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

2008－DBS－146 (17)
2008／9／22

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の効率化について

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
E-mail: †{lifr,iida}@db.itc.nagoya-u.ac.jp, ††ishikawa@itc.nagoya-u.ac.jp

あらまし 近年，科学などの様々な分野では，データの信頼性を確保するために，データの出所を追跡するあるいはデータの流通過程を追跡する系統管理（data provenance, lineage tracing）が大きな課題になっている．本研究は，情報交換が頻繁になっているP2Pネットワークにおける系統管理に着目して，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を可能とするレコード交換システム機構を提案している．本稿では，系統管理を実現するための基盤となる効率的な問合せ処理方法を述べる．
キーワード P2Pデータベース，レコード交換，トレーサビリティ，問合せ処理

Efficient Query Processing in a Traceable P2P Record Exchange System

Fengrong LI†, Takuya IIDA†, and Yoshiharu ISHIKAWA††

† Graduate School of Information Science, Nagoya University
†† Information Technology Center, Nagoya University
E-mail: †{lifr,iida}@db.itc.nagoya-u.ac.jp, ††ishikawa@itc.nagoya-u.ac.jp

Abstract In recent years, data provenance or lineage tracing which refers to the process of tracing and recording the origins of data and its movement between databases has become an acute issue in many fields, such as scientific databases. This research focuses on data provenance of information exchanges in peer-to-peer networks in which data replications and modifications are performed independently by peers. To ensure reliability among the data exchanged 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 discusses efficient query processing methods which are the basis for realizing traceability.
Key words P2P database, record exchange, traceability, query processing

1. はじめに

近年，低コストで運用可能，耐障害性が高いと言われる **peer-to-peer (P2P)** ネットワークを利用したアプリケーションの普及が目覚しく，幅広い分野で利用されている．例えば，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ネットワークでは，ネットワークに接続されたピアをすべて対等な存在と見なし，ネットワーク内に特定のサーバを設置することなくサービスの提供を行うので，データの複製や変更が情報交換・流通中にネットワークの各所で発生する．そのため，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のは非常に困難である．データの信頼性を確保するために，データの出所の追跡や保障をする系統管理 (data provenance, lineage tracing) は多くのアプリケーションにとって重要である．たとえば，科学技術分野における情報交換においては，情報の信頼性は極めて重要である [3]．

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], [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 [5], [6]．本稿では，問合せ処理の効率化に重点をおき，議論する．

本稿の構成は以下のようになる．2. でトレーサブルな P2P レコード交換システムについて述べ，3. では論理レイヤにおける問合せについて述べ，4. で物理レイヤにおける問合せ処理の効率化について述べる．5. で関連研究を述べる．最後に 6. でまとめと今後の課題を述べる．

2. トレーサブルな P2P レコード交換システム

2.1 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を考える．レコードとは，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．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．

例えば，図 1 に示すように，ある時刻，ピア A-F はそれぞれこのような小説集合を保存している．各ピアには，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．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．基本的には，各ピアに 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．

しかし，P2P ネットワークを介して取得されたデータに対しては必ずしも十分な信頼がおけないことから，トレーサビリティが重要である．そのため，本研究では，トレーサブルなレコード交換システムを提案している．

ピア A

title	author
t1	a1
t5	a5
t7	a6

ピア B

title	author
t1	a1
t2	a2

ピア C

title	author
t1	a1
t6	a6

ピア D

title	author
t5	a5

ピア E

title	author
t1	a1
t3	a3

ピア F

title	author
t1	a1
t0	a0

図 1　各ピアにおけるレコード集合 Novel

2.2 レコード交換システムの構成

P2P 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ため，以下のような機能の実現が不可欠である．

- **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．
- **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．
- **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する．
- **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．

これらの機能を確保するために，我々は 3 層から構成されるレコード交換システムを提案した．

（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．Data ビュー，Change ビュー，Exchange ビューという仮想的な 3 つのビューが存在すると想定して，問合せを簡単に記述できる．

（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．各ピアに自分自身のデータと履歴情報を Data，Change，From と To の 4 つのリレーションに保存する．

2.3 論理レイヤと物理レイヤのリレーション対応関係

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．

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．

物理レイヤでは，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．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が薄いためである．

（1） **論理レイヤのリレーション**

図 2 の Data[Novel] リレーションは，先に示した図 1 に含まれる各ピアのレコード集合 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．

図 3 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．peer 属性にはそのレ

title	author	peer	id
t1	a1	A	#A011
t5	a5	A	#A028
t7	a7	A	#A040
t7	a6	A	#A041
t8	a8	A	#A055
t1	a1	B	#B032
t2	a2	B	#B035
t1	a1	C	#C005
t6	a6	C	#C038
t5	a5	D	#D001
t1	a1	E	#E010
t3	a3	E	#E088
t1	a1	F	#F017
t0	a0	F	#F021

図 2　ビュー Data[Novel]

コードを所有するピアの情報が記述され，former_id 属性は修正前のレコード ID を，current_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．former_id 属性が空値（–）である場合はレコードの挿入を表し，current_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．

peer	former_id	current_id	time
A	–	#A055	3/17/08
A	–	#A040	4/10/08
A	–	#A011	5/2/08
A	–	#A028	7/18/08
A	#A040	–	8/10/08
A	#A040	#A041	8/10/08
A	#A055	–	9/20/08
B	–	#B032	4/20/08
B	–	#B035	8/26/08
C	–	#C005	1/15/08
C	–	#C038	11/2/08
D	–	#D001	9/10/08
E	–	#E010	7/5/08
E	–	#E088	9/16/08
F	–	#F017	8/12/08
F	–	#F021	8/28/08

図 3　ビュー Change[Novel]

図 4 に示すリレーション Exchange[Novel] はピア間のレコードの交換に関する情報を保持する大域的なビューである．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．

（2） 物理レイヤのリレーション

以下は物理レイヤのリレーションである．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．

図 5, 6 に，図 2, 3 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

from_peer	to_peer	from_id	to_id	time
C	B	#C005	#B032	4/20/08
B	A	#B032	#A011	5/2/08
C	E	#C005	#E010	7/5/08
A	F	#A011	#F017	8/12/08
A	D	#A028	#D001	9/10/08

図 4　ビュー Exchange[Novel]

するタプルのみを抜き出したものが内容となる．なお，図 5 と図 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
t7	a6	#A041

図 5　ピア A の Data[Novel]

former_id	current_id	time
–	#A055	3/17/08
–	#A040	4/10/08
–	#A011	5/2/08
–	#A028	7/18/08
#A040	–	8/10/08
#A040	#A041	8/10/08
#A055	–	9/20/08

図 6　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．具体的には，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る From[Novel] リレーションと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 To[Novel] リレーションに分けられる．図 7 と図 8 はピア A に関する例であり，それぞれピア A が受け取ったレコードに関する情報と自身のレコードをどのピアに渡したかという情報を保持している．

id	from_peer	from_id	time
#A011	B	#B032	5/2/08

図 7　ピア A における From[Novel]

id	to_peer	to_id	time
#A011	F	#F017	8/12/08
#A028	D	#D001	9/10/08

図 8　ピア A の To[Novel]

3. 論理レイヤにおける問合せ

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]．

前節で述べたように，仮想的な 3 つのビューのみを利用して，さまざまな問合せを記述できる．例えば，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探せよ，あるいは，ピア `A` が新たに入手したレコードが，既に保持しているものと重複しているかを調べよなど，などである [5], [7]．以下では問合せの記述例を示す．

問合せ 1：ピア C が保持する著者 a1 により書かれた t1 という小説のレコードをピア A がコピーしたかを調べよ．

```
Reach(P, I1) :- Data[Novel]('t1', 'a1', 'C', I2),
                Exchange[Novel]('C', P, I2, I1, _)
Reach(P, I1) :- Reach(P1, I2),
                Exchange[Novel](P1, P, I2, I1, _)
CheckReach(I) :- Reach('A', I)
Query(I) :- CheckReach(I)
```

次節でこの問合せを処理する異なるアプローチを示し，効率的な問合せ処理方式を検討する．

4. 物理レイヤにおける問合せ処理

問合せを実行する前に，物理レイヤのリレーションを利用する形式に変換する必要がある．変換規則を以下にまとめる．R はリレーション名を表す．@ 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．

R_1: Data[R](attrs, peer, id) ⇒ Data[R]@peer(attrs, id)
R_2: Exchange[R]('from_peer', to_peer, from_id, to_id, time)
⇒ To[R]@'from_peer'(from_id, to_peer, to_id, time)
R_3: Exchange[R](from_peer, to_peer, from_id, to_id, time)
⇒ From[R]@'to_peer'(to_id, from_peer, from_id, time)
R_4: Exchange[R](from_peer, _, from_id, to_id, time)
⇒ To[R]@from_peer(from_id, _, to_id, time)
R'_4: Exchange[R](_, to_peer, from_id, to_id, time)
⇒ From[R]@to_peer(to_id, _, from_id, time)
R_5: Change[R](peer, from_id, to_id, time)
⇒ Change[R]@peer(from_id, to_id, time)

図 9　論理レイヤから物理レイヤへのリレーション変換規則

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．

```
Reach(P, I1) :- Data[Novel]@'C'('t1', 'a1', I2),
                To[Novel]@'C'(I2, P, I1, _)
Reach(P, I1) :- Reach(P1, I2),
                To[Novel]@P1(I2, P, I1, _)
CheckReach(I) :- Reach('A', I)
Query(I) :- CheckReach(I)
```

以下には，この問合せを処理する 2 種類のアプローチを示す．なお，問合せはピア C において発行されたものと考える．

4.1 Seminaive 法による問合せ処理

いわゆる seminaive 法 [2] を用いる方式であり，[5], [6] で詳しく説明した．この問合せについては，ピア C から，レコードがコピーされた経路を辿り，ピア A に辿りつくかを確認することになる．

a） 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：@'C'）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 のルールは再帰を含まないため，ピア C 上で問合せを実行し，結果を一時リレーションに格納すればよい．この結果，Reach には，ピア C の該当するレコードを直接的に渡したピアとそのピアにおけるレコードの ID が入る．

b） ステップ 2：再帰的問合せ処理

ピア上にリレーションが分散していることを考慮して，seminaive 方式による問合せ処理を実行する．例を用いて説明する．

（1） ピア C は問合せを開始する．まず空のリレーション Reach, CheckReach, Query を作成する．すなわち，$\Delta_{\mathtt{Reach}} = \Delta_{\mathtt{CheckReach}} = \Delta_{\mathtt{Query}} = \emptyset$ となる．

（2） ピア C ではステップ 1 の実行により，以下の Reach が得られたとする．

Reach (={(B, #B032),(E, #E010)})

この結果を $\Delta_{\mathtt{Reach}}$ に代入する．また，この場合 $\Delta_{\mathtt{CheckReach}}$ と $\Delta_{\mathtt{Query}}$ 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 C は図 10 の繰り返しプログラムを実行する．実行結果は以下のようになる．,

- Reach $(= \emptyset)$, $\Delta_{\mathtt{Reach}}$ $(= \{(\mathtt{B}, \#\mathtt{B032}), (\mathtt{E}, \#\mathtt{E010})\})$
- CheckReach $(= \emptyset)$, $\Delta_{\mathtt{CheckReach}}$ $(= \emptyset)$
- Query $(= \emptyset)$, $\Delta_{\mathtt{Query}}$ $(= \emptyset)$

```
1. tempReach(P1, I1) :- ΔReach(P1, I2),
                        To[Novel]@P1(I2, P, I1, _)
2. tempCheckReach(I) :- ΔReach('A', I)
3. tempQuery(I) :- ΔCheckReach(I)
4. Reach := Reach ∪ ΔReach
5. CheckReach := CheckReach ∪ ΔCheckReach
6. Query := Query ∪ ΔQuery
7. ΔReach := tempReach - Reach
8. ΔCheckReach := tempCheckReach - CheckReach
9. ΔQuery := tempQuery - Query
```

図 10　繰り返し問合せ処理プログラム 1

（4） 次に，ピア C は，この中間処理結果と問合せプログラムをピア B とピア E にフォワードする．これをピア E での実行結果は

- Reach $(= \{(\mathtt{B}, \#\mathtt{B032}), (\mathtt{E}, \#\mathtt{E010})\})$, $\Delta_{\mathtt{Reach}}$ $(= \emptyset)$
- CheckReach $(= \emptyset)$, $\Delta_{\mathtt{CheckReach}}$ $(= \emptyset)$
- Query $(= \emptyset)$, $\Delta_{\mathtt{Query}}$ $(= \emptyset)$

となり，ここで終了になる．ピア E は結果を逆方向に返す．

一方，ピア B のでの実行結果は，

- Reach $(= \{(\mathtt{B}, \#\mathtt{B032}), (\mathtt{E}, \#\mathtt{E010})\})$,
  $\Delta_{\mathtt{Reach}}$ $(= \{(\mathtt{A}, \#\mathtt{A011})\})$
- CheckReach $(= \emptyset)$, $\Delta_{\mathtt{CheckReach}}$ $(= \emptyset)$
- Query $(= \emptyset)$, $\Delta_{\mathtt{Query}}$ $(= \emptyset)$

になる．

（5） ピア B からピア A にフォワードする．ピア A の実行結果は以下のようになる．

- Reach $(= \{(\mathtt{B}, \#\mathtt{B032}), (\mathtt{E}, \#\mathtt{E010}), (\mathtt{A}, \#\mathtt{A011})\})$,

$\Delta_{\texttt{Reach}}$ (= {(F, #F017)})

- CheckReach (= {(#A011)}), $\Delta_{\texttt{CheckReach}}$ (= ∅)
- Query (= {(#A011)}), $\Delta_{\texttt{Query}}$ (= ∅)

（6） この場合，本来ピア A に達した時点で終了したいが，その先も処理が継続する．ピア A からピア F にフォワードする．ピア F は繰り返しプログラムを実行して，結果は

- Reach (= {(B, #B032), (E, #E010), (A, #A011), (F, #F017)}), $\Delta_{\texttt{Reach}}$ (= ∅)
- CheckReach (= {(#A011)}), $\Delta_{\texttt{CheckReach}}$ (= ∅)
- Query (= {(#A011)}), $\Delta_{\texttt{Query}}$ (= ∅)

になる．$\Delta_{\texttt{Reach}}$ が空集合であるので，ピア F は次にピアにフォワードできず，実行が終了する．この 2 つのブランチの実行結果を逆方向にピア C に返し，問合せが終了する．この問合せの結果として，ピア A がピア C のレコードを ID #A011 のレコードとしてコピーしていたことがわかる．

4.2 マジックセット法による問合せ処理

もう一つの問合せ処理戦略として，マジックセット法 [2] を用いる．先の seminaive 方式ではピア C から問合せが開始されたのに対し，この場合にはピア C からピア A に問合せ処理の依頼が出されることになる．ピア A は，「自分が所有するレコードのうち，ピア C から得られたものがあるか」という問合せを，レコードの入手経路を逆に辿ることにより処理することになる．

上記の問合せに対しては，magic_Reach() を導入し，元の問合せを以下のようになる．

```
Reach(P, I1) :- magic_Reach(P, I1),
                Data[Novel]@'C'('t1', 'a1', I2),
                To[Novel]@'C'(I2, P, I1, _)
Reach(P, I1) :- magic_Reach(P, I1), Reach(P1, I2),
                To[Novel]@P1(I2, P, I1, _)
magic_Reach(P1, I2) :- magic_Reach(P, I1),
                       To[Novel]@P1(I2, P, I1, _)
magic_CheckReach(I) :- magic_Reach('A', I)
Query(I) :- magic_CheckReach(I)
```

変形した問合せはピア A から問合せ処理を開始する．すなわち，ピア A は保存しているレコードのうち，ピア C のレコード (t1, a1) が元になっているレコードがあるかをチェックする．以下のように問合せ処理を行う．

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

まず，第 1 のルールは再帰を含まないため，ローカルな処理を実行して，ピア A が保存しているレコードのうち，他のピアからもらったレコードを取り出す．2 番目のルールからの問合せ処理には seminaive 方式が適用できる．

（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．すなわち，$\Delta_{\texttt{BReach}} = \Delta_{\texttt{FromC}} = \Delta_{\texttt{Query}} = \emptyset$.

（2） ピア A ではローカルな実行により，以下の BReach が得られていたとする．

BReach (={(B, #B032, t1, a1)})

この結果を $\Delta_{\texttt{BReach}}$ に代入する．また，この場合 $\Delta_{\texttt{CheckReach}}$ と $\Delta_{\texttt{Query}}$ 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 は図 11 の繰り返しプログラムを実行する．この結果，

- BReach (= ∅), $\Delta_{\texttt{BReach}}$ (= {(B, #B032, t1, a1)})
- FromC (= ∅), $\Delta_{\texttt{FromC}}$ (= ∅)
- Query (= ∅), $\Delta_{\texttt{Query}}$ (= ∅)

となる．

```
1. temp_BReach(P1, I1, T, A) :- Δ_BReach(P2, I2, T, A),
                                From[Novel]@P2(I2, P1, I1, _)
2. temp_FromC(I) :- Δ_BReach('C', I, 't1', 'a1')
3. temp_Query(I) :- Δ_FromC(I)
4. BReach := BReach ∪ Δ_BReach
5. FromC := FromC ∪ Δ_FromC
6. Query := Query ∪ Δ_Query
7. Δ_BReach := temp_BReach - BReach
8. Δ_FromC := temp_FromC - FromC
9. Δ_Query := temp_Query - Query
```

図 11 繰り返し問合せ処理プログラム 2

（4） 次に，ピア A は，この中間処理結果と問合せプログラムをピア B にフォワードする．ピア B で実行した結果は

- BReach (= {(B, #B032, t1, a1)}), $\Delta_{\texttt{BReach}}$ (= {(C, #C005, t1, a1)})
- FromC (= ∅), $\Delta_{\texttt{FromC}}$ (= ∅)
- Query (= ∅), $\Delta_{\texttt{Query}}$ (= ∅)

となる．

（5） ピア B からピア C にフォワードする．ピア C の実行結果は

- BReach (= {(B, #B032, t1, a1), (C, #C005, t1, a1)}), $\Delta_{\texttt{BReach}}$ (= ∅)
- FromC (= ∅), $\Delta_{\texttt{FromC}}$ (= {(#C005)})
- Query (= ∅), $\Delta_{\texttt{Query}}$ (= ∅)

となる．$\Delta_{\texttt{BReach}}$ が空集合なので，次のピアにフォワードできない．さらに 2 回繰り返し実行して，以下の結果になる．

- BReach (= {(B, #B032, t1, a1), (C, #C005, t1, a1)}), $\Delta_{\texttt{BReach}}$(= ∅)
- FromC (= {(#C005)}), $\Delta_{\texttt{FromC}}$ (= ∅)
- Query (= {(#C005)}), $\Delta_{\texttt{Query}}$ (= ∅)

最後に，ピア A からピア C に結果を返す．

この方法は単純な seminaive 方式と比べ効率的である．前節で述べた前向きの追跡を行うには，ピア C において提供されたレコード (t1, a1) をどのピアがコピーしたかという情報をすべて追跡する必要がある．あるピアが提供したレコードは一般に複数のピアによりコピーされる．ここでは簡単のために n 個

のピアにコピーされるものとする．もしあるピアが適用したレコードが n 個のピアにコピーされ，さらにそのレコードが次に n 個のピアにコピーされるなら，m 回の前向きの追跡の際には n^m 個のピアが関連すると考えられ，かなりの時間がかかると予想される．

一方，マジックセット法に基づく手法では，問合せを後ろ向きに処理する．1 つのレコードのコピー元はただ 1 つのピアに限定されるため，問合せ処理の過程には枝分かれが存在しない．上記の例において，ピア A が N 個のレコードを保存しており，そのうちの N' 個がコピーにより得られたものとする．単純にはこの N' 個のレコードの出所を探ることになる．たとえ N' が大きくとも，指数的にコストが拡大することはない．

実際の処理においては，P2P ネットワークの構成に関する統計データや，ピア A, C が保持する各種情報をもとに，どちらの処理戦略が効率的かを推定して実行する必要がある．

5. 関 連 研 究

近年，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．そのため，データの出所を追跡可能とする *lineage tracing* あるいは *data provenance* が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，情報交換が頻繁になっている P2P ネットワークにおけるデータがどのピアからどのような経路で入手されたか，また，どのピアに複製，修正されたかを追跡することに注目している．

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]．それらの目標は，異種混合・自律的な P2P ネットワーク環境中のデータ統合問題，異種スキーマ，信頼性管理などの問題である．特に，P2P ネットワークにおける協調的なデータ共有を目指す Orchestra プロジェクト [4] は本研究と関連が深い．これに対し本研究では，単純なレコード交換に注目し，より完全なトレースできるフレームワークを構築した．datalog を用いる問合せを P2P ネットワークに実行できる．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．

本研究で提案し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．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．[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，マジックセット法を加え，独自の拡張を行って，問合せの効率化を図る．

6. 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，seminaive 法とマジックセット法によって，論理レイヤの問合せを物理レイヤで効率的に実行するための方式の概略を示した．分散環境での問合せ処理の効率化について議論した．

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り，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．
- SQL データベースによる実現手法：最近のリレーショナルデータベース管理システム（RDBMS）においては，SQL の再帰的問合せ処理機能の支援が進められている．datalog により記述された問合せを SQL で処理可能な形に変換し，各ピア上の RDBMS で実行できれば，進化した RDBMS の問合せ処理能力を効果的に活用できる．
- 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基盤研究（19300027）の助成による．

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] P. Buneman, J. Cheney, W.-C. Tan, and S. Vansummeren. Curated databases. In *Proc. ACM PODS*, 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] F. Li, T. Iida, and Y. Ishikawa. Traceable P2P record exchange: A database-oriented approach. *Frontiers of Computer Science in China*, 2(3):257–267, 2008.
[6] 李峰栄, 飯田卓也, 石川佳治. 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. 情報処理学会研究報告, 2008(56):105–112, 2008.
[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.