を満たすかを BIRDS 上で検証することができる．
  - 管理者権限を持つスーパーユーザに対して権限の指定を行える．
- ひな型の短所
  - データベースレベルで権限の指定をしていないため，更新戦略や制約が想定しない動きをしてしまい，データを壊す可能性がある．

ひな型では，BIRDS の制約としてアクセス権限を指定するため，アクセス権限も考慮した状態で双方向変換が well-behaved であるか検証できる．しかし，PostgreSQL が持つアクセス権限指定機能を用いて，データベースに対して権限の指定をした場合，その検証がうまく働くとは限らない．例えば，アクセス権限を考慮すれば well-behaved であるのに検証に成功しない場合や，逆に検証が成功してもアクセス権限を考慮すると well-behaved でない場合が存在する．

これらの結果から，権限を変更しつつ運用する場合は，権限を考慮した well-behavedness の検証ができるよう，ひな型を用いてアクセス権限を指定することが有用であると考えられる．逆に，権限を固定して運用する場合は，双方向変換が想定外の動きをした場合に備えて，GRANT や REVOKE を用いて権限を指定するのがよいと考えられる．

7 おわりに

本研究では，BIRDS で扱うことが可能な行単位および列単位で INSERT，DELETE，UPDATE それぞれの許可，禁止をするアクセス権限を指定する制約のひな型を与えた．さらに，PostgreSQL が持つアクセス権限指定機能である GRANT，REVOKE コマンドを用いて，ひな型との比較をした．その結果，アクセス権限を BIRDS の制約として与えることで，アクセス権限を含めた双方向変換が well-behaved であることの検証を行えるなどのメリットが得られることが分かった．

今後の課題としては，BIRDS における well-behavedness の静的解析機能を改変し，制約として与えたアクセス権限に違反するようなデータの更新が起きうるかを静的解析する機能を実現することが挙げられる．

参考文献

[1] Van-Dang Tran, Hiroyuki Kato, and Zhenjiang Hu. Programmable view update strategies on relations. *46th International Conference on Very Large Data Bases*, Vol. 13, No. 5, pp. 726–739, 2020.

[2] Van-Dang Tran, Hiroyuki Kato, and Zhenjiang Hu. BIRDS/Examples on a ride-sharing schema. `https://github.com/dangtv/BIRDS/tree/master/examples/ride_sharing`.

[3] Yasuhito Asano, et al. Bidirectional collaborative frameworks for decentralized data management. In George Fletcher, Keisuke Nakano, and Yuya Sasaki, editors, *Software Foundations for Data Interoperability*, Vol. 1457 of *Communications in Computer and Information Science*, pp. 13–51. Springer, 2022.

[4] Aaron Bohannon, Benjamin C. Pierce, and Jeffrey A. Vaughan. Relational lenses: A language for updatable views. In *Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 338–347, 2006.

[5] Grigoris Karvounarakis, Todd J. Green, Zachary G. Ives, and Val Tannen. Collaborative data sharing via update exchange and provenance. *ACM Transactions on Database Systems.*, Vol. 38, No. 3, 2013.