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

2007－DBS－143（78）
2007／7／4

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

李 峰栄† 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@db.itc.nagoya-u.ac.jp, ††ishikawa@itc.nagoya-u.ac.jp

あらまし P2Pネットワーク上での情報の交換が盛んになっているが，どのような過程を経てデータがピア間を流通したか，また，流通の過程でデータに対してどのような変更が加えられたかなどを追跡することは容易ではない．本稿では，P2P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ための，データベース技術を基盤とした情報交換システムの構成法を提案する．システムは，ユーザレイヤ，論理レイヤ，および物理レイヤの3層から構成され，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実現する．
キーワード 情報流通，情報交換，トレーサビリティ，P2Pデータベース，データモデル，再帰的問合せ

A Data Model for Traceable Information Distribution on P2P Network

Fengrong LI† and Yoshiharu ISHIKAWA††

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

Abstract Information exchange on P2P networks is very prosperous in recent years. However, it is not easy to trace how data is circulated between peers, and how data modification is performed while the circulation. In this paper, we propose an architecture of an information exchange system on a P2P network. The system is based on the database technologies and realizes the notion of traceability in P2P information exchange. It consists of three layers, the user layer, the logical layer, and the physical layer, and its trace operations are executed as distributed recursive queries between cooperating peers on a P2P network.
Key words information distribution, information exchange, traceability, P2P databases, data model, recursive queries

1. はじめに

データやサービスをネットワーク上の個々のコンピュータ（ピア）で分散管理して，特定のサーバに依存せず柔軟なシステムを構築できる**P2Pネットワーク**（Peer-to-Peer network）は，ファイル交換などにおいて広く利用されている．P2Pネットワーク上の情報交換では，データの複製や変更が，特定のサーバにより集中管理されることなくネットワークの各所で発生する．そのため，他のピアから入手したデータがどのような経緯で手元にあるかという判断が困難となり，データの信頼性を保障することが容易ではなくなる．

このような背景に基づいて，本研究ではP2P環境での情報流通におけるトレーサビリティを実現し，情報の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図る．そのためのアプローチとして，本研究では，リレーショナルデータモデルを基盤とする情報交換システムのアーキテクチャを提案する．システムは，ユーザレイヤ，論理レイヤ，物理レイヤの3層から構成される．P2Pネットワーク上で交換されるレコードの交換・変更の履歴を各ピア上で分散管理し，追跡処理の際には，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．

本稿の構成は以下のようになる．2.で関連研究を述べ，3.で情報交換システムの基本的アイデアについて述べる．4.では論理レイヤについて述べ，5.で物理レイヤのアイデアを示す．最後に6.でまとめと今後の課題を述べる．

2. 関連研究

データベースにおいては，データの加工などの結果得られたデータについて，それが元々どこから出てきたかを知りたいという要求がしばしば発生する．そのため，近年，データの出所を追跡可能とするための研究が進んでいる[6], [8]．また，バイオインフォマティクスなどの科学分野ではネットワーク上での

データベースの共有が盛んであるが，データベースに対して個別に注釈や修正が行われることから，それらの統合や妥当性の検証が必要となっている．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* などと呼ばれるが，本稿ではこれをデータの**系統管理**と呼ぶことにする．

[3] によれば，データの系統管理には，「そのデータがどこから来たか」という観点でとらえる *where-provenance* と「そのデータがなぜ存在するか」という観点でとらえる *why-provenance* の 2 つのアプローチが存在する．本研究は，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，どのピアに複製されたかを追跡するという点で，where-provenance の立場に立つ．系統管理のアプローチの別の分類としては，必要になったときにはじめてデータの出所を求める**遅延処理**のアプローチと，データにその出所に関する情報をあらかじめ付随させておく**先行処理**のアプローチ [2] に分類する見方もある．後者において，データに付随させる情報は，メタデータ，注釈（annotation）などと呼ばれる．本研究は，情報交換の履歴を取得し管理することから，先行処理のアプローチに該当する．また，あるデータベース内部での系統管理を考えているのか，外部から入手したデータに対する系統管理を考えるのかで，*internal lineage* と *external lineage* という分類もある [4]．本研究では，P2P ネットワークの各ピアにおけるデータベースが，外部から入手したデータに対して管理を行うという点では external lineage に該当する．ただし，P2P ネットワーク全体でとらえた場合，系統管理機構を持つ一つの P2P データベースを仮想的に構築するという点で，internal lineage ととらえることもできる．

P2P 環境におけるデータベースに関しては，異種性への対処，問合せ処理，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．

3. P2P ネットワーク上のレコード交換システム

3.1 レコード交換システム

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．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．本研究では，ユーザごとにプライベートなレコード集合が管理されるような状況を想定している．同一マシンを複数ユーザで共有する場合には，そのマシン上でユーザごとにレコード集合を管理し，論理的には別々のピアとして扱う．

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．図 1 に，例としてレコード集合 Novel を示す．この例では，小説のタイトル，著者，言語，発表年度に関する情報を含むレコード集合が，あるピア内に保持されている状況を表している．このピア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．ただし，その内容は同一ではない．

title	author	language	year
Madame Bovary	Gustave Flaubert	French	1857
David Copperfield	Charles Dickens	English	1849

図 1　レコード集合 Novel

この例では，レコードの属性は単純値のみからなるが，ファイル交換を行う場合には (title, author, date, file) といった形式のレコードを用いることが考えられる．この場合，title, author, date はファイルのメタデータにあたる通常の属性であるが，file 属性にはファイルそのものが収められる．file 属性は直接的な検索対象とはならないが，このような機能により，ファイル交換におけるトレーサビリティにも対応することを考えている．

3.2 レコード交換システムの構成

本研究で提案するレコード交換システムは以下の 3 層の構成をとる．

（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．4. で説明する．

（3） **物理レイヤ**：論理レイヤの仮想的なビューを，自律的で分散したピアの協調に基づいて実現する．5. で説明する．

ここでは，ユーザレイヤの機能について簡単に説明する．基本的には，ユーザに対し以下のような機能を提供する．

- **問合せ機能**：「"Charles Dickens" が著者である小説のレコードを検索せよ」といった問合せを実行する．P2P ネットワーク上の他のピアに問合せを転送し，該当するレコード集合を取得する．
- **登録機能**：新たなレコードを，そのピアのシステムに登録する．ユーザが作成したレコードを登録する場合や，問合せにより他のピアから得られたレコードを登録する場合が考えられる．登録されたレコードは系統管理の対象となる．
- **削除・更新機能**：既存のレコードの削除・修正を行う．
- **系統管理の機能**：レコードの出所の追跡など，トレーサビリティに関する処理を実行する．

これらの機能を用いて，ユーザは必要に応じて P2P ネットワーク上に問合せを発行し，関心があるレコードの集合を取得する．ユーザは取得したレコード集合の内容を吟味し，選択したものをシステムに登録する．登録されたレコードは永続的に管理され，系統管理の対象に置かれる．このような機能により，ユーザは自身のレコード集合の内容をより充実させ，その価値の向上を図ることができる．一方で，このシステムでは，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．それぞれのピアは自律的に自身のデータを管理しており，他のピアとの内容の同期は特に行わないものとする．

本稿では，ユーザレイヤの詳細については触れず，その下位に位置する論理レイヤ，物理レイヤの実現方式について主に議論する．ユーザレイヤの実現方式に関する検討は今後の課題としたい．

3.3 求められる系統管理の機能

前節で述べたレコード交換方式は，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的であるが，単純な実装では，たとえば以下のような問題が発生しうる．

（1） **レコードの出所が明らかでない**：ピア A のレコード集合中に存在するレコードが，ピア A 自身が作成したものか，他のピアから取得されたものかが明確でない．また，そのレコードが，たとえばピア B から取得されたということが記録されていたとしても，そのレコードを元々作成したのがどのピアであるかが明らかでない．着目するレコードがどこで作られ，どのようなピアを経由したかが定かでないため，信頼性が不十分となり，他のピアと情報を共有する利点をうまく生かせない．

（2） **レコードの重複を検出できない**：たとえば，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が明確でない．同じサイトから同一のレコードを複数回取得することは無駄であり，避けるべきである．しかし，P2P ネットワークでは，同じレコードが別の経路をたどって到達することも考えられることから，判定は容易ではない．

（3） **レコードの行き先が明らかでない**：たとえば，ピア A が，自身が提供したレコードに誤りを発見したとする．その誤りを訂正したいが，どのピアがそのレコードを保持しているかを追跡できなければ，訂正することは容易ではない．訂正する前のレコードの内容を用いて P2P ネットワーク上に問合せを発行すれば，同じ値のレコードを探索することは可能であるが，それが実際にピア A により提供されたものかは分からない．

（4） **更新処理を追跡できない**：たとえば，ピア A からピア B に複製されたレコードに対してピア B が修正を行ったとする．さらに，ピア B が修正を行ったそのレコードをピア C が複製したとする．ここで，ピア C が，入手したレコードの出所を知りたいとする．この場合，P2P ネットワーク上に問合せを行うだけではレコードの出所は分からない．レコードをどのピアが作成し，どのような経緯で得られたかだけでなく，どのピアで更新処理が生じたかも把握する必要がある．別の例として，ピア A が提供したレコードがピア B に複製された後，ピア A がそのレコードを自身のレコード集合から削除してしまったとする．この場合，後の時点において，ピア B が入手したレコードが元々どのピアから得られたものであるかを知りたくなったとしても，単に P2P ネットワーク上に問合せを発行するだけでは，元のレコードを発見することはできない．

以降では，これらの問題に対処するための，系統管理のためのシステム実現手法を示す．

4. 論理レイヤの実現手法

4.1 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，系統管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

title	author
t1	a1

ピア B

title	author
t1	a1
t2	a3

ピア C

title	author
t1	a1

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，系統管理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
t1	a1	B	#B032
t2	a2	B	#B040
t2	a3	B	#B051
t1	a1	C	#C005

図 3 ビュー Data[Novel]

from_id	to_id	time
–	#A011	5/2/07
–	#B032	4/10/07
–	#B040	4/20/07
#B040	#B051	6/10/07
–	#C005	3/20/07

図 4 ビュー Change[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，系統管理のために必要となる情報を表現する．

図 5 に示すリレーション Exchange[Novel] 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 B がピア A 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]

4.2 論理レイヤにおける問合せ処理

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処理について述べる．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 [7] 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 [5]．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) AND
                Exchange[Novel](P, 'A', I1, I2, _)
BReach(P1, I1) ← BReach(P2, I2) AND
                 Exchange[Novel](P1, P2, I1, I2, _)
Origin(P) ← BReach(P, I) AND NOT Exchange[Novel](_, P, _, I)
```

第 1, 第 2 のルールにより，リレーション BReach を定義している．このリレーションは “Backward Reachable” の意味であり，該当する小説のレコードをピア A が入手するまでに経由したピアの情報を集積する．BReach(P, I) と書いたとき，P は経由したピアを表し，I はそのピアにおけるレコードの ID を表す．第 1 のルールでは，Data[Novel] リレーションと Exchange[Novel] リレーションの情報をもとに，ピア A に情報を直接渡したピアの情報が求められる．第 2 のルールは再帰的な処理であり，グラフの探索により到達可能なノードをすべて列挙する処理に相当する．この結果，ピア A に情報をもたらすのに関与したすべてのピアの情報が BReach に含まれることになる．第 3 のルールは，それらのピアの中で終端にあるものを選択するためのものである．ピア A から到達可能で，Exchange[Novel] において to_peer, to_id に対応するエントリがないことが条件となる．

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) AND
                       Exchange[Novel](P, 'A', I1, I2, _)
BReach2(P1, I1, T, A) ← BReach2(P2, I2, T, A) AND
                        Exchange[Novel](P1, P2, I1, I2, _)
ViaB(T, A) ← BReach2('B', _, T, A)
```

ここでは，例 1 で定義した BReach と似た構造を持つ 2 つのルールにより，BReach2 を定義している．先の BReach との違いは，小説の名前と著者名の情報を保持することであり，BReach2(P, I, T, A) は，ピア A が持つタイトル T，著者名 A の本のレコードが，ピア P では ID I で表されていたことを表している．3 つ目のルールで，ピア B が仲介したという制約を入れて，条件を満たす T, A の組を抜き出す．

問合せ 3：レコード交換においては，過去に他のピアから得たレコードを再び入手することがある．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たい．得たレコードが以前に入手したものと値として同一かどうかは，ピア A のレコード集合の内容と比較すれば簡単に判明する．しかし，値として同一であっても，異なるピアにより生成された可能性もあるため，出所が同じであるかを判定したい．ここでは，ピア A が，ピア D から ID が “#D051” であり値が (t1, a1) であるレコードを受け取ったとする．そのレコードの出所が，ピア A が保持する同じ内容のレコード（ID は “#A011”）の出所と同じであるかを知りたいとする．

```
BReachA(P, I1) ← Data[Novel]('t1', 'a1', 'A', '#A011')
                 AND Exchange[Novel](P, 'A', I1, I2, _)
BReachA(P1, I1) ← BReachA(P2, I2) AND
                  Exchange[Novel](P1, P2, I1, I2, _)
BReachD(P, I1) ← Data[Novel]('t1', 'a1', 'D', '#D051')
                 AND Exchange[Novel](P, 'D', I1, I2, _)
BReachD(P1, I1) ← BReachD(P2, I2) AND
                  Exchange[Novel](P1, P2, I1, I2, _)
Dup ← BReachA(_, I) AND BReachD(_, I)
```

1, 2 番目のルールでピア A からの追跡情報を集め，3, 4 番目のルールでピア D からの追跡情報を集める．5 番目のルールは，A, D から追跡した結果の中に，流通の過程において同じレコード ID を持つものがあるかどうかを調べるものである．このルールが充足された場合には，出所が同じといえる．

問合せ 4：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，それが最新の情報であるかを知りたい．すなわち，ピア A が複製したレコードについて，その後になって誰かが変更を行ったかを知りたい．たとえば，該当するレコードをピア A がピア B からコピーしたとする．その後でピア B がそのレコードを自身のピア上で修正したとする．ここでは，どのような修正があったかを知りたいとする．また，さらにピア B がそのレコードをピア C から取得した場合，C がどのように修正したかについても検出したい．この場合には以下のような問合せを発行する．

```
BReach3(P, I1) ← Data[Novel]('t1', 'a1', 'A', I2) AND
                 Exchange[Novel](P, 'A', I1, I2, _)
BReach3(P1, I1) ← BReach3(P2, I2) AND
                  Exchange[Novel](P1, P2, I1, I2, _)
Modified(P, T, A) ← BReach3(P, I1) AND
                    Change[Novel](P, I1, I2, _) AND I2 != NULL
```

BReach3 は，先に示した BReach とまったく同じである．第 3 のルールでは，ピア A の該当するレコードを入手した経路上にあるピア P において ID I1 のレコードに修正があったことを検出する．ルールの第 2 項と第 3 項において，ピア P の ID I1 を持つレコードが ID I2 を持つレコードに修正されたことをチェックする．I2 が空値でないという制約は，レコード I1 が削除されたのではなく，修正されたことを意図している．

なお，実際には，その後にも引き続き修正が発生する可能性があるが，上の問合せはそこまでは対応していない．そのような修正結果も調べたい場合にはさらにルールを追加する必要がある．

5. 物理レイヤの実現方法

5.1 物理レイヤの考え方

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，以下の理由により，P2P ネットワーク上の情報を常時統合しておくことは適切ではない．

- P2P ネットワーク上のそれぞれのピアは自律的に振舞い，全体として緩やかな連携を実現している．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．
- 情報を追跡する処理は頻繁に生じるものではなく，必要が発生した時点で実行されるのが一般的である．統合されたデータベースを常に維持管理するためには，P2P ネットワーク上のデータベースの間での緊密な通信処理が必要となり，オーバヘッドが大となる．また，データを集中管理するサーバのピアに多大な負荷が発生し，ボトルネックの原因となる．

このような理由から，本研究では，P2P ネットワーク上の各ピアは，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．論理レイヤにおける問合せは，物理レイヤにおいてはピア間の協調的処理により処理される．

5.2 物理レイヤのリレーション

先に示した論理レイヤのリレーションの実現方法について述べる．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

図 6 ピア A の Data[Novel]

from_id	to_id	time
–	#A011	5/2/07

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．

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．

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

図 9 ピア B の To[Novel]

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．なお，ピア B からピア A に上記のようなレコードの複製が行われる際の処理は次のようになる．

（1） ピア A からピア B に対して問合せを行い，ピア A がピア B のレコード集合を入手する．

（2） ピア A は入手したレコード集合の中から，実際にピア A に格納するレコードを選択する．そのようなレコードの 1 つが，ID が#B032 であるようなピア B のレコードである．

（3） ピア A はそのレコードに#A011 という ID を付与して Data[Novel] に追加する．また，複製したという情報を From[Novel] に追加する．

（4） ピア A は，このような更新情報（#B032 を#A011 として，タイムスタンプ 5/02/07 の時刻に格納した）をピア B に送信する．

（5） その情報を受信したピア B は，自身の To[Novel] にその情報を格納する．

このような処理を行う理由は，ピア B は，ピア A から情報を受け取らないと，ピア A がピア B から得たレコード集合のうち実際どれを格納したかが分からないためである．

5.3 問合せ処理のアプローチ

論理レイヤでは，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．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，物理レイヤの構成に応じて変換して実行する必要がある．ここではそのアイデアを簡単に説明する．詳細な手法の開発については今後の研究課題としたい．

先に示した問合せ 1 について再び考える．この問合せは，ピア A が保持するタイトル t1，著者名 a1 のレコードの作成元ピアを特定するものであった．

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

この論理レイヤの問合せがピア A において発行されたものとする．次のように処理を進める．

（1） まず，ピア A 内で処理可能なローカルなルールを見つける．この例の場合，第 1 のルールはピア A の情報のみを用いるものであるため，これが該当する．このルールを以下のように書き換え実行する．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.

```
BReach(P, I1) ← Data[Novel]('t1', 'a1', I2) AND
                From[Novel](I2, P, I1, _)
```

（2） ピアAは，第2のルールを処理するために，P2Pネットワーク上に問合せを発行する．具体的には，ステップ1でピアAが作成したBReach(P, I)に含まれる各ピアPに対し，以下の論理レベルの問合せをそのピアで実行し，結果を返してもらうように要求を出す.

```
BReach(P1, I1) ← BReach(P2, I2) AND
                 Exchange[Novel](P1, P2, I1, I2, _)
Origin(P) ← BReach(P, I) AND NOT Exchange[Novel](_, P, _, I)
```

ただし，これらの問合せを送付すると同時に，ピアA上でローカルに求められたBReachリレーションの情報も一緒に送付する．たとえば，ピアAのBReachリレーションが図10のようだったとする(注1).

P	I
B	#B032
C	#C077
B	#B093

図10 ピアAのBReachリレーション

このとき，ピアAはピアBとピアCに対して問合せを発行するが，ピアBに渡すデータは，属性Pの値がBであるようなレコード（リレーショナル代数で書けば σ[P='B'](BReach)）となる．ピアCについても同様に，一部のレコードのみ送信する.

（3） ステップ2において論理レイヤの問合せ（および問合せに付随するデータ）を受け取った各ピアは，物理レイヤの問合せに変換して実行する．問合せは再帰的にP2Pネットワーク上で伝播し実行される．その結果として，ステップ2の例についていえば，ピアAはピアB, Cより問合せ結果であるOriginのレコードを受け取る.

（4） 最終的に，ピアAは各ピアからのOriginのレコードの和集合をとり，問合せ結果として返す.

この例では，論理レイヤの問合せのうち，第1のルールはピアAのみで評価されるローカルなルールであり，P2Pネットワークを探索する処理の初期設定に該当する．一方，第2のルールは問合せをネットワーク上で処理するための実行経路を得るために用いられる．第3のルールは問合せ結果を得る具体的な処理に対応するものであり，問合せが実行される各ピア上で実行される.

6. まとめと今後の課題

本稿では，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の概念について述べ，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．本研究では，トレーサビリティを実現するための具体的なアーキテクチャについて議論し，ユーザレイヤ，論理レイヤ，物理レイヤの3階層からなる構成を示した．論理レイヤと物理レイヤについては，リレーショナルデータモデルを基盤とした具体的な実現方式について述べた.

今後の課題については，以下が挙げられる.

- 論理レイヤにおける問合せを物理レイヤの問合せに変換するための，問合せの書き換え手法の開発
- 物理レイヤにおける問合せ処理を効率化する最適化手法の開発：Datalogにより記述された問合せを効率的に実行するアプローチについては，これまでさまざまな研究がなされているが，与えられた問合せを書き換えることによる効率化が基礎となる．本研究においても，問合せの効率的な処理方式の開発を図りたいと考えている.
- 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.
- ログ情報の取得手法：RDBMSにおいて，取得されたタプルの格納，修正，削除に関するログ情報を管理し，先に示したData[Novel]などのリレーションを具体的に構成する手法について開発する．最近のRDBMSではデータベース監査などの機能が導入されていることから，それらの機能を有効活用する.
- 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），セコム科学技術振興財団，および，柏森情報科学振興財団の助成による.

文　献

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

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

[3] 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.

[4] A. Halevy, M. Franklin, and D. Maier. Principles of dataspace systems. In *Proc. ACM PODS*, pp. 1–9, 2006.

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

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

[7] J. D. Ullman and J. Widom. *A First Course in Database Systems*. Prentice Hall, 2nd edition, 2002.

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

(注1)：第2，第3のタプルは図6と図8からは出てこないが，説明の便宜上含めている.