科学メタデータクリーニングのための自律分散型データ共有・更新システムに向けて

清水　敏之† 　加藤　弘之†† 　吉川　正俊†

† 京都大学大学院情報学研究科　〒606–8501 京都府京都市左京区吉田本町
†† 国立情報学研究所アーキテクチャ科学研究系　〒101–8430 東京都千代田区一ツ橋 2-1-2
E-mail: †{tshimizu,yoshikawa}@i.kyoto-u.ac.jp, ††kato@nii.ac.jp

あらまし　現在，膨大なデータが分散的に生成されており，もはや単一の利用者はデータ全体を把握できない場合が多い．そのような場合でもシステムの多くの参加者（データの管理主体に相当する．以後，ピアと呼ぶ）が自律分散的にデータを管理し，局所的な信頼関係に基づきデータを交換したいという要求は強い．このような要求を持つデータとして科学メタデータがあり，関連する分野のピア間でメタデータの共有が行われている．本論文では，ピア間で科学メタデータを共有する状況を想定し，データクリーニングによる値の修正を適切に扱うために，各ピアが隣接するピアと交換するデータを厳密に定義できる自律分散型データ共有・更新システムを設計する．各ピアが自身のデータのどの部分をどの隣接ピアに輸出するか，また隣接ピアからの輸入データのうちどの部分を自身のデータベースに取り入れるかを厳密に制御できる．輸出，輸入するデータの定義と更新の相互反映を実現するために，プログラミング言語の分野で用いられている双方向変換技術を応用する．

キーワード　双方向変換, P2P, データクリーニング

1 はじめに

現在，多種多様なデータが分散的に生成されている．様々な分野でデータの分析が盛んに行われているが，関連するデータの全容を把握することは困難な状況になっている．組織や事業によってデータが収集され，データベースによって管理されている場合があるが，類似するデータでも管理主体によって異なる形式で管理されている場合が多いと思われる．また，異なる管理主体（以降，ピアと呼ぶ）間でデータを共有することによって利便性の向上が期待できるが，関係するピアの全体像が不明確な場合は，まずは局所的な信頼関係に基づいて少数のピア間で自律分散的にデータ共有を行うことが有効であると考えられる．

本研究では科学メタデータの管理を想定して自律分散型データ共有・更新システムの設計を行う．専門性の高い科学データは，付随するメタデータも組織ごと，事業ごとに生成され管理されていることが一般的であり，一方で公開データに対するメタデータは広く周知されることが望まれるため共有したいという要求が強い．また，メタデータは人手によって記述される部分も多く，記述の誤りや値の不整合などが起こりえる．そのため共有にあたっては値の更新についても考慮することが望まれる．その際，各ピアが自身が保有していたメタデータの更新を行うだけでなく，データ共有による新たな気づきによって値の不整合などが明らかになり，元々は他のピアが保有していたメタデータも含めて更新を行いたい場合があると考えている．

本研究で設計する自律分散型データ共有・更新システムでは，各ピアが自らに取り入れた共有データに対してデータクリーニングを目的とした値の更新を行うことを想定する．更新された値は更新を行ったピアから他のピアに伝播されるが，科学メタデータのような専門性の高いデータでは値の更新の適切性判断が難しい場合があるため，各ピアがそれぞれのポリシーに基づいて更新を受け入れるかどうかを定義できるべきだと考えた．

提案するシステムでは共有テーブル ST(Shared Table) 上で provenance 情報を付与した共有データを管理し，各ピアは制御テーブル CT(Control Table) を介してデータの輸出および輸入を制御する．各ピアの基テーブル BT(Base Table) と CT 間，および CT と ST 間の相互反映には双方向変換 [1] を応用し，各ピアは双方向変換の変換定義を記述することでデータの輸出入のポリシーを表現する．

以降の本論文の構成は以下の通りである．2 節では本論文が想定する状況を具体化するための例について説明する．その後，3 節で本研究で提案するシステムの概要を述べる．関連研究については 4 節で紹介し，5 節で残された課題などについて議論を行い，最後に 6 節で本研究をまとめる．

2 例

本論文では科学データに対する記述されたメタデータの共有を想定して説明する．各ピアは自身が管理するデータに対するメタデータを保有しており，関連するデータを保有するピアとメタデータを共有したい．その際，各ピアはそれぞれ異なるスキーマでメタデータを管理している場合がある．

具体的な例として Collaborative Data Sharing System(CDSS) [2] で用いられている例を参考に用いて説明する．ピアとして $P_{GUS}, P_{uBio}, P_{BioSQL}$ の三つがあり，P_{GUS} はテーブ

G

id	can	nam
1	4	5
3	5	2

(a) P_{GUS}

U

nam	can
3	6

(b) P_{uBio}

B

id	nam
2	4

S

id	can
2	7

(c) P_{BioSQL}

図 1: 各ピアのスキーマおよび初期値

G

id	can	nam
1	4	5
3	5	2
$f(3,6)$	6	3
2	7	4

(a) P_{GUS}

U

nam	can
3	6
5	4
2	5
4	7

(b) P_{uBio}

B

id	nam
2	4
1	5
3	2
$f(3,6)$	3

S

id	can
2	7
1	4
3	5
$f(3,6)$	6

(c) P_{BioSQL}

図 2: 共有後の状態

ル G(id, can, nam)，P_{uBio} はテーブル U(can, nam)，P_{BioSQL} はテーブル B(id, nam) および S(id, can) によってデータを管理している．また，各ピアは図 1 の値を初期値として保有している状況を想定する．

これらのピアはスキーマの異なるテーブルでデータを管理しているが，属性名の対応によりデータの交換が可能であることを想定し，これら三つのピアでデータ共有が行われると図 2 の状態になる．図 2 では，各テーブルで点線より上部が元々そのピアが保有していたデータであり，点線より下部が他のピアから輸入したデータである．なお，f は nam と can の値を引数として id を得る関数である．

本研究ではこのようなデータ共有が行われた際に，各ピアが他のピアで行われた値の更新を自身のデータに反映するかどうか制御することを考える．各ピアが持ちうるポリシーとしては例えば以下のようなものが考えられる．

- 特定のタプルに対する更新を受け入れる（もしくは受け入れない）．例えば $id \geqq 5$ のタプルの更新は受け入れる，などの条件によりポリシーを規定することが考えられる．
- 特定のピアが行った更新を受け入れる（もしくは受け入れない）．例えば P_{GUS} は P_{BioSQL} が行った更新は受け入れるが P_{uBio} が行った更新は受け入れない，などの基準を規定することが考えられる．

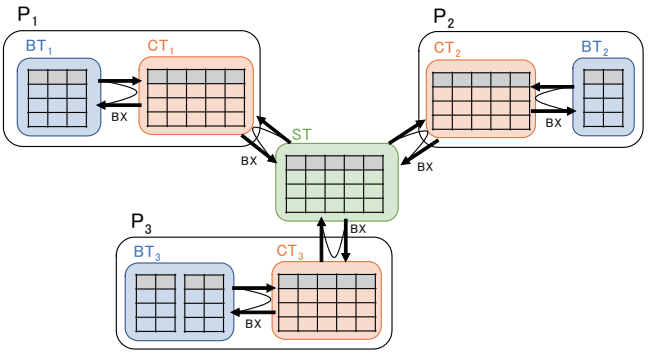

図 3: システム構成

3 システムの概要

ピア間でデータ共有を行い，かつ各ピアがそれぞれのポリシーを持って値の更新を制御するために，各ピアが元々保有している基テーブル BT(Base Table) に加えて制御テーブル CT(Control Table) および共有テーブル ST(Shared Table) を導入する．図 3 に三つのピアが参加する場合のシステム構成を示す．

BT と CT 間および CT と ST 間では双方向変換を応用することで相互反映を実現し，データログを用いて双方向変換の定義を記述することで各ピアの更新ポリシーを規定する [3]．BT と CT 間でデータの輸出を制御し，CT と ST 間でデータの輸入を制御する．ST には共有したいデータが格納される．本論文では ST にどの属性を保持して共有するかはあらかじめ参加するピアで合意して決定されていることを想定する．CT_i には ST に保持されているデータのうち，P_i で保持したいデータのみを取り入れる．また，CT_i は BT_i から共有データとして輸出したいデータを保持しているとみなすこともできる．

CT では ST で共有したいデータのスキーマに合わせ，各タプルにユニークに付与される tid 属性と，そのタプルがどこから来たか (origin) を示す org 属性を拡張して保持することを考えた．ST には共有データそのものに加え，各ピアが更新ポリシーの判断材料とするための情報を格納するための属性を持つ．本論文ではデータがどのピアに受け入れられているか (acceptance) を示す acc 属性および元々どのピアのデータだったか (origin) を示す org 属性を格納するものとした．acc だけでなく org を格納する理由は，複数のピアに輸入されたデータは acc だけでは org の情報を判断できないためである．また，データの削除により acc から org に相当する情報が消去される場合もある．図 2 の例において，提案するシステムを適用した場合，CT と ST は図 4 のようになる．

この時，例えば P_{GUS} において (1, 4, 5) のタプルが (1, 8, 5) に更新された場合，ST はそれに対応して図 5 のように更新される．図 5 の ST では，(1, 8, 5) のタプルは acc 属性の値が $\{g_1\}$ であるため P_{GUS} が (1, 8, 5) を保持していることが分かり，一方，(1, 4, 5) のタプルの acc 属性の値は $\{u_2, b_2\}$ であるため P_{uBio} と P_{BioSQL} は (1, 4, 5) を保持していることが分

CT_{GUS}

id	can	nam	tid	org
1	4	5	g_1	$\{g_1\}$
3	5	2	g_2	$\{g_2\}$
$f(3,6)$	6	3	g_3	$\{u_1\}$
2	7	4	g_4	$\{b_1\}$

CT_{uBio}

id	can	nam	tid	org
$f(3,6)$	6	3	u_1	$\{u_1\}$
1	4	5	u_2	$\{g_1\}$
3	5	2	u_3	$\{g_2\}$
2	7	4	u_4	$\{b_1\}$

CT_{BioSQL}

id	can	nam	tid	org
2	7	4	b_1	$\{b_1\}$
1	4	5	b_2	$\{g_1\}$
3	5	2	b_3	$\{g_2\}$
$f(3,6)$	6	3	b_4	$\{u_1\}$

ST

id	can	nam	acc	org
1	4	5	$\{g_1, u_2, b_2\}$	$\{g_1\}$
3	5	2	$\{g_2, u_3, b_3\}$	$\{g_2\}$
$f(3,6)$	6	3	$\{u_1, g_3, b_4\}$	$\{u_1\}$
2	7	4	$\{b_1, g_4, u_4\}$	$\{b_1\}$

図 4: CT および ST

ST

id	can	nam	acc	org
1	4	5	$\{u_2, b_2\}$	$\{g_1\}$
3	5	2	$\{g_2, u_3, b_3\}$	$\{g_2\}$
$f(3,6)$	6	3	$\{u_1, g_3, b_4\}$	$\{u_1\}$
2	7	4	$\{b_1, g_4, u_4\}$	$\{b_1\}$
1	8	5	$\{g_1\}$	$\{g_1\}$

図 5: 更新された ST

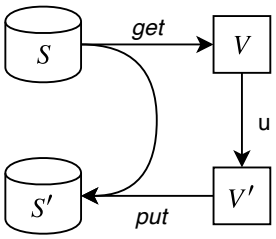

図 6: 双方向変換

かる．このような更新情報は各ピアが記述する更新ポリシーに従って各ピアへの反映が行われ，さらにそれに従って ST の更新が行われる．つまり，P_{uBio} と P_{BioSQL} は P_{GUS} が $(1, 8, 5)$ を保持していることを踏まえ，自身の更新ポリシーに従ってこの更新を受け入れるかどうかを判断する．各ピアは ST における acc および org を用いつつ更新ポリシーを反映した双方向変換を定義するが，具体的なデータログでの記述方法は現在の検討課題である．

4 関 連 研 究

4.1 協調型データ共有システム

複数のピアが協調してデータを共有する手法に，Collaborative Data Sharing System(CDSS) [2] がある．CDSS は，データ交換 (data exchange) に基づきデータを共有し，データの制御は local rejections と local contributions を定義することで実現している．CDSS では，ピアが更新できるのは，自分自身のピアで所有しているデータだけであり，他のピアが所有するデータを輸入する際に，local rejections によって反映したくないデータを定義し，他のピアにデータを輸出する際に，local contributions によって輸出するデータを定義している．これに対して，本論文で提案する枠組みでは，他のピアが所有するデータの，共有だけでなく更新も実現する．他のピアが所有するデータの更新は，データクリーニングにおいて必要な機能である．

4.2 双方向変換

双方向変換 (bidirectional transformation, bx) [1] とは，ソースデータをターゲットデータに変換した後，ターゲットデータ上の更新をソースデータに反映させることが可能な計算の枠組みのことであり，プログラミング言語の分野で研究されている技術で，データベースのビュー更新問題を解決する手法としても注目されている [4], [5].

双方向変換は，図 6 に示すように，順方向変換 *get* と逆方向変換 *put* の対で構成されている．順方向変換 *get* は，ビュー定義であり，ソースデータベース S に対する問合せで，ビュー V を結果として返すものである．その一方で，逆方向変換 *put* は，ビューへの更新をソースデータに反映させるもので，元のソースデータベース S と，u によって更新されたビュー V' の二つを引数として，更新されたソース S' を返すものである．

ソースデータベースとビューの整合性を保証するための主な性質に，GetPut と PutGet がある．[1]

$$\forall\, S, \qquad put\,(S,\ get(S)) = S \qquad \text{(GetPut)}$$

$$\forall\, S,\ V', \qquad get\left(put\left(S,\ V'\right)\right) = V' \qquad \text{(PutGet)}$$

(GetPut) は，ビューが更新されていなければソースは更新されないことを保証している．その一方で，(PutGet) は，ビューに対する更新の全てはソースに反映されるので，更新されたソースに *get* を適用すると更新されたビューを得ることができることを保証している．

双方向変換の構成要素である *get* と *put* を構築するには二つのアプローチがある．一つは，ビュー定義 (*get*) から *put* を導

1：双方向変換の性質には他にも様々な種類が議論されている [6].

出する方法 (get_based bx) と，もう一つは，*put* から *get* を導出する方法 (put_based bx) である．ビュー定義は，直感的で書きやすい反面，一般にビュー定義は単射 (injective) でないため，get_based bx では 複数の *put* が候補となりうる．このことが，データベースにおけるビュー更新問題を難しくしている．その一方で，put_based bx では，well_defined な *put* から高々一つの *get* を導出できることが示されている [6] ので，*put* さえ記述すれば良いが，well-defined な *put* の定義が *get* に比べて難しい．最近，この *put* の書きにくさを，データログを用いることで解決する枠組みが提案されている [3]．本稿では，[3] での手法に基づき，データログを使って変換を定義することを検討している．但し，[3] では整合性を保つために上記 (GetPut) と (PutGet) を満たすデータログを用いているのに対して，本研究では，共有テーブルと制御テーブル間の同期を取るためにデータベーストランザクションを用いることを検討している．そのためにどのようなトランザクションを構成するかについては，今後の課題である．

5 議　　論

本論文では各ピアが持つデータの輸出入に関するポリシーについて例を示したが，データクリーニングに際して各ピアが持ちうる要求と，それをどのように双方向変換の変換定義として記述するかは今後より詳細に考察する予定である．その際，本研究で設計した基テーブル BT(Base Table)，制御テーブル CT(Control Table)，共有テーブル ST(Shared Table) を用いた構成によって対応できる要求の範囲についても合わせて考える必要があると考えている．更新ポリシーの記述は各ピアがそれぞれ行うことを想定しているが，更新ポリシーに対応する双方向変換の変換定義の記述は容易ではない場合もあると考えられるため，実用的な更新ポリシーのパターンを用意しておき，記述を補助することを考えている．

共有テーブル ST の実装方法についてはいくつかの方向性が考えられるため，処理効率も考慮しつつ，どのように実装するかを考察したい．共有に参加しているピア以外の cloud サービスなどの第三者が格納することや，各ピアがそれぞれ同期された ST を保持することなどが考えられる．さらに，ST を仮想化することが可能であれば実体を保存する必要がなくなるため，仮想化の可能性についても検討したい．

また，あるピアが複数の共有に参加している場合のトランザクション処理については一考の余地がある．ロックの仕組みがどのようにあるべきかや，更新が伝播される中，ピアのつながりにサイクルがある場合でも問題なく処理が停止するかといった点についての議論は課題となっている．

6 おわりに

本論文では，科学メタデータの共有においてデータクリーニングを行うことを想定し，自律分散型データ共有・更新システムの検討を行った．データクリーニングのための更新として，各ピアは元々は他のピアが所有していたデータであっても共有データであれば各ピア中で更新可能であると考え，更新情報は各ピアが定義する共有と更新に関するポリシーに従って反映される枠組みを考案した．

謝辞　本研究の一部は JSPS 科研費 JP17H06099, JP18H04093 の助成を受けたものです．

文　　献

[1] Faris Abou-Saleh, James Cheney, Jeremy Gibbons, James McKinna, and Perdita Stevens. Introduction to bidirectional transformations. 9715:1–28, 2018.

[2] Grigoris Karvounarakis, Todd J. Green, Zachary G. Ives, and Va l Tannen. Collaborative data sharing via update exchange and provenance. *ACM Transactions on Database Systems*, 38(3):19:1–19:42, 2013.

[3] Van-Dang Tran, Hiroyuki Kato, and Zhenjiang Hu. Programmable view update strategies on relations. *PVLDB*, 13(5):726–739, 2020.

[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*, pages 338–347, 2006.

[5] 加藤 弘之, 胡 振江, 日高 宗一郎, and 松田 一孝. 『ソフトウェアサイエンスの基本』シリーズ第 4 回高談闊論：双方向変換の原理と実践. **コンピュータ ソフトウェア**, 31(2):44–56, 2014.

[6] Sebastian Fischer, Zhenjiang Hu, and Hugo Pacheco. A clear picture of lens laws. In Ralf Hinze and Janis Voigtländer, editors, *Mathematics of Program Construction*, pages 215–223, Cham, 2015. Springer International Publishing.