社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－DBS－138（20）
2006－GN－58（20）
2006／1／27

P2P データ流通における注釈を用いたトレーサビリティの実現手法の検討

石川 佳治[†,††]，　張 建偉[†]，　永元 芳幸[§]，　北川 博之[†,††]
[†]筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻
[††]筑波大学計算科学研究センター
[§]筑波大学第三学群情報学類
{ishikawa,kitagawa}@cs.tsukuba.ac.jp, {zjw,nagamoto}@kde.cs.tsukuba.ac.jp

概要

P2P（Peer-to-Peer）ネットワークは，柔軟な情報の流通・交換のための基盤として近年大いに着目されている．本稿では，データベースのデータを P2P 環境上で交換する状況において，情報がどのようにピア（peer）間を流通したか，どのように修正がなされたかなどを事後に検証するためのトレーサビリティ確保の技術について議論する．データの流通に関するログ情報を，特殊なリレーションである注釈（annotation）の形で通常のリレーションに付与し，これを問合せ言語を用いて問合せ可能とすることで，流通経路などに関する柔軟かつ多様な問合せを可能とする．本稿では，注釈を用いたトレーサビリティの実現方式の提案を行い，その実現手法について議論する．

Enabling Trasability Based on Annotations in Data Exchange

Yoshiharu Ishikawa[†,††]，　Jianwei Zhang[†]，　Yoshiyuki Nagamoto[§]，
Hiroyuki Kitagawa[†,††]
[†]Department of Computer Science,
Graduate School of Systems and Information Engineering,
University of Tsukuba
[††]Center for Computational Sciences, University of Tsukuba
[§]College of Information Sciences, Third Cluster of Colleges, University of Tsukuba

Abstract

Recently, P2P (Peer-to-Peer) networks have emerged as attractive infrastructure to achieve flexible information delivery and exchange. This paper discuss how to develop traceability facilities that enable to examine how data was transfered between peers and how data modification was performed in peers. We attach log information about data transfer to relational tuples as annotations; annotations are represented as special types of relations. It makes possible to query annotations using a query language for tracing data lineage in a flexible manner. This paper also proposes some implementation approaches of annotation-based traceability facilities.

1 はじめに

インターネットの発達につれ，多数の自律的なコンピュータの間でのデータ交換・情報流通が盛んになっている．とりわけ，近年着目されている P2P（Peer-to-Peer）コンピューティング [1] においては，不特定多数のピア（peer）が自律的に連携して情報を交換する形態が広くみられる．この種の情報流通においては，ピア間でデータの複製や加工が繰り返されるため，あるピアが現在保持しているデータが，どのピアにより作成され，どのような経過をたどって得られたかが明確でなくなる．また，逆に，あるピアが提供したデータが，ネットワークを介して現在どのピアに渡っているかということを追跡することも容易ではなくなる．これらの結果，ネットワークを介して取得されたデータに対し十分な信頼がおけなくなり，情報流通において生み出されるメリットを必ずしも享受できなくなるという問題が発生する．

このような背景から，本稿では，特に P2P コンピューティングのような，自律的なコンピュータが協調して情報流通・交換を行う環境を想定して，流通するデータの**トレーサビリティ**を実現するための枠組みを提案する．本研究では，データの交換の基盤として，リレーショナルデータベースを想定し，ピア間での問合せの発行によりデータの交換が行われるものと想定する．問合せの結果，ピア間を流通するタプルには，データの作成者や仲介者などの情報を記録するメタデータである**注釈**（annotation）が付与される．注釈はタプルに付随する情報であり，通常のデータベース問合せ処理においては明示的には扱われないが，データの出所や流通経路などを意識する場合には，それらに対する条件を問合せ中に明示的に含めて表現する．このような機能により，たとえば「このデータを作成したピアを答えよ」,「このデータはどのような経路をたどって得られたか」,「このデータの複製を保持するピアを探索せよ」などの，トレーサビリティの基本となる処理の実現を図る．

以下では，リレーショナルモデルに注釈を付与するデータモデルについて述べ，次いで，トレーサビリティの基礎となる基本演算について説明する．さらに，P2P 環境でそのような注釈機能を実現するための実現手法について検討を行う．2 節で関連研究について触れ，3 節で注釈情報の概念について基本的なアイデアを説明する．4 節ではその実現方式に関する検討を行い，5 節では 4 節の実装方式に基づく別の観点からの注釈情報の利用について述べる．最後に 6 節でまとめと今後の課題を述べる．

2 関連研究

P2P 環境のみならず，加工や複製を伴い複数データベース間を流通するデータや，基幹データベースとデータウェアハウスのデータの相互の関係を把握することは，重要な課題となっている．加工や複製の結果得られたデータについて，その出所（provenance）を把握し，データの信頼性を確保することは，*data provenance* あるいは *lineage tracing* などと呼ばれる [14]．本研究ではこれをデータの**系統管理**と呼ぶことにする．

系統管理が着目されている分野の 1 つとしてデータウェアハウスがある．データウェアハウス中のデータがどのような経緯を経て基幹データベースから導かれたかを知ることは分析の質や信頼性を高める上で重要である．また，ネットワーク上に分散している科学分野のデータベースの連携や情報交換などにおいても，情報の出所の管理が求められている [3, 4]．プロトタイプシステムとしては，必ずしもデータウェアハウスに限らないが，[3, 6, 16] などがある．

系統管理には，必要なときにはじめてデータの出所を求める**遅延処理**（lazy processing）のアプローチ [4, 7, 8, 17] と，データに出所に関する情報を付随させる**先行処理**（eager processing）のアプローチ [2, 3, 5, 11, 13, 15] に分類できる．付随させる情報は，状況に応じて，メタデータ，注釈（annotation），source tagging, attribution などと呼ばれる．

また，データの系統管理には,「そのデータがなぜ存在するか」という観点と「そのデータがどこから来たか」という観点が存在する．前者は **why-provenance** [7, 17]，後者は **where-provenance** [4] などと区別される．

P2P 環境におけるデータベースに関してはさまざまな研究がなされているが [9, 10, 12]，異種のスキーマのマッピングや P2P ネットワーク上の問合せ処理に関する研究が主であり，本研究で述べるような情報の流通経路を追跡する研究に関しては，著者の知る限り研究は存在しない．

3 基本的なアイデア

3.1 データモデル

例として，複数のピアが含まれる P2P ネットワークを考える．そこでは音楽の曲目に関する情報が交換されると考える．各ピアは図 1 のようなスキーマのデータベースをそれぞれローカルに持つものとする．異種分散情報の連携においては，各ピアが有するデータベースが同一のスキーマを有し，また，各リレーションや属性の意味が同一であるとは限らないが，ここではラッ

Album

aid	artist	title	...
1 $\{a_{11}\}$	The Beatles $\{a_{12}\}$	Rubber Soul $\{a_{13}\}$	...
2 $\{a_{21}\}$	Eric Clapton $\{a_{22}\}$	Unplugged $\{a_{23}\}$	...
3 $\{a_{31}\}$	Eric Clapton $\{a_{32}\}$	One More Car, ... $\{a_{33}\}$	...
...	...	...	...

Song

name	album	...
Drive My Car $\langle s_{11} \rangle$	1 $\langle s_{12} \rangle$	...
Nowhere Man $\langle s_{21} \rangle$	1 $\langle s_{22} \rangle$	...
Layla $\langle s_{31} \rangle$	2 $\langle s_{32} \rangle$	...
Layla $\langle s_{41} \rangle$	3 $\langle s_{42} \rangle$	...
...	...	...

図 1: データベースの例

パー（wrapper）などの機構により，このような異種性はすでに吸収されているものとする．

各属性値に対し $\langle a_{11} \rangle$ などの形で付与されている情報を**注釈**（annotation）と呼ぶ．注釈は，それを明示的に指定しない限り問合せ結果には現れない．たとえば，次の問合せ Q1 は，この例については図 2 の結果を返す．

```
Q1:
SELECT DISTINCT a.title, s.name
FROM Album a, Song s
WHERE a.aid = s.album AND a.artist = 'The Beatles'
```

title	name
Rubber Soul $\langle a_{12} \rangle$	Drive My Car $\langle s_{11} \rangle$
Rubber Soul $\langle a_{12} \rangle$	Nowhere Man $\langle s_{21} \rangle$
...	...

図 2: 問合せ Q1 の結果

a_{12}, s_{11} などは，元のデータベース中の注釈が伝播したものである．ただし，この情報は問合せ結果においては，明示的に指定しない限りは実際にはユーザの目に触れることはない．一方，以下の問合せ Q2 については，図 3 のように，というように，注釈を無視した場合にちょうどリレーショナルモデルによる操作と等価となるように処理が行われるものとする．DISTINCT の影響で，1 つのタプルに値がまとめられた場合，対応する注釈は統合されるものとする．$\oplus$ は注釈を統合する演算である（後述）．

```
Q2:
SELECT DISTINCT s.name
FROM Album a, Song s
WHERE a.id = s.album AND a.artist = 'Eric Clapton'
```

name
Layla $\langle s_{31} \oplus s_{41} \rangle$
...

図 3: 問合せ Q2 の結果

なお，以降では簡単のため，SQL については簡単な SELECT-FROM-WHERE の構文のみを対象とし，SELECT 句には必ず DISTINCT を付与するものとする．すなわち，リレーショナルモデルにおける基本演算で実現できる処理を想定する．また,「リレーション」と「テーブル」の用語についても同じ意味として区別せず用いる．

このように注釈を付与し，問合せなどに伴い注釈を伝播させるアプローチは [3] に見られる．[3] では，特に注釈の伝播をユーザが制御できるようにすることに着目し，そのための SQL 拡張である問合せ言語 pSQL を提案している．pSQL では，どの注釈を伝播させるかなどをユーザが細かく指定することが可能である．本研究では，注釈情報をトレーサビリティのために利用するという目的から，ユーザ自身が伝播の具体的処理を制御することは想定しない*.

3.2 注釈リレーション

本節では，注釈をどのように利用するかについて述べる．本提案手法では，注釈自体が内部的にはタプル構造をもったテーブルとして仮想的に表現され，拡張した SQL 言語を用いて問合せ可能と想定する．注釈の構造を，以下のような形式の**注釈スキーマ**（annotation schema）により記述する．

```
log@Album(id, peer_id, timestamp, command, arg)
log@Song(id, peer_id, timestamp, command, arg)
```

このような記載により，リレーション Album, Song に含まれる各タプルの属性に log という名前のリレーションが附属するようになる．log@Album を，リレーション Album に附属する**注釈リレーション**（annotation relation）と呼ぶ．これは，それぞれの値に対してどのような変更が行われたかを追跡するためのログ情報を含む．

図 4 は図 1 の注釈 s_{31} の中に仮想的に含まれる log リレーションの内容の例を表す．この情報により，該当する属性値の変化を記録する．属性 id は注釈が付与されているタプルの ID を表す．本提案手法では，注釈を管理するために，各タプルについてはリレーション内部で一意な ID が付与されていることを想定する．peer_id はピアの ID を表す．timestamp はイベント

*注釈の伝播方式は，[3] における propagate all 方式に従うものとする．

のタイムスタンプを表す．command 属性と arg 属性の内容については後述する．

id	peer_id	timestamp	command	arg
#10	p1	05.10.19	created	t1
#10	p2	05.11.22	copied	p1
	...			

図 4: 注釈 s_{31} に含まれる log リレーションの例

ログ情報を扱う注釈リレーションは本提案手法において本質的であることから，各リレーションに対しこのようなリレーションが基本的に付与されるものとする．これ以外にも，要求に応じてユーザ定義の注釈スキーマを追加可能とするが，こちらについては省略する．

3.3 注釈情報に関する問合せ

本研究では，注釈情報を直接的にアクセスするための SQL の拡張を提案する．まず，以下に問合せ例を示す．

注釈情報の表示

```
Q3:
SELECT DISTINCT s.name, l.peer_id, l.timestamp,
                l.command, l.arg
FROM Album a, Song s, s.name@log l
WHERE a.aid = s.album AND a.artist = 'Eric Clapton'
```

s.name@log は，Song テーブルに対応する変数 s の name 属性の値に付随するログ情報をテーブルとみなすための略記である．s.name@log l により，変数 l にこのテーブルが束縛される．この問合せ Q3 は，「Eric Clapton のアルバムに入っている各曲の曲名（name）について，注釈情報を表示する」というものである．なお，この問合せはピア p2 において実行されたものとする．そのため，各ピアからピア p2 が取得したタプルの属性値に付与された注釈（p2 自体でなされた修正なども含む）が取得される．

name	peer_id	timestamp	command	arg
Layla $\langle s_{31} \oplus s_{81} \rangle$	p1	05.10.19	created	t1
Layla $\langle s_{31} \rangle$	p2	05.11.22	copied	p1
Layla $\langle s_{41} \rangle$	p3	05.10.21	created	t2
Layla $\langle s_{41} \rangle$	p2	06.1.5	copied	p3
Layla $\langle s_{41} \rangle$	p2	06.1.10	modified	layla
Layla $\langle s_{81} \rangle$	p4	05.12.01	copied	p1
Layla $\langle s_{81} \rangle$	p2	05.12.31	copied	p4
	...			

図 5: 問合せ Q3 の結果

問合せ Q3 の結果は図 5 のようになる．ここでも，$\langle \ldots \rangle$ で示された注釈は参考のため表示しているもので，これ自体はユーザには提示されないことに注意する．command, arg 属性のペアの解釈は以下のようになる．

- (created, tid)：該当するピアがその値を生成したことを意味する．tid は，リレーションへのタプル生成に応じて生成されるシーケンス値である．利用法については後述する．
- (copied, pid)：ピア ID pid のピアから値をコピーしたことを意味する．
- (modified, val)：val という文字列値から値を変更したことを意味する．

これらにより，図 5 に表示されている結果タプルのみに限っていえば，“Layla” という曲名のデータは，以下のような経緯で p2 に得られたことになる．

- ピア p1 で 05.10.19 に作成された．それが 05.11.22 にピア p2 によりコピーされた．
- また，p1 の同じデータが，05.12.01 にピア p4 によりコピーされ，それがさらに p2 により 05.12.31 にコピーされた．
- ピア p3 でも 05.10.21 に作成されたが，そのときの値は “layla” であった．それが 06.1.5 に p2 によりコピーされ，06.1.10 に p2 において “Layla” に修正された．

このようにして，現在手元にあるデータに関する注釈情報を表示することが可能となる．なお，この例では，ピア p1 にいて値が生成されたという情報は 2 つの注釈 s_{31}, s_{81} に重複して記録されている．よって，図 5 の 1 行目に見られるように，重複除去の結果，2 つの情報が統合され 1 タプルとなっている．

注釈情報の選択表示　別の例として，ピア p2 が他のピアでなされた処理のみを表示したい場合には，以下の問合せ Q4 を発行する．

```
Q4:
SELECT DISTINCT s.name, l.peer_id, l.timestamp,
                l.command, l.arg
FROM Album a, Song s, s.name@log l
WHERE a.aid = s.album AND a.artist = 'Eric Clapton'
      AND l.peer_id <> 'p2'
```

その結果は図 6 のようになる．

再帰的問合せの利用　SQL-99 において導入された再帰的問合せの機能を利用することで，得られた情報の経路を考慮した問合せを行うことも可能となる．たとえばピア p2 が，ピア p4 を経由してどのような情報が

name	peer_id	timestamp	command	arg
Layla $\langle s_{31} \oplus s_{81} \rangle$	p1	05.10.19	created	t1
Layla $\langle s_{41} \rangle$	p3	05.10.21	created	t2
Layla $\langle s_{81} \rangle$	p4	05.12.01	copied	p1
	...			

図 6: 問合せ Q4 の結果

得られたかを検証したいとする．以下の問合せ Q5 は，「p4 が介在して（すなわち p4 からのコピーの結果）ピア p2 の手元に得られた曲名を挙げよ」というものである．

```
Q5:
WITH RECURSIVE Trace(name, aid, pid)
AS (SELECT s.name, l.id, l.peer_id
    FROM Song s, s.name@log l
    WHERE l.command = 'copied' AND l.arg = 'p4'
    UNION ALL
    SELECT t.name, t.aid, l.peer_id
    FROM Song.name@log l, Trace t
    WHERE t.dest = l.arg AND t.aid = l.id)
SELECT DISTINCT name
FROM Trace
WHERE pid = 'p2'
```

問合せは 2 つの SQL 文からなる．1 番目の SQL 文では，Trace テーブルに p4 からコピーされたデータが，その後どのようにピア間を伝播したかの情報を再帰的に累積する．Trace の各タプルには，p4 を介して得られた曲の名前（name），注釈 ID（aid），情報を受け取ったピアの ID（pid）が含まれる．2 番目の SQL 文により，Trace テーブルの中から p2 に到着した曲名のみを選択する．

このような機能を用いることにより，手元に存在するデータに関して，その出所や流通経路に関する問合せを行うことが可能となる．

その他の問合せ　注釈情報を用いることで，上記以外にも他の問合せに応えることが可能となる．たとえば，以下のような問合せが考えられる．

- 複数の情報源の整合性のチェック：ある情報（たとえば先の例では曲名）が，単一の情報源において作成されたものであるか，複数の情報源において作成されたものであるかといったチェックが可能である．複数の情報源で同一の情報が作成されている場合，その情報の信頼性がより高いというような評価を行うことも可能となる．一方，複数の情報源からの情報に食い違いがあることを発見することも，場合によっては可能となると考えられる．
- 履歴情報を考慮した問合せ：イベントの発生時刻とともにログ情報が記録されていることにより，履歴情報に関するさまざまな問合せも可能となると考えられる．たとえば，情報がどの時点で修正されたか，ある時刻における注釈情報の内容，情報の生成源のピアから現在のピアまで情報が転送されるのに要した時間など，さまざまな問合せが可能となると考えられる．
- スナップショットの復元：図 5 における t1 などのように，本提案手法では，最初にタプル値が生成される際に，ログ情報には一意なタプル ID が付与されることを想定している．この情報を用いることで，手元にある情報が元々の時点ではどのような値とタプルを成していたかなどを原理的には追跡できることになる．

4　実現方式に関する検討

この節では，前節まで述べた注釈リレーションを実現するための方式について検討する．

4.1　方式 1：注釈情報の添付

第 1 のアプローチは，提案手法の考え方を直接的に反映したものである．各タプルが生成される際に対応する注釈が生成され，ピア間でデータのコピーが発生する際には，対応する注釈データもコピーされ既存の注釈データと統合される．

図 7 にその実現例を示す．基本的な構造は図 4 に示した log リレーションの概念レベルの内容と同様であるが，対応するリレーション名および属性名を識別するための属性として，rel および attr が新たに付与されている†．

rel	attr	id	peer_id	timestamp	command	arg
Song	name	#10	p1	05.10.19	created	t1
Song	name	#10	p2	05.11.22	copied	p1
Album	artist	#55	p8	06.01.15	created	t7
	...					

図 7: 注釈情報の実現例

このアプローチを採用した場合の注釈情報の管理は以下のようになる．

(a) 最初にタプルが作成された時点では，該当するピアにおいてタプル ID の付与がなされ，各属性について生成情報（created）の注釈が作成される．

(b) 他のピアからリレーションに対する問合せが出された際，問合せがなされたピアは，問合せ結果の

†このような実装である必要は必ずしもなく，id 属性に対応するリレーション・属性の情報を含めておいてもよい．

値に加えて注釈情報も併せて返却する．

(c) 問合せ結果を取得したピアは，問合せ結果を棄却するか，もしくは，既存のリレーションに追加する．棄却する場合には一切その後の配慮は不要である．一方，追加する場合には各タプルの属性に対してコピーされたという情報（`copied`）が注釈として付与される．

(d) タプルに修正が行われた際には，修正された内容が注釈（`modified`）として付与される．

これらを実現するには，データベースに対してなされた操作を記録する機能が必要となる．近年の商用データベースでは，セキュリティ対策の面から，データベース監査に役立つ機能として，データベースのデータの挿入・削除・更新および問合せを監視する機能が備わっている．そのため，(a) におけるデータの生成，(c) におけるデータの追加，(d) におけるデータの修正に対する注釈情報は，監視ログをもとにして構築できると考える．

問題となるのは，(b) における問合せと (c) におけるコピー（データの追加）である．各ピアにおけるデータベースのスキーマとその意味が完全に一致していた場合で，単にあるリレーションの一部のタプルをコピーして追加する場合については問題ないが，スキーマが異なりマッピングが必要となる場合，および，リレーショナルデータベースで許される変換処理を用いた問合せの結果をコピーする場合には，注釈情報の追跡が困難となりうる．

このような問題に関しては，すでに Widom らが中心となって，データウェアハウス環境におけるデータの変換の追跡を行う lineage tracing の手法を開発している [7, 8, 16]．本アプローチにおいても，このような研究で提案された追跡手法が活用可能であると考える．詳細の検討は今後の課題とする．

4.2 方式 2：注釈情報の分散管理

前節で述べた第 1 の方式には，データの複製に伴いすべての注釈情報がコピーされるという問題がある．同じデータを多数のピアがコピーした場合，注釈情報自体も複製されてしまうため，コピーの際のデータ転送や注釈の格納におけるオーバヘッドが発生する．そこで，もう 1 つのアプローチとして，各ピアが自分が所有するデータに関する注釈情報を管理し，問合せが発生した際には，ピア間の問合せ処理で注釈の追跡を図る方式を提案する．

この方式では，図 5 に示す問合せ `Q3` の結果に含まれる注釈情報を図 8 のように各ピアにおいて分散して管理する．基本的な考え方は次のようになる．

ピア `p1` におけるログ情報

rel	attr	id	timestamp	command	arg1	arg2
Song	name	#40	05.10.19	created	t1	
	...					

ピア `p2` におけるログ情報

rel	attr	id	timestamp	command	arg1	arg2
Song	name	#77	05.11.22	copied	p1	#40
Song	name	#102	06.1.5	copied	p3	#11
Song	name	#102	06.1.10	modified	layla	
Song	name	#99	05.12.31	copied	p4	#85
	...					

ピア `p3` におけるログ情報

rel	attr	id	timestamp	command	arg1	arg2
Song	name	#11	05.10.21	created	t2	
	...					

ピア `p4` におけるログ情報

rel	attr	id	timestamp	command	arg1	arg2
Song	name	#85	05.12.01	copied	p1	#40
	...					

図 8: 注釈情報の分散管理

(a) タプルを生成したという注釈情報は，その生成のピアにおいて管理する．この例においては，ピア `p1` とピア `p3` にそれぞれ 1 タプルずつの注釈が含まれる．たとえばピア `p1` の例の場合，1 行目の注釈情報は，タプル ID #40 のタプルが生成された際に，`Song` リレーションの `name` 属性に対して構築されたことを意味している．

(b) 値をコピーしたという情報は，コピーした側のピアにのみ保持する．たとえば図 8 のピア `p2` のリレーションの 1 行目は，ピア `p1` のタプル ID #40 の該当する値をコピーしたものが，`Song` リレーションのタプル ID #77 のタプルの `name` 属性の値となっているということを意味する．また，`p2` の 4 行目の行（#99）はピア `p4` が `p1` からコピーした値を再コピーすることでピア `p2` に得られているが，その元となるピア `p4` が `p1` から該当する値をコピーしたという情報はピア `p4` にのみ保持されている．

(c) データへの修正がなされた場合，その情報はそのピアのログにのみ格納される．ピア `p2` に対するリレーションの 3 行目がこれにあたる．

このような方式を採用する利点として，各ピアでログ情報を重複してもつ必要がなく，データ管理のオーバヘッドが削減できるという利点が存在する．各ピアが関連する部分に関してのみ情報を管理することにより，見通しの立てやすい方式となっている．

一方，この方式の欠点として，注釈情報を用いた分析処理を行う際にピア間での問合せ処理が発生すると

いう問題が挙げられる．例として，ピア p2 において問合せ Q3 を実行して，図 5 のような結果を得ることを考える．この際には次のような処理が行われることになる．

1. ピア p2 は手元のログ情報リレーションを参照し，ログ情報を抽出する．
2. 取得した情報をもとにログ情報を取得するための処理を各ピアに発行する．図 8 の例では，ピア p2 に対するリレーションの 1, 2, 4 行目から，ピア p1, p3, p4 が関連するログ情報を有していると分かることから，これらのピアに対応するログ情報の送付を依頼する．
3. 各ピアでは必要に応じて再帰的に問合せを転送し，最終的にログ情報を集約してピア p2 に返す．
4. ピア p2 では手持ちのログ情報と統合してユーザに結果を提示する．

このような処理は，方式 1 の管理手法に比べ，問合せ処理時のオーバヘッドが明らかに大きい．しかし，問合せの頻度自体が少ない場合には全体としてのコストを考えるとむしろ有効である場合も考えられる．管理コストとのトレードオフをとることが重要といえる．

5 「前向き」の問合せによる追跡

前節では，注釈リレーションを具体化するための実装方式として，注釈情報を添付し複製するアプローチと，分散管理するアプローチの 2 つを述べた．特に後者のアプローチは柔軟性が高く，これを拡張することで以下に述べるような問合せにも活用できる．

3 節で述べた問合せの例は，基本的にあるピアに（仮想的に）保持されている注釈情報に対して問合せを行い，その入手経路などを把握するものであった．しかし，注釈情報はこのような後ろ向きの問合せだけでなく，情報を発信した側（コピーされた側）から前向きに発行することも可能である．これは，あるピアが作成し公開したデータに誤りがあった場合などに，コピーしたピアにその旨を通知する場合などに利用できる．

具体的な例として，図 8 においてピア p1 が，自身が管理するログリレーションの 1 行目のエントリに対応する情報をどのピアがコピーしているかを知りたいとする．これは，ピア p1 における Song リレーションのタプル ID t1 のタプルの name 属性に含まれる値 “Layla” の値を追跡することに相当する．

このような前向きの追跡を行うには，あるピアにおいて提供された情報をどのピアがコピーしたかという情報を把握する必要がある．それを実現する方法として，以下のようなアプローチが考えられる．

5.1 方式 1：被コピー情報の記録

「このデータがこの時点でこのピアにコピーされた」という情報を各ピアにおいて記録する方式：たとえば，図 8 の例においては，ピア p2 がピア p1 からデータをコピーしたときに p2 のログリレーションの 1 行目のようなデータが記録されるが，これに相当する情報をピア p1 側にも記録するというものである．

このような情報が管理されているならば，コピー先へと問合せをフォワードしていくことにより，自分のピアが提供したデータがどのピアにコピーされていったかということを再帰的に追跡することができる．

しかし，このアプローチには問題点も存在する．問合せをしたピアが本当にデータをコピーしたのか，それとも問合せ結果を保存せず廃棄したのかが直接的には分からないという点である．これを把握するためには，コピーした側のピアがそれを保存する際（既存リレーションに追加する際），コピー元のピアにコピーした旨を通知する必要がある．これのための処理はオーバヘッドも大きく，実際の利用においては適用が容易ではないと考えられる．

5.2 方式 2：ブロードキャスト問合せによる解決

方式 1 がコピー情報の管理に大きな負担を払うのに対し，逆にコピーに関する情報の管理を極力省くアプローチとして，動的にブロードキャスト問合せを発行するアプローチも考えられる．すなわち，潜在的には P2P ネットワーク上のすべてのピアに対し，該当するピアからのデータのコピーを持っているかどうかを確認する問合せを発行する．これは一種のブロードキャストであると考えられる．このアプローチは明らかにオーバヘッドが大きいという欠点があるが，実現自体は容易である．

5.3 方式 3：中間的な手法

上記の方式 1, 2 の中間的な手法としては，方式 1 のようにデータをコピーしたという情報を完全に維持管理するのではなく，該当するデータをに対する問合せがあったというログのみを管理するアプローチがある．すなわち，他のピアから問合せが与えられたとき，実際に相手側のピアが問合せ結果のデータをコピーしたか否かに関わらず,「コピーした可能性がある」と記録するものである．追跡要求が出た場合には，コピーした可能性があるピアに対して問合せを発行する．情報の管理などに関するオーバヘッドと問合せ処理のコストのトレードオフを考えると，この方式は妥当な候補

といえるとも考えられる．

6 まとめと今後の課題

本稿では，P2P 環境におけるリレーショナルデータベースのデータの流通・交換を追跡する，トレーサビリティ確保のための手法について議論した．データの追跡に必要となるログ情報を注釈リレーションとして表現し，これを問合せ可能とすることでデータがどのように伝播していったか，修正されたかなどの追跡を可能とする．

本稿における提案にはさらに詳細な検討が必要である．また，管理コストと処理コストを考慮した実装方式の詳細化とその評価を今後の課題としたい．また，以下のような課題も存在する．

- 問合せモデルの開発：本稿ではリレーションを用いた注釈の表現と SQL 風の言語を用いた問合せ例を示したが，注釈情報の詳細化と問合せ言語の明確化を図る必要がある．5 節で述べた前向きの問合せを，問合せ言語を用いて完結に表現可能とすることも重要な課題である．
- 一部のピアが稼動していない，もしくは通信できないなどの際に，トレーサビリティを確保するための手法の開発：注釈情報の複製などを用いて安定した情報管理を行うアプローチなどが考えられる．
- 信頼できないピアがある場合の対応：本稿では P2P 環境上のすべてのピアが協調してトレーサビリティ機構を実現することを想定している．しかし，悪意があったり協調を十分に行わないピアが存在した場合にどのように対応可能かなどについて議論する必要がある．

以上の課題について今後取り組んでいきたいと考えている．

謝辞

本研究の一部は，日本学術振興会科学研究費基盤研究 (C)(2)(16500048)，旭硝子財団研究助成，稲森財団研究助成による．

参考文献

[1] K. Aberer and P. Cudré-Mauroux. Semantic overlay networks. In *Proc. VLDB*, p. 1367, 2005 (tutorial).

[2] P. A. Bernstein and T. Bergstraesser. Meta-data support for data transformations using Microsoft repository. *IEEE Data Eng. Bull.*, 22(1):9–14, 1999.

[3] D. Bhagwat, L. Chiticariu, W.-C. Tan, and G. Vijayvargiya. An annotation management system for relational databases. In *Proc. VLDB*, pp. 900–911, 2004.

[4] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In *Proc. ICDT*, Vol. 1973 of *LNCS*, pp. 316–330, 2001. Springer.

[5] P. Buneman, S. Khanna, and W.-C. Tan. On propagation of deletions and annotations through views. In *Proc. ACM PODS*, pp. 150–158, 2002.

[6] L. Chiticariu, W.-C. Tan, and G. Vijayvargiya. DBnotes: A post-it system for relational databases based on provenance. In *Proc. ACM SIGMOD*, pp. 942–944, 2005. (demo paper).

[7] Y. Cui and J. Widom. Lineage tracing in a data warehousing system. In *Proc. ICDE*, pp. 683–684, 2000.

[8] Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. In *Proc. VLDB*, pp. 471–480, 2001.

[9] A. Kementsietsidis, M. Arenas, and R. J. Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In *Proc. SIGMOD*, pp. 325–336, 2003.

[10] A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data management infrastructure for semantic web applications. In *Proc. WWW*, pp. 556–567, 2003.

[11] T. Lee, S. Bressan, and S. Madnick. Source attribution for querying against semi-structured documents. In *Workshop on Web Information and Data Management (WIDM'98)*, 1998.

[12] W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou. PeerDB: A P2P-based system for distributed data sharing. In *Proc. ICDE*, pp. 633–644, 2003.

[13] W.-C. Tan. Containment of relational queries with annotation propagation. In *Proc. DBPL*, pp. 37–53, 2003.

[14] W.-C. Tan. Research problems in data provenance. *IEEE Dat Eng. Bull.*, 27(4):45–52, 2004.

[15] Y. R. Wang and S. E. Madnick. A polygon model for heterogeneous database systems: The source tagging perspective. In *Proc. VLDB*, pp. 519–538, 1990.

[16] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *Proc. CIDR*, pp. 262–, 2005.

[17] A. Woodruff and M. Stonebraker. Supporting fine-grained data lineage in a database visualization environment. In *Proc. ICDE*, pp. 91–102, 1997.