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

2008－DBS－145 (14)
2008－ＦＩ－ 91 (14)
2008／6／20

トレーサブルなP2Pレコード交換システムにおける問合せ処理

李 峰栄[†] 飯田 卓也[†] 石川 佳治[††]
[†]名古屋大学大学院情報科学研究科
[††]名古屋大学情報連携基盤センター
[†]{lifr,iida}@db.itc.nagoya-u.ac.jp, [††]ishikawa@itc.nagoya-u.ac.jp

概要

近年，ピアツーピア (P2P) と呼ばれる新しいネットワークサービス形態における情報交換が頻繁になっている。しかし，データの複製や変更がネットワークの各所で発生するため，データの信頼性の欠如が問題になる．この問題を解決するため，我々はネットワークにおける情報流通の過程を追跡可能とする，データベース技術を基盤としたレコード交換システム機構を提案している．このシステムはP2Pネットワーク上で協調するピア間において再帰的なデータベース問合せを分散実行することで追跡処理を行う．本稿では，トレーサビリティを実現するための基盤となる問合せ処理について述べる．

Query Processing in Traceable P2P Record Exchange System

Fengrong LI[†], Takuya IIDA[†], and Yoshiharu ISHIKAWA[††]
[†]Graduate School of Information Science, Nagoya University
[††]Information Technology Center, Nagoya University
[†]{lifr,iida}@db.itc.nagoya-u.ac.jp, [††]ishikawa@itc.nagoya-u.ac.jp

Abstract

Information exchanges in peer-to-peer networks have become very popular in recent years, but, there is a lack of reliability among the data exchanged because data replications and modifications are performed independently by peers. To solve this problem and to provide reliable and flexible information exchange facilities in P2P networks, we proposed a framework for a record exchange system based on database technologies. In this system, tracing operations are executed as distributed recursive queries among cooperating peers in a P2P network. This paper focuses on query processing which is the basis for realizing traceability.

1 はじめに

近年では，**P2P**（**peer-to-peer**）技術の普及により，ネットワーク上に存在する多数の自律的なピア間で，柔軟に情報の送受信を行うことが可能となっており，そのアプリケーションも増大している．具体的な例として，バイオインフォマティクスの分野では，関連する研究グループ間での迅速な情報交換・流通のために緩やかな情報の共有が求められており，P2P 技術が有望とされている [4]．しかし，P2P ネットワーク上の情報交換・流通では，特定のサーバによる管理が行われないことから，データの複製や変更がネット

ワークの各所で発生し，また，その把握も容易ではなく，このことは流通する情報の信頼性の欠如へとつながる．ネットワークを介して入手した情報の信頼性を保障するためには，入手した情報がどのような経緯で手元に存在するのかという証拠が必要となる．

このような背景から，我々の研究グループでは，P2Pネットワークにおける情報交換により得られたデータの信頼性を確保するための基盤技術の確立を目指して，流通するデータの**トレーサビリティ**（traceability）を実現するためのレコード交換システムのアーキテクチャを提案した [5, 6, 7]．各ピアでは，レコード（タプル）形式のデータの集合が管理され，他のピアとの交換が自律的に行われる．レコードの交換・変更の履歴は，データとともに各ピア上のリレーショナルデータベースシステムを用いて分散管理される．トレーサビリティに関する追跡処理が発生した際には，再帰的問合せをP2Pネットワーク上で分散実行して情報を収集するという方式で実現を図る．本稿では，問合せ処理に重点をおき，具体的には，論理レイヤの問合せをどのように物理レイヤの問合せに変換するか，変換された問合せをどのように実行するかについて議論する．

本稿の構成は以下のようになる．2節で関連研究を述べる．3節でP2Pレコード交換システムについて述べ，4節では論理レイヤにおける問合せについて述べ，5節で物理レイヤにおける問合せ変換について述べる．6節で問合せ処理のアプローチを示す．最後に7節でまとめと今後の課題を述べる．

2 関連研究

P2Pデータベースに関しては，動的な環境でのデータ管理，異種のスキーマのマッピングや，問合せ言語の開発，索引・複製技術など，さまざまな研究がなされている [1]．これに対し本研究では，P2Pネットワーク上の情報交換を背後で支え，トレーサビリティを実現するための基盤技術としてデータベース技術を用いる点に特徴がある．

本研究で提案された問合せ処理のアプローチは [8] における，ネットワーク上のデータに対する効率的な問合せ処理を実現するための枠組みである *declarative networking* と関連が深いが，こちらでは主としてセンサネットワークなどが対象とされている．一方，本研究ではP2Pネットワーク環境を対象とし，トレーサビリティに焦点を当てている点が異なっている．また，[8] では，datalogの問合せ処理手法のうちで基本的であるseminaive問合せ処理法 [2] をもとにした問合せ法が示されている．本アーキテクチャにおいてもseminaive法による問合せ評価法が有効であると考えられるが，declarative networkingのアプローチと比べて基本的な前提に多くの違いがあるため，独自の拡張を行う．

データの出所の追跡や保障をする問題は，*lineage tracing* あるいは *data provenance* などと呼ばれており，現在大いに注目を浴びているトピックである [9, 10]．本研究ではこれをデータの**系統管理**と呼んでいる．レコード交換システムは，個々のピアの自律性を保ちつつ，相互の情報の連携を図るという点で魅力を持つが，単純な実装ではさまざまな問題が発生する．本研究では，以下のような機能の実現を目指している．

1. **レコードの出所の把握**：問合せ対象のレコードがどのピアで作成され，どのようなピアを経由して得られたかを把握できる．
2. **レコードの重複の検出**：同じ値を持ったレコードが複数存在するとき，それが別々のピアから得られたものか，また，同じピアから複数回取得されたかを検出できる．
3. **レコードの行き先の取得**：自身が提供したレコードに誤りを発見した場合など，そのレコードの複製を有するピアを追跡したいことがある．本提案手法の技術により，データをコピーしたピアの探索が行える．
4. **レコード更新処理の追跡**：データベースにおけるデータの管理では，しばしば更新処理が伴う．この機能により，先に入手したレコードがその後更新を受けたかなどの情報を取得できる．

3 P2Pレコード交換システム

3.1 レコード交換システムの概念

本研究では，P2Pネットワーク上のピア間でレコードが交換される**レコード交換システム**（record exchange system）を考える．**レコード**（record）とは，属性と値からなるタプル構造のデータを意味しており，そのスキーマ（属性集合と各属性の役割）はP2Pネットワーク上で共有されているとする．各ピアには基本的には1人のユーザが対応し，そのユーザが興味あるレコードの集合を保持すると想定する．各ピアは，P2Pネットワーク上の他のピアに対して条件を指定した問合せを行い，問合せ結果のレコードを自身が管理するレコード集合に選択・追加することが可能であるとする．

本研究では各ピアにおけるレコード集合の内容が同一であることは想定しない．各ピアは自律的に個々のレコード集合を管理しており，他のピアのレコード集合と内容の同期などは特に行わないものとする．また，本研究では，P2P ネットワーク上に存在する各ピアにおいて，リレーショナルデータベース管理システム（RDBMS）が稼動していることを想定する．

図 1 に，例としてレコード集合 Novel を示す．このレコード集合は小説のタイトル，著者，言語，発表年度に関する情報を含んでおり，ピア A により所有されているとする．ピア A と連携する他のピアにも，同じ構造を持ったレコード集合 Novel が存在するが，その内容は必ずしも同じではない．

title	author	language	year
Pride and Prejudice	Jane Austen	English	1813
David Copperfield	Charles Dickens	English	1849
Madame Bovary	Gustave Flaubert	French	1857

図 1: ピア A のレコード集合 Novel

3.2 レコード交換システムの構成

レコード交換システムは以下の 3 層から構成される．

1. **ユーザレイヤ**：ユーザに対するインタフェースとしての問合せ機能や追跡処理の機能を提供する．
2. **論理レイヤ**：P2P ネットワーク上に分散したデータベースを統合した，トレーサビリティ支援のための仮想的なビューを構成する．
3. **物理レイヤ**：論理レイヤの仮想的なビューに対する問合せを，自律的で分散したピアの協調に基づいて実行する．

以下でこれらの概要について述べる．

3.2.1 論理レイヤのリレーション

論理レイヤでは，各ピアの情報を統合した仮想的なビューを構成する．実際のデータは各ピアに分散しているが，論理レイヤを設けることで見通しを立てやすくし，追跡処理のための各種機能の実現を容易にする．図 1 に示したレコード集合の例をさらに簡略化して説明する．図 2 は，それぞれピア A, B, C に保持されている，ユーザレイヤの小説（Novel）に関するレコード集合である．各レコードには，小説のタイトルと著者の情報が含まれている．

ピア A		ピア B		ピア C	
title	author	title	author	title	author
t1	a1	t1	a1	t1	a1
t5	a5	t2	a3		

図 2: 各ピアにおけるレコード集合 Novel

論理レイヤおよび物理レイヤでは，リレーショナルデータモデルに基づいて，トレーサビリティのための情報を表現する．図 3 の Data[Novel] リレーションは，先に示したピア A, B, C に含まれるレコード集合 Novel の全レコードを統合したビューを表す．なお，各リレーションに現在存在している値だけでなく，過去に存在していたが，修正・削除の結果，ユーザの立場からは存在していないレコードについても情報が含まれる．Data[Novel] リレーションの左側の 2 つの属性はユーザレイヤのレコードの属性を表している．残りの属性は管理用の属性である．peer 属性にはそのレコードを所有するピアの情報が記述され，id 属性には各ピアがレコードに付与する論理 ID が収められる．

title	author	peer	id
t1	a1	A	#A011
t5	a5	A	#A028
t1	a1	B	#B032
t2	a2	B	#B040
t2	a3	B	#B051
t1	a1	C	#C005
t6	a6	C	#C088

図 3: ビュー Data[Novel]

図 4 に示す Change[Novel] ビューは，挿入・修正・削除に関する履歴を保持する大域的なビューである．from_id 属性は修正前のレコード ID を，to_id 属性は修正後のレコード ID を表し，time 属性は修正時のタイムスタンプを表す．from_id 属性が空値（–）である場合はレコードの挿入を表し，to_id 属性が空値である場合はレコードの削除を表す．このように，履歴情報と論理的な ID を用いて，トレーサビリティのために必要となる情報を表現する．

from_id	to_id	time
–	#A011	5/2/07
–	#A028	8/18/07
–	#B032	4/10/07
–	#B040	4/20/07
#B040	#B051	6/10/07
–	#C005	3/20/07
#C088	–	01/02/08

図 4: ビュー Change[Novel]

図 5 に示すリレーション Exchange[Novel] ビューも，論理レイヤにおいて仮想的に構築される大域的なビューである．このリレーションは，ピア間のレコードの交換に関する情報を保持する．from_peer, to_peer はレコードのコピー元，コピー先をそれぞれ表し，from_id,

to_id は各レコードのそれぞれのピアにおける論理 ID を表す．time 属性はタイムスタンプ情報であり，コピー先のピアにレコードがコピーされた時刻を表す．たとえば，1 番目のタプルは，ピア A がピア B からレコードを複製したことを示している．このような情報により，ピア間でのレコードの複製を追跡する．

from_peer	to_peer	from_id	to_id	time
B	A	#B032	#A011	5/2/07
C	B	#C005	#B032	4/10/07

図 5: ビュー Exchange[Novel]

3.2.2 物理レイヤのリレーション

論理レイヤでは，P2P ネットワーク上のデータベース中に存在する情報をすべて統合した仮想的なビューを用いて問合せを表現した．しかし，実際には，各ピアはネットワーク上のすべての情報に必ずしも興味があるわけではなく，興味は限定されたものであり，ネットワーク全体のデータベースの情報を一括して管理することの意義は薄い．このレイヤでは，各ピアは，自身に関連する情報のみを個別に管理することを想定する．

以下は物理レイヤのリレーションである．図 6, 7 に，図 3, 4 に示した論理レイヤのリレーション Data[Novel] および Change[Novel] に対応する，ピア A における物理レイヤのリレーションを示す．具体的には，ピア A に関するタプルのみを抜き出したものが内容となる．なお，図 6 では，peer 属性の値が A であることは明らかであるため，この属性は削除している．

title	author	id
t1	a1	#A011
t5	a5	#A028

図 6: ピア A の Data[Novel]

from_id	to_id	time
–	#A011	5/2/07
–	#A028	8/18/07

図 7: ピア A の Change[Novel]

論理レイヤのリレーション Exchange[Novel] に関しては，その内容をピア間で分散して管理する．ピア A における物理レイヤのリレーション From[Novel] では，ピア A が受け取ったレコードに関する情報のみを保持する．以下の例では，#A011 というレコードは，ピア B の#B032 という ID のレコードを複製したことを意味している．#A028 というレコードはピア A における From[Novel] に含まれていない．すなわち，ピア A が自分でこのレコードを作成したということを表す．

id	from_peer	from_id	time
#A011	B	#B032	5/2/07

図 8: ピア A における From[Novel]

一方，各ピアでは，自身がレコードをどのピアに渡したかという情報も記録する．たとえば，上に示したピア A の From[Novel] のタプルに対応して，図 9 に示すように，ピア B の To[Novel] リレーション中で対応するタプルを管理する．ここでは，ピア B の#B032 というレコードが，ピア A に複製されて#A011 という ID で格納されたことを表す．図 9 ではピア B の To[Novel] を示したが，ピア A でも同様に To[Novel] リレーションが管理される．From[Novel] リレーションはレコードの複製元にさかのぼって追跡する場合に利用され，To[Novel] はレコードの複製先に向かって追跡する場合に利用される．

id	to_peer	to_id	time
#B032	A	#A011	5/2/07

図 9: ピア B の To[Novel]

4 論理レイヤにおける問合せ

論理レイヤにおけるトレーサビリティ実現のための問合せ記述について述べる．情報を追跡するには再帰的な処理が求められることから，datalog [2] を用いた記述を試みる．datalog は，近年ではネットワーク上での問合せ処理などで，新たな応用が見られる [8]．以下では問合せの記述例を示す．

問合せ 1：ピア A が保持する，タイトル t1，著者 a1 というレコードに対し，その元々のデータを最初に作成したピアを知りたいとする．そのような問合せは以下のように記述できる．英数大文字により変数名を表し，下線（_）により無名の変数を表す．最後の行に示すルールが問合せを表す．

```
BReach(P, I1) :- Data[Novel]('t1', 'a1', 'A', I2),
                 Exchange[Novel](P, 'A', I1, I2, _)
BReach(P1, I1) :- BReach(P2, I2),
                  Exchange[Novel](P1, P2, I1, I2, _)
Origin(P) :- BReach(P, I), NOT Exchange[Novel](_, P, _, I)
Query(P) :- Origin(P)
```

図 3 の例についてこの問合せを実行すると，結果として，Origin(P) の変数 P には “C” が束縛される．すなわち，ピア A の該当するレコードはピア C が最初に作成したことがわかる．

問合せ 2：ピア A が保持するレコードのうち，どれがピア B を介して得られたものかを求めたい．

```
BReach2(P, I1, T, A) :- Data[Novel](T, A, 'A', I2),
                        Exchange[Novel](P, 'A', I1, I2, _)
BReach2(P1, I1, T, A) :- BReach2(P2, I2, T, A),
                         Exchange[Novel](P1, P2, I1, I2, _)
ViaB(T, A) :- BReach2('B', _, T, A)
Query(T, A) :- ViaB(T, A)
```

この例以外にも，たとえば以下のような問合せを記述することができる．

- ピア A が新たに入手したレコードが，既に手元に保持しているレコードと同一のものかを調べる問合せ
- ピア A が保持するレコードが最新の情報かどうかをチェックする問合せ：ピア A にそのレコードが到着する過程で経由したピア上で，その後そのレコードに対して更新が行われたかどうかを調べる．
- ピア A が提供したレコードの複製を持っているピアをすべて列挙する問合せ

具体的な問合せ記述例は [6] にある．

5 物理レイヤへの問合せ変換

論理レイヤにおける問合せでは，P2P ネットワーク上の情報をすべて統合した仮想的なビューについて問合せが記述される．一方，それを実行する物理レイヤでは，リレーションは各ピアに分散して管理されている．そこで，論理レイヤにおける問合せを処理するには，与えられた問合せを，まず物理レイヤの構成に応じて変換する必要がある．本節では，論理レイヤの問合せを物理レイヤの問合せに変換するための規則を示す．与えられた問合せ内のリレーションの種類に応じて，それらを適宜書き換えることになる．具体的には，問合せ中の外延（*edb*）リレーションの書き換えを行い，内包（*idb*）リレーションについては変換は行わない．

Data リレーションに対する変換規則　R をユーザレイヤの該当リレーション名とし，attrs を R と同じ次数を有する変数または定数からなるリストとしたとき，

Rule D1:
`Data[R](attrs, peer, id)` → `Data[R]@peer(attrs, id)`

とする．論理レイヤと物理レイヤでは，どちらも `Data[R]` という表記を用いているが，物理レイヤにおいては `@` 記号の後にそのリレーションを保持するピアの名前（もしくはピア名を保持する変数）を付与することで，特定のピアへの束縛を表現する．例えば，論理レイヤの問合せ 1 で現れる `Data[Novel]('t1', 'a1', 'A', I2)` は，物理レイヤの `Data[Novel]@'A'('t1', 'a1', I2)` へとマッピングされる．

Change リレーションに対する変換規則　基本的な考え方は Data リレーションの場合と同様であり，以下のようになる．

Rule C1:
`Change[R](peer, from_id, to_id, time)`
→ `Change[R]@peer(from_id, to_id, time)`

となる．

Exchange リレーションに対する変換規則　場合分けを行うことで処理する．まず，第 1 引数がピア名を表す定数 'P' の場合，次のように変換する．

Rule E1:
`Exchange[R]('P', to_peer, from_id, to_id, time)`
→ `To[R]@'P'(from_id, to_peer, to_id, time)`

この変換の理由について説明する．ここでは，ピア P からレコードを送ることで情報の交換が行われたという情報が Exchange リレーションに記載されている．そこで，ピア P における To リレーションの情報にアクセスするように問合せを書き換える．

第 2 引数がピア名を表す定数 'P' の場合，逆に，From リレーションを用いるように

Rule E2:
`Exchange[R](from_peer, 'P', from_id, to_id, time)`
→ `From[R]@'P'(to_id, from_peer, from_id, time)`

と変換する．

また，一方のピア変数が無名変数 '_' の場合には，もう一方のピア変数（または定数）が `@` 以降に続くように変換する．

Rule E3:
`Exchange[R](from_peer, _, from_id, to_id, time)`
→ `To[R]@from_peer(from_id, _, to_id, time)`

および

Rule E4:
`Exchange[R](_, to_peer, from_id, to_id, time)`
→ `From[R]@to_peer(to_id, _, from_id, time)`

で与えられる．

なお，両方のピア変数が定数でも無名変数でもない場合，上に示したような変数の束縛に関する分析を行い，実行時に束縛される変数を特定し，上記 E1 または E2 の変換規則と同様に処理する．両者の変数とも束縛される場合には，変換規則 E1 と E2 のどちらを選んでもよい（コスト評価結果などを利用することも考えられる）．両者とも束縛されない場合は，問合せ

自体の評価ができないため，そもそも問合せ自体に誤りがあると考えてよい．

また，リレーション Exchange において，両方のピア変数が定数の場合も考えられる（ピア A からピア B にどのレコードが渡されたかを知りたい場合など）．この場合にも，上記 E1 または E2 のどちらの変換規則を用いてもよい（コスト評価結果などを利用する）．

上記の変換規則を先に示した問合せ 1 に適用すると，以下のようになる．

```
BReach(P, I1) :- Data[Novel]@'A'('t1', 'a1', I2),
                 From[Novel]@'A'(I2, P, I1, _)
BReach(P1, I1) :- BReach(P2, I2),
                  From[Novel]@P2(I2, P1, I1, _)
Origin(P) :- BReach(P, I), NOT From[Novel]@P(I, _, _, _)
Query(P) :- Origin(P)
```

また，問合せ 2 を変換した結果は次のようになる．

```
BReach2(P, I1, T, A) :- Data[Novel]@'A'(T, A, I2),
                        From[Novel]@'A'(I2, P, I1, _)
BReach2(P1, I1, T, A) :- BReach2(P2, I2, T, A),
                         From[Novel]@P2(I2, P1, I1, _)
ViaB(T, A) :- BReach2('B', _, T, A)
Query(T, A) :- ViaB(T, A)
```

6 物理レイヤにおける問合せ処理

前節では，与えられた論理レイヤの問合せをどのようにして物理レイヤの問合せに変換するかについて述べた．本節では，物理レイヤの問合せをどのように処理するかの概略について述べる．

ここでは，説明を簡略化するため，問合せ 1 をさらに簡単にした以下の問合せ 1' を考える．この問合せは，ピア A が保持するタイトル t1，著者 a1 というレコードがピア A に到着するまでの経路上にあるすべてのピアの名前を出力する．

問合せ 1'：ピア A が保持する，タイトル t1，著者 a1 というレコードに対し，その元々のレコードを作成したピアから，ピア A がそれを得るまでに経由した経路上のすべてのピアの名前を求めよ．論理レイヤにおけるこの問合せの記述は以下のようになる．

論理レイヤの問合せ 1'：

```
BReach(P, I1) :- Data[Novel]('t1', 'a1', 'A', I2),
                 Exchange[Novel](P, 'A', I1, I2, _)
BReach(P1, I1) :- BReach(P2, I2),
                  From[Novel](P1, P2, I1, I2, _)
Query(P) :- BReach(P, _)
```

一方，この問合せを先節の規則により物理レイヤの問合せに変換すると，以下のようになる．

物理レイヤの問合せ 1'：

```
BReach(P, I1) :- Data[Novel]@'A'('t1', 'a1', I2),
                 From[Novel]@'A'(I2, P, I1, _)
BReach(P1, I1) :- BReach(P2, I2),
                  From[Novel]@P2(I2, P1, I1, _)
Query(P) :- BReach(P, _)
```

以下では，問合せ処理を具体例を踏まえながら段階を追って説明する．

ステップ 1：即座に実行できるルールの発見と実行　まず，ステップ 1 として，即座に処理可能なルールを見つける．即座に処理可能なルールとは，

- 本体部分に *edb* 述語しか現れない
- 各 *edb* 述語のロケーション部分が定数（例:`@'A'`）であること

を満たすようなルールである．上の例の場合，第 1 のルール

```
BReach(P, I1) :- Data[Novel]@'A'('t1', 'a1', I2),
                 From[Novel]@'A'(I2, P, I1, _)
```

がこれに該当する．すなわち，このルールはピア A 内で実行可能である．

このようなルールは再帰を含まないため，即座に SQL に変換して実行できる．上のルールについては，ピア A 上で

```
SELECT f.to_peer, f.to_id
FROM Data[Novel] AS d, From[Novel] AS f
WHERE d.title = 't1' AND d.author = 'a1'
      AND d.id = f.id
```

という SQL 問合せを実行し，結果を一時リレーションに格納すればよい．この結果，BReach には，ピア A の該当するレコードを直接入手したピアとそのピアにおけるレコードの ID が入る．

なお，これまでに示した問合せ例にはまったく出てこないが，

```
BReach(P, I1) :- Data[Novel]@'A'('t1', 'a1', I2),
                 From[Novel]@'B'(I2, P, I1, _)
```

というようなルールも記述することは可能である．この場合には，分散データベースにおける問合せ処理の技術を使うことになる．ピア A, B のどちらかがマスタとなり，たとえば A がマスタの場合には，まずピア A が Data[Novel] からタイトル t1，著者名 a1 のレコードの ID の選択処理を行い，その ID をピア B に送信する．これを受け取ったピア B は自身の From[Novel] と結合処理を行い，対応するピア名，ID 番号を選んでピア A に返す．

ステップ2：再帰的問合せの発行　ステップ1を実行した後について考える．このとき，ピア A で1番目のルールを問合せ処理した結果である，BReach の部分的な結果がリレーションとして存在する．このリレーションを BReachA と呼ぶことにする．

1番目のルールは，その実行がすでに終了したため，その結果である BReachA を考慮し，プログラムを以下のように書き換える．

問合せ2の変換結果:

```
BReach(P, I) :- BReachA(P, I)
BReach(P1, I1) :- BReach(P2, I2),
                  From[Novel]@P2(I2, P1, I1, _)
Query(P) :- BReach(P)
```

ステップ3のための議論：Seminaive 方式での実行（非分散の場合）　変換された問合せ2を，datalog の基本的な問合せ処理戦略の一つである seminaive 方式 [2] で実行することを考える．ただし，まず準備として，データベースが各ピアに分散しておらず，1箇所に集中して管理されている場合を考える．問合せ処理のステップは次のようになる．

1. 各 *idb* リレーションを $\texttt{BReach} := \emptyset, \texttt{Query} := \emptyset$ と初期化
2. 本体に *idb* 述語が現れないルール（の集合）を先に実行して，差分（Δ）リレーションを初期化する：この例では，1番目のルールが対応する．1番目のルールを実行すると，BReachA の内容が BReach にコピーされ，それが $\Delta_{\texttt{BReach}}$ の内容となる．Query については何も起きないため，$\Delta_{\texttt{Query}} = \emptyset$ である．
3. 各 *idb* 述語（この場合 BReach と Query）について，繰返し処理を行う．具体的には，まず BReach について，

   $$\texttt{BReach} := \texttt{BReach} \cup \Delta_{\texttt{BReach}}$$

   および，問合せ

   $$\texttt{temp}_{\texttt{BReach}}(\texttt{p1}, \texttt{i1}) \leftarrow \Delta_{\texttt{BReach}}(\texttt{p2}, \texttt{i2}), \texttt{From[Novel]@p2}(\texttt{i2}, \texttt{p1}, \texttt{i1}, \_)$$

   を実行する．そして，

   $$\Delta_{\texttt{BReach}} := \texttt{temp}_{\texttt{BReach}} - \texttt{BReach}$$

   を実行する．
   Query についても同様に，

   $$\begin{aligned} \texttt{Query} &:= \texttt{Query} \cup \Delta_{\texttt{BReach}} \\ \texttt{temp}_{\texttt{Query}}(\texttt{p}) &\leftarrow \Delta_{\texttt{BReach}}(\texttt{p}, \_) \\ \Delta_{\texttt{Query}} &:= \texttt{temp}_{\texttt{Query}} - \texttt{Query} \end{aligned}$$

   を実行する．もし，差分リレーション $\Delta_{\texttt{BReach}}, \Delta_{\texttt{Query}}$ の双方が空集合になった場合には終了する．そうでなければ，3に戻って同様の処理を繰り返す．

ステップ3：Seminaive 方式での実行　実際には，ピア上にリレーションが分散していることを考慮しなければならない．seminaive 方式による問合せ処理を例を用いて説明する．

1. ピア A は問合せを開始する．まず空のリレーション BReach, Query を作成する．
2. ピア A ではステップ1の実行により，以下の BReachA が得られていたとする．

P	I
B	#B032

   この結果を $\Delta_{\texttt{BReach}}$ に代入する（$\Delta_{\texttt{BReach}} := \texttt{BReachA}$）．また，この場合 $\Delta_{\texttt{Query}}$ については何も処理が発生しないので，何も行わない．
3. 繰返し処理を開始する．ピア A は

   $$\texttt{BReach} := \texttt{BReach} \cup \Delta_{\texttt{BReach}}$$

   を実行する．この結果，BReach の内容は

P	I
B	#B032

   となる．
   次に，ピア A は，繰返し処理の次のステップ

   $$\texttt{temp}_{\texttt{BReach}}(\texttt{p1}, \texttt{i1}) \leftarrow \Delta_{\texttt{BReach}}(\texttt{p2}, \texttt{i2}), \texttt{From[Novel]@p2}(\texttt{i2}, \texttt{p1}, \texttt{i1}, \_)$$

   を実行したい．これは具体的には，$\Delta_{\texttt{BReach}}$ の各タプルについて上記のルールを評価することで処理できる．この場合 $\Delta_{\texttt{BReach}}$ にはタプル (B, #B032) のみしかないため，このタプルについて実行処理を行うことになる．ここで，p2 = 'B', i2 = '#B032' と変数を置換すると，先のルールは

   $$\texttt{temp}_{\texttt{BReach}}(\texttt{p1}, \texttt{i1}) \leftarrow \Delta_{\texttt{BReach}}(\texttt{B}, \#\texttt{B032}), \texttt{From[Novel]@'B'}(\texttt{i2}, \texttt{p1}, \texttt{i1}, \_)$$

   と具体化される．このルールは，ピア B でないと効率的に評価できないことに注意する．そのため，このルールとそれに引き続く処理

   $$\begin{aligned} &\texttt{temp}_{\texttt{BReach}}(\texttt{p1}, \texttt{i1}) \leftarrow \Delta_{\texttt{BReach}}(\texttt{B}, \#\texttt{B032}), \texttt{From[Novel]@'B'}(\texttt{i2}, \texttt{p1}, \texttt{i1}, \_) \\ &\Delta_{\texttt{BReach}} := \texttt{temp}_{\texttt{BReach}} - \texttt{BReach} \end{aligned}$$

   の処理をピア B に**フォワード**することになる．
   同様に Query についても考える．

   $$\texttt{Query} := \texttt{Query} \cup \Delta_{\texttt{BReach}}$$

   の実行結果は空である．次いで

   $$temp_{\texttt{Query}}(\texttt{p}) \leftarrow \Delta_{\texttt{BReach}}(\texttt{p}, \_)$$

   を実行すると，$\texttt{temp}_{\texttt{Query}}(\texttt{p})$ の内容は

P
B

   となる．なお，本研究における問合せのセマンティクスを考えると，新しいタプルが Query に追加されると，その結果はすぐにピア A に報告したい．このため，上記の結果（B）はピア A の問合せ結果としてすぐに返答することも考えられる．

次いで

$$\Delta_{\mathtt{Query}} := \mathtt{temp}_{\mathtt{Query}} - \mathtt{Query}$$

を実行する．つまり，いまの処理では，ピア A に保持されている情報で処理できる範囲の問合せ処理が終了したことになる．次はフォワード処理になる．

4. 具体的には，ピア A は以下の情報をピア B にフォワードする．

   - BReach, $\Delta_{\mathtt{BReach}}$
   - Query, $\Delta_{\mathtt{Query}}$
   - 問合せプログラム：具体的には

$$\begin{aligned}
&\mathtt{BReach} := \mathtt{BReach} \cup \Delta_{\mathtt{BReach}}\\
&\mathtt{temp}_{\mathtt{BReach}}(\mathtt{p1},\mathtt{i1}) \leftarrow \Delta_{\mathtt{BReach}}(\mathtt{p2},\mathtt{i2}),\\
&\qquad\qquad \mathtt{From[Novel]@p2}(\mathtt{i2},\mathtt{p1},\mathtt{i1},_)\\
&\Delta_{\mathtt{BReach}} := \mathtt{temp}_{\mathtt{BReach}} - \mathtt{BReach}
\end{aligned}$$

   および

$$\begin{aligned}
&\mathtt{Query} := \mathtt{Query} \cup \Delta_{\mathtt{BReach}}\\
&\mathtt{temp}_{\mathtt{Query}}(\mathtt{p}) \leftarrow \Delta_{\mathtt{BReach}}(\mathtt{p},_)\\
&\Delta_{\mathtt{Query}} := \mathtt{temp}_{\mathtt{Query}} - \mathtt{Query}
\end{aligned}$$

   である．

   実際には，$\Delta_{\mathtt{BReach}}$ については，問合せに関与する部分（ピア B に対応する部分）のみフォワードすればよい．ただし，この問合せ例については，さらに簡略化できる余地もある．

具体的な問合せの処理システムは現在開発中である．問合せ処理だけではなく，処理の効率化について検討する必要もある．今後の課題としたい．

7 まとめと今後の課題

本研究では，P2P ネットワークにおける緩やかな情報交換を実現するためのレコード交換システムに基づいて，その際に重要となる情報の出所，行き先の追跡のためのトレーサビリティの考え方について述べた．トレーサビリティを実現するためのレコード交換システム機構について述べ，問合せ処理の方式を具体的に論じた．特に，論理レイヤの問合せを物理レイヤの問合せ処理に変換するステップを示し，また，物理レイヤの問合せを seminaive 方式で実行する場合の概略を示した．

現在，提案した問合せ処理アプローチに基づく P2P レコード交換システムのプロトタイプを作成中である [3]．以下の課題について今後取り組んでいきたいと考えている．

- 物理レイヤにおける問合せ処理のさらなる詳細化：seminaive 方式に加え，マジックセット法などの利用も考慮し，分散環境での問合せ処理方式の開発を図る．最適化処理の実現などについても検討したい．
- 物理レイヤにおける他の効率化方式の検討：ピア間で情報を複製するなどにより，データの安全性を高め，問合せの応答時間を高めることが可能である．このような複製処理の実現方式と，複製が存在する場合の問合せ処理方式について検討したい．
- プロトタイプシステムの開発と評価：管理コストと処理コストを考慮したプロトタイプシステムの実現および実験による評価を行う．

謝辞

本研究の一部は，日本学術振興会科学研究費特定領域研究（19300027）および放送文化基金の助成による．

参考文献

[1] K. Aberer and P. Cudre-Mauroux. Semantic overlay networks. In *VLDB*, 2005. (tutorial notes).

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] 飯田卓也, 李峰栄, 石川佳治. トレーサビリティ機構を有する P2P レコード交換システムの開発. 情報処理学会第 70 回全国大会, 2008.

[4] Z. Ives, N. Khandelwal, A. Kapur, and M. Cakir. Orchestra: Rapid, collaborative sharing of dynamic data. In *Proc. CIDR*, pp. 107–118, 2005.

[5] 李峰栄, 飯田卓也, 石川佳治. P2P ネットワークにおけるトレーサビリティを有するレコード交換システム機構. 電子情報通信学会第 19 回データ工学ワークショップ (DEWS 2008), 2008.

[6] 李峰栄, 石川佳治. トレーサブルな P2P 情報流通のためのデータモデルの提案. 情報処理学会研究報告, 2007(65):461–466, 2007.

[7] F. Li and Y. Ishikawa. Traceable P2P record exchange based on database technologies. In *Proc. Asia Pacific Web Conference (APWeb 2008)*, pp. 660–671, 2008.

[8] B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M. Hellerstein, P. Maniatis, R. Ramakrishman, T. Roscoe, and I. Stoica. Declarative networking: Language, execution and optimization. In *Proc. ACM SIGMOD*, pp. 97–108, 2006.

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

[10] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *Proc. CIDR*, pp. 262–276, 2005.