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

2008－DBS－146 (49)
2008／9／23

トレーサブルなP2Pレコード交換システムPI-REXの設計

飯田 卓也† 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
〒464–8601 名古屋市千種区不老町
E-mail: †{iida,lifr}@db.itc.nagoya-u.ac.jp, ††ishikawa@itc.nagoya-u.ac.jp

あらまし P2Pネットワーク上の情報交換において，データの信頼性の確保などの目的で，情報の流れを追跡したいという要求がしばしば生じる．本稿では，P2Pネットワーク上の柔軟な情報交換においてトレース処理を実現するためのシステムPI-REXの設計について述べる．P2Pネットワーク上で協調するピア間において再帰的にデータベース問合せを分散実行するためのシステムの機能，アーキテクチャ，および実装手法について論じる．
キーワード レコード交換，P2Pデータベース，トレーサビリティ，再帰的問合せ，系統管理

Design of PI-REX: A Traceable P2P Record Exchange System

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

† Graduate School of Information Science, Nagoya University
†† Information Technology Center, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464–8601 Japan
E-mail: †{iida,lifr}@db.itc.nagoya-u.ac.jp, ††ishikawa@itc.nagoya-u.ac.jp

Abstract To assure the reliability of exchanged data in peer-to-peer (P2P) networks, we often find the situation in which we want to trace the flow of information. In this paper, we present the design of PI-REX: a P2P record exchange system that supports trace facilities. The system executes distributed recursive queries among cooperating peers in P2P networks. We discuss the system's features, architecture, and implementation ideas.
Key words redord exchange, P2P databases, traceability, recursive queries, lineage tracing

1. まえがき

インターネットのブロードバンド化に伴い，ネットワークを有効利用できる**Peer-to-Peer (P2P)** ネットワークが注目を集めている．P2Pネットワークでは，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）で分散管理でき，特定のサーバに依存しない柔軟なシステムを構築できる．高い自律性・耐故障性を備える一方，P2Pで取得された情報は，情報発信者自身から入手したものではないために情報の信頼性が問題となることがある．そのため，P2Pネットワークにおける情報の流れを逆方向に追跡することが時として必要となる．

このような背景から，本研究グループでは，P2P環境において情報の流通・交換を行うことを想定し，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を提案している[7],[8]．本研究におけるレコード交換（record exchange）とは，P2Pネットワーク上のピアが共通のスキーマに基づくタプル構造のレコードを交換することを意味している．各ピアは，それぞれ独自にレコード集合を管理するが，他のピアに対し自身のレコードを提供したり，逆に他のピアからレコードを取得し自身のレコード集合に追加してもよい．各ピアが自律的にレコード集合を保持しつつ，他のピアとの交換を許すという柔軟性が特徴となっている．

本システムでは，自身のレコード集合について各ピアが個別に修正を行うことも可能とし，他のピアから得た情報をカスタマイズするような形での情報共有も可能としている．このような緩やかな情報共有の形態は，たとえばバイオインフォマティクスなどの科学技術分野における情報共有において求められている．これらの分野では，関連する研究グループ間での迅速な情報交換・流通が求められる一方，個々の研究グループの独自性も必要とされており，P2P技術が有望とされている[6]．

本稿では，現在開発中であるシステム**PI-REX**(注1)の概要について述べ，その設計方針について議論する．特に，システムの概念，アーキテクチャ，及びトレース処理の実現方法について述べる．

(注1)：P2P-based Implementation of Record EXchange services

本稿の構成は以下のようになる．2節では 関連研究について述べる．3節ではシステムの特徴および機能について，その概要を述べる．4節では，システムのアーキテクチャについて説明する．5節では，問合せの記述方式と処理方式について述べ，6節でまとめと今後の課題について述べる．

2. 関 連 研 究

P2P 環境におけるデータベースに関しては，異種のスキーマ間のマッピングや，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]．本研究の内容は，P2P ネットワーク上の情報統合に関する研究と関連が深い[5]．しかし本研究では，交換される情報は単一のスキーマに基づくレコード集合であると簡略化し，情報の異種性の問題については考慮せず，トレーサビリティの概念の支援のための技術開発を中心に据えている．本システムの特徴は，レコード交換システムの実現のために内部的にデータベースシステム技術を活用する点にある．ただし，システムを利用するユーザ自体は，必ずしも下位のデータベースシステムを意識する必要はない．システム実現の上でデータベース技術を活用することで，問合せの効率化やスケーラビリティを実現することを目指している．

近年，データの出所を追跡する，*lineage tracing* あるいは *data provenance* に関する研究が盛んに進められている[3], [4], [10]本研究グループでは，これを総称してデータの**系統管理**と呼んでいる．対象分野の一つはデータウェアハウスであり，さまざまな操作により加工されたデータウェアハウスのデータに対し，元の基幹データベースのデータを遡及的に特定し，データの信頼性を保証することがなされている．もう一つの応用領域としては，バイオインフォマティクスなどの科学技術分野におけるデータ共有が挙げられる．遺伝子に関する実験結果などのデータベースを分散した研究組織において共有することが一般的になってきているが，そのようなデータは後々になって誤りが見つかるなど，修正が加えられたり注釈がなされたりする．その種の応用では，情報の出所を追跡する機能は重要となっている．本システムでは，これらの研究と同様，信頼性のある情報の交換のための枠組みを提供しようとしている点を特徴としている．ただし，情報の出所を追跡するのみでなく，あるピアが提供した情報がどのピアによりコピーされたかなど，情報の行先についての追跡を可能とする点にも特徴がある．

本研究と関連が深い研究としては，Ives らによる ORCHESTRA プロジェクトが挙げられる[6](注2)．このシステムは，P2P 環境において異種のスキーマを持つデータベースを緩やかに統合することを目指している．情報の異種性などさまざまな問題を扱っているが，系統管理に関する機能も一部有している．

また，Hellerstein らのグループによる *declarative networking* のプロジェクトも関連が深い[9](注3)．センサネットワークなどのネットワーク環境における問合せをコンパクトに記述し実行するため，再帰的な問合せ能力を持つ **datalog** [2] を活用している点に特徴がある．本システムにおいても，問合せの記述において datalog を用いている．目的や対象となる情報が異なるが，問合せの実行処理方式においては共通する部分も多い．

本システムでは，筆者らの研究グループにより提案されているトレーサビリティを有する P2P レコード交換機構のための問合せ処理技術[7], [8] の実装を行う．問合せ処理のみならず，ユーザへ提供する各種機能の実現や，システムの有用性を高めるための技術など，さまざま要素技術を実現する必要がある．

(注2)：http://www.cis.upenn.edu/~zives/orchestra/
(注3)：http://p2.berkeley.intel-research.net/

3. システムの概要

3.1 特　　徴

本システムが既存の P2P 情報交換システムと異なるのは，その情報がどこからきたかなどの流通経路を追跡できるトレーサビリティ機能を有すること，データベース技術を基盤としていること，レコード構造の情報を交換することなどが挙げられる．レコードの構造は，ネットワークで大域ていに共有されるスキーマに従う．ユーザは，検索，更新などのレコード交換システムの基本操作の他に，トレース処理を実行することができる．

3.2 機　　能

本システムにおけるレコードとは，属性と値からなるタプル構造のデータを意味しており，そのスキーマは P2P ネットワーク上で共有されている．想定する応用は，たとえば科学技術データの交換であり，P2P ネットワークを用いて，自律的でありながら信頼性の高い情報交換のネットワークを作り上げたいという組織が参加することを想定している．そのため，従来の P2P 情報交換システムで強調されているような匿名性の保持については特に配慮しない．また，ピアは自身の振る舞いに対してある程度責任を持つことを想定しており，P2P ネットワークから突然に離脱するようなことも考えない．

以下では簡単な例として，図 1 に示すレコード集合 Novel を考える．これは，あるピア A において保持されているレコード集合を示している．

タイトル	著者	ジャンル	発表年
Murder On The Orient Express	Agatha Christie	mystery	1934
And Then There Were None	Agatha Christie	mystery	1939
Foundation	Isaac Asimof	SF	1951
The Starry Rift	James Tiptree Jr.	SF	1985

図 1　ピア A のレコード集合 Novel

P2P ネットワークに参加したユーザ（ピア）は，興味のあるレコードについての情報を検索，閲覧し，自分の気に入ったものがあれば，ローカルなレコード集合に追加することができる．ユーザはローカルなレコード集合に対し，レコードの追加・削除・更新を行うことができる．また，検索されたレコード，登録されているレコードに関してレコード作成者や変更履歴などのより詳細な情報を得たいときは，それに対応したトレース処理を指示することもできる．

本システムの機能を具体例とともに述べる．コマンド形式のユーザインタフェースが提供されているものとし，実行例も含めて説明する．

- **検索機能**：レコードの属性に対し指定された条件をもと

に，該当するレコードを P2P ネットワーク上のピアから探しだし表示する．たとえば，

```
> search title = 't1' and author = 'a1'
```

では，タイトルが't1'，著者が'a1' であるレコードを検索する．結果として，$\langle record, record\_id, peer\_id \rangle$ という形式のタプルの集合が得られる．*record* はレコード本体，*record_id* はその ID，*peer_id* はピアの ID である(注4)．検索機能については，既存の P2P 検索技術を活用する．たとえば，分散ハッシュテーブル（DHT）の技術などを用いることで，検索処理を効率的に行うことが考えられる．

- **登録機能**：検索の結果得られたレコードや，ユーザ自身が作成したレコードをローカルなレコード集合に登録する．たとえば

```
> regiter B #B013
```

では，ピア B（B はピア ID を意図している）のレコード ID'#B013' のレコードが，自身のレコード集合に登録される．その結果として，成功もしくは失敗が返され，成功した場合，登録されたレコードはトレース処理の対象となる（逆にいえば，単に検索を行っただけでは，その結果のレコードはトレースの対象とはならない）．

上記のような登録がたとえばピア A から指示されたときには，ピア A からピア B に対してレコードを登録する旨，通知が行われ，ピア B 側でも登録処理が行われる．これは，ピア B において，「該当するレコードがピア A によりコピーされた」という記録をするためであり，これによりピア B からの情報の追跡を可能とする．ピア A とピア B で保持される情報を整合した情報に保つため，上記の処理は分散トランザクションとして実行される．

- **修正・削除機能**：ローカルなレコード集合内のレコードについて修正・削除を行うことができる．たとえば

```
> update #A021 {author='a3'}
> delete #A022
```

では，レコード ID が'#A021' のレコードの著者名を'a3' に変更し，レコード ID '#A022' のレコードを削除している．なお，修正・削除の際，過去のデータが完全に削除されることはなく，履歴情報がピア内のデータベースに保持される．これはトレーサビリティ機能の実現のために必要となる．

- **トレース機能**：トレーサビリティを実現する，本システムの最も特徴的な機能であり，レコードの出所や入手経路に関する問合せを行うことができる．たとえば，

```
> origin {title='t1'}
```

では，ローカルなレコード集合内でタイトルが't1' であるレコードについて，もともとの作成者を行ったピアを求める．この結果としてピア ID（条件にマッチするレコードが複数ある場合にはピア ID の集合）が返る．このような問合せが与えられたとき，P2P ネットワーク上で再帰的問合せを実行し，各ピアに保持されている履歴情報を基にそれに答える．この機能については後で詳しく述べる．

以上の例ではコマンド形式のユーザインタフェースを示したが，GUI のサポートなども考えられる．

3.3 システム機能に関する検討

本システムを構築する上で特に以下のような点に検討が必要である．

3.3.1 問合せ言語によるトレース処理の記述

トレース処理には，履歴情報を利用した様々なものが考えられる．いくつかの基本的なトレース機能はあらかじめシステムに登録されており，ユーザは自由に利用することができるものとする．しかし，ユーザの要求によって使う機能は異なり，それらの機能を前もって全て実装するのは困難であり，無駄でもある．よって，必要な機能をデータベースシステム管理者（DBA）が自由に追加できるようにする必要がある．ただし，トレース処理は，分散した各ピアが所有する履歴データを連携させることで初めて処理できることから，一般のユーザはもちろん，DBA にとっても記述は容易ではない．

そのため，本システムでは，トレース処理を記述するための問合せ言語を提供する．トレース機能は，問合せ言語を用いて，一種のストアドプロシージャとしてコマンド登録できるものとする．問合せ言語は datalog を拡張したものであり，DBA により一般には記述・登録されるが，もちろんある程度の知識を持つユーザであれば，アドホックな問合せを直接記述し実行することも可能である．分散した情報源を個別に考慮して問合せを書くことは容易ではないため，本システムでは，P2P ネットワーク上のレコード集合に関するすべての情報を仮想的に保持する一種のビューを DBA に対して提供する．このビューに対して問合せを書くことで，DBA は困難な分散問合せ処理を簡潔に記述できる．

3.3.2 システム管理者のための機能

上述のとおり，システム管理者は，トレース処理に関するコマンドを，問合せ言語を用いて記述し，システムに登録することが可能である．また，システム管理者に対しては，P2P レコード交換ネットワークに参加し退出するための**参加・退出機能**も支援される．参加の際には，レコード交換ネットワークにすでに参加しているピアにコンタクトをとることで参加を行う．その際，接続先のピアからスキーマ情報をコピーすることで，共通のスキーマで情報を共有することを可能とする．退出の場合には，ユーザには直接的な負担はないが，システム内部ではトレーサビリティを確保するための後始末が必要となる．トレース機能の維持のため，近隣のピアに履歴情報をコピーした後，ネットワークを離脱する．また，離脱するピアの肩代わりをするピアを特定できるよう，ピア ID が指定されたときに，実際にどのピアがその処理を受けるかというマッピング情報の管理も必要となる．

(注4)：ピア ID がわかれば，そのピアに対して直接的にコンタクトをとることが可能であると想定する．また，レコード ID は，そのレコードを保持するピア内において一意であるとする．

4. PI-REX のアーキテクチャ

4.1 アーキテクチャの概要

本節ではトレーサブルな P2P レコード交換システム PI-REX のアーキテクチャについて述べる．まず，本システムは大きく三階層に分けられる [7], [8]．ユーザに対するインタフェースとしての機能を提供するユーザレイヤ，P2P ネットワーク上に分散したデータを統合し，仮想的なビューを構成する論理レイヤ，実際に各ピアが保持しているデータベース上のリレーションを表す物理レイヤの三階層である．このモデルに基づいたシステムの構成を図 2 に示す．

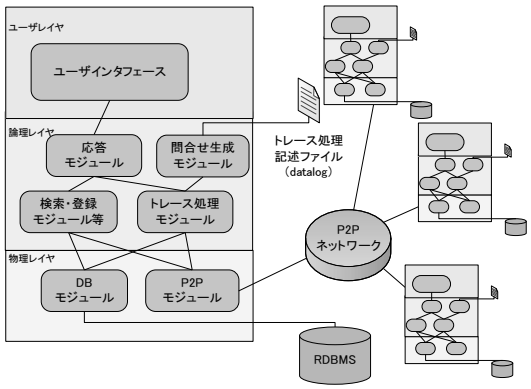

図 2　システムの構成

各モジュールの概要について説明する．

- **ユーザインタフェース**：ユーザがレコード集合を操作・閲覧するための機能を提供する．前節で述べたコマンドベースのインタフェースなどがこれに該当する．
- **応答モジュール**：ユーザインタフェースからのリクエストに対する応答，及び，他ピアからの呼び出しに対し，要求を解析し他のモジュールに作業を依頼する，制御機能を司る．
- **問合せ生成モジュール**：トレース処理が指示されたとき，対応する問合せの実行処理を行う．与えられた条件をもとに問合せを変換し，問合せプランを生成する．
- **トレース処理モジュール**：トレース処理の問合せプランを実行する．必要に応じて下位のデータベースへのアクセスや，P2P ネットワークを介した他ピアとの通信を行う．
- **検索・登録モジュール等**：ユーザに提供する各種機能を，データベースや P2P ネットワークの連携により実現する．
- **データベースモジュール/P2P モジュール**：データベースモジュールは，下位のリレーショナルデータベースへのアクセスを提供する．P2P モジュールは，P2P ネットワークによる通信機能を提供する．

4.2 レコード管理

レコード集合の RDBMS での管理方式について述べる．なお，詳細は論文 [7], [8] で述べられている．

ここでは，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 3 は，それぞれピア A, B, C が保持している小説（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
t5	a5

ピア B

title	author
t1	a1
t2	a3

ピア C

title	author
t1	a1

図 3　各ピアにおけるレコード集合 Novel

4.2.1 Data リレーション

各ピアの持つレコード集合は，そのレコード集合ごとにひとつの Data リレーションとして管理される．各レコードには，主キーとしてレコード ID（RID）が割り当てられる．なお，Data リレーションでは，ユーザが削除，変更したデータも，実際には削除されずに保持している．図 3 のピア A のレコード集合 Novel のリレーションである Data[Novel] を図 4 に示す．Data[Novel] という表記は，レコード集合 Novel の Data リレーションであることを表している．

id	title	author
#A011	t1	a1
#A012	t3	a3
#A013	t4	a4
#A014	t5	a2

図 4　ピア A の Data[Novel] リレーション

Data リレーションのスキーマ情報はレコード集合の種類によって異なるものが使われる．また，スキーマ情報はピア間で共有され，レコード交換の際に利用される．

4.2.2 履歴情報の管理と利用

トレース処理においては，どのピアからどのピアにレコードがコピーされたかといった履歴情報の利用が必要となる．本システムでは，ピアが交換をおこなった際，データを最初に作成したピアから受け取ったピアまでのすべての履歴を全て記録していく方式はとらず，自分が交換したピアの情報のみ記録する方式をとる．これにより，履歴情報の管理が軽量になる．トレース処理に関する多様な問合せは，ピア間に分散した履歴情報を連携することで実現する．たとえば，基本的な問合せとしては，データの作成者を求める問合せや，逆に自分のファイルをコピーしていったピアを求める問合せが考えられる．さらに，自身が提供したレコードに対して，後で修正を行った際，コピー先に更新処理を通知するというような利用法も可能となる．

そこで，各ピアのリレーショナルデータベースシステムでは，レコード情報本体の他に，トレース処理の実現に必要となる履歴情報を管理する．以下の 3 つのリレーションがある．

（1） **Change** リレーション：Change リレーションは，レコードの作成，修正，削除の履歴を管理している（図 5）．レコードの変更をおこなった際には，変更されたレコードには新しく RID が割り当てられる．そのため，Change リレーションでは，変更前の RID と変更後の RID，変更時の時刻を記録する．なお，新規にレコードを作成した際には，変更前の RID の値は空値となる．削除の場合には変更後の RID を空値とする．

（2） **From** リレーション：From リレーションは，他ピア

id	from_id	time
#A012	–	08/1/25
#A013	–	08/2/5
#A014	#A012	08/2/8
–	#A013	08/2/10

図 5　ピア A の Change[Novel] リレーション

から自身のピアへレコードをコピーした際の履歴を管理している (図 6). 自身のデータベース (DB) におけるレコードの RID, コピー元のピア, コピー元の RID, コピー時の時刻を記録する.

（3） **To** リレーション：To リレーションは，自身のピアから他ピアへレコードがコピーされた際の履歴を管理している (図 7). 自身の DB におけるレコードの RID，コピー先のピア, コピー先の RID，コピー時の時刻を記録する.

トレース処理をおこなう際，From リレーションはレコードの複製元を追跡する処理に利用され，To リレーションはレコードの複製先を追跡する処理に利用される.

id	from_peer	from_id	time
#A011	B	#B032	08/2/1

図 6　ピア A の From[Novel] リレーション

id	to_peer	to_id	time
#B032	A	#A011	08/2/1

図 7　ピア B の To[Novel] リレーション

4.2.3　論理レイヤでの表現

上記の 4 つのリレーションは各ピア上で分散して保持されるが，P2P ネットワーク上での問合せを記述する場合には，分散したリレーションを直接指定して問合せを記述することは容易ではない．そこで，上位の**論理レイヤ**を設ける．論理レイヤでは，各ピアで管理されているリレーションを統合化した，仮想的なビューを提供する．これにより，トレース処理のための問合せ記述を容易にする．具体的には，Data および Change リレーションには属性 peer が追加され統合化される．To, From リレーションは一つに統合され Exchange ビューとなる（図 8, 9, 10）.

peer	id	title	author
A	#A011	t1	a1
A	#A012	t3	a3
A	#A013	t4	a4
A	#A014	t5	a2
B	#B032	t1	a1
B	#B034	t2	a2
B	#B035	t2	a3
C	#C005	t1	a1

図 8　Data[Novel] ビュー

peer	id	from_id	time
A	#A012	–	08/1/25
A	#A013	–	08/2/5
A	#A014	#A012	08/2/8
A	–	#A013	08/2/10
B	#B034	–	08/1/30
B	#B035	#B034	08/2/7
C	#C005	–	08/1/21

図 9　Change[Novel] ビュー

from_peer	to_peer	from_id	to_id	time
B	A	#B032	#A011	08/1/25
C	B	#C005	#B032	08/1/24

図 10　Exchange[Novel] ビュー

4.3　トレース処理の実行

トレース処理は，P2P ネットワーク上に分散した Data, From, To, Change リレーションを組み合わせることで，ピア上で再帰的に実行される．各ピアは，自身のデータベースの履歴情報を元に問合せを実行し，次に処理を転送するピアを特定し，再帰的に問合せを発行する．最終的に得られた結果は問合せと同様の経路を通り，問合せ元のピアへと返される.

トレース処理がおこなわれるとき，システム内部では，図 11 のような処理が実行される．トレースコマンドに対応する問合せ記述とユーザから与えられたパラメータが問合せ生成モジュールへと渡され，解析され物理レイヤ上で実効可能な問合せプランへと変換される．トレース処理モジュールは，与えられた問合せプランを問合せを実行する．トレース処理モジュールは，自ピアからの問合せのみならず，他ピアから依頼された問合せについても実行処理を行う.

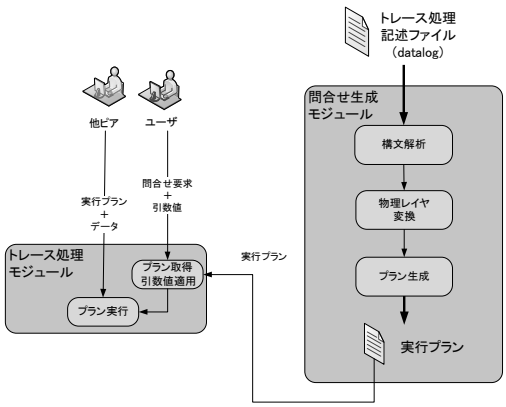

図 11　トレース処理の仕組み

5.　問合せの記述と処理

本システムでは，再帰的な問合せ能力を有する datalog を用いて，トレース処理の記述を可能とする [11]. 論理レイヤにおける統合されたビューを対象とすることで，問合せを記述する DBA やユーザは，データが分散していることを意識せずに問合せを記述できる．問合せ記述や実行処理方式については, [7], [8] を参照いただきたい．ここでは，問合せの記述例と，PI-REX 独自の拡張部分について述べる.

5.1　Datalog による問合せ記述

Datalog による問合せの記述について，例をあげ説明する.

［例 1］ **問合せ 1** として

> ピア A が持っているレコード Novel の中で，タイトルが ‘t1’ であり，著者が ‘a1’ であるようなレコードの作成者を求めよ

を考える．この問合せを datalog によって記述すると，以下のようになる.

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

この問合せは，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 4 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．

論理レイヤ上のビューを用いて記述された datalog の問合せは，構文解析の後，実際のデータベース上のリレーションを使った物理レイヤ上の問合せに変換され，最終的には，個々のピア内の RDBMS に対する SQL 問合せに変換される [7], [8].

5.2 記述法の拡張

上記の問合せの記述例は，特定の定数に対応したものであるため，汎用的なものではない．よって，datalog による問合せを拡張し，スキーマ情報やレコード情報を抽象化した記述を可能にすることで，スキーマに非依存なより一般化した問合せを記述可能にする．例えば，問合せ 1 をより汎用的に拡張した問合せとして，以下の**問合せ 2** を考える．

> ピア$my_pname が持っているレコード$rec_name の中で，属性$attr_list に一致するようなレコードの作成者を求めよ

$から始まるものは実行プラン生成時に具体化される変数を表している．この問合せの記述例が以下であり，テンプレートとしてシステムに登録される．

```
$rec_name = $ARGV[0]
$attr_list = $ARGV[1]
$my_pname = get_mypeer()
$attr_list = check_attr_list(attr_list,$rec_name)
BReach(P, I1) :- Data[$rec_name]($my_pname, I2,$attr_list),
                 Exchange[$rec_name](P, $my_pname, I1, I2, _).
BReach(P1, I1) :- BReach(P2, I2),
                  Exchange[$rec_name](P1, P2, I1, I2, _).
Origin(P) :- BReach(P, I),
             NOT Exchange[$rec_name](_, P, _, I).
Query(P) :- Origin(P).
```

まず実行時に引数を渡すため$ARGV を定義している．get_mypeer 関数のように，自分のピア名を取得したり，スキーマ情報と各属性値との妥当性をチェックしたりするなどの，問合せ記述を支援するための関数群が提供されるものとする．

6. 議論と今後の課題

本稿では，トレーサビリティを実現する P2P レコードシステム PI-REX の機能，及び設計について述べた．トレース処理を実現するためのアーキテクチャを示し，スキーマやレコードの値に依存しない，汎用的な問合せ記述方法を示した．トレース処理の記述方法については，条件文の導入などによるスクリプト言語的な更なる拡張が考えられるが，記述法が複雑になり過ぎないような配慮が必要である．現在，この設計に基づくシステムの実装を進めている．今後は，評価のためのシミュレータの実装や，問合せ処理の最適化についても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 (19300027) の助成による．

文　献

[1] K. Aberer and P. Cudre-Mauroux. Semantic overlay networks. In *Proc. VLDB*, 2005. (tutorial notes).
[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
[3] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In *Proc. VLDB*, pp. 953–964, 2006.
[4] P. Buneman, J. Cheney, W.-C. Tan, and S. Vansummeren. Curated databases. In *Proc. ACM PODS*, 2008.
[5] A. Y. Halevy, Z. G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The piazza peer data management system. *IEEE TKDE*, 16(7):787–798, 2004.
[6] Z. Ives, N. Khandelwal, A. Kapur, and M. Cakir. ORCHESTRA: Rapid, collaborative sharing of dynamic data. In *Proc. CIDR*, pp. 107–118, 2005.
[7] F. Li, T. Iida, and Y. Ishikawa. Traceable P2P record exchange: A database-oriented approach. *Frontiers of Computer Science in China*, (3):257–267, Sept. 2008.
[8] F. Li and Y. Ishikawa. Traceable P2P record exchange based on database technologies. In *Proc. APWeb*, pp. 660–671, 2008.
[9] 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.
[10] W.-C. Tan. Research problems in data provenance. *IEEE Data Eng. Bull.*, 27(4):45–52, 2004.
[11] 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 (DEWS2008), 2008.
[12] 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 (データベースシステム・情報学基礎). 情報処理学会研究報告, 2008(56):105–112, 2008/6/19・20.