

トレーサブルな P2P レコード交換システムにおける イベント駆動型問合せ

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

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。我々の研究グループでは、P2P ネットワーク上でトレーサブルな情報交換をおこなうシステム PI-REX を開発している。本稿では本システムにおけるイベント駆動型問合せについて述べる。イベント駆動型の問合せの導入により、ピアの行動監視、更新情報の監視などをネットワークに負荷をかけることなくおこなうことが可能となる。また、システムのイベント駆動型問合せの機能、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、トレーサビリティ、P2P データベース、イベント駆動問合せ

Event-driven Queries for 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

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. We are developing PI-REX system, a P2P record exchange system that supports trace facilities. In this paper, we present the feature of its event-driven queries. Using event-driven queries, we can monitor updates and exchanges of information without heavy network load. We discuss the outline of the feature and implementation ideas.

Key words record exchange, traceability, P2P databases, event-driven queries

1. はじめに

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 PI-REX^(注1) を提案・開発している [9], [10], [15]。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

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

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 [7]．

本システムにおけるトレーサビリティの実現方法は，トレース処理は頻繁におこなわれないという想定のもと，最低限の情報のみ保持し，トレースをおこなう時にその都度ネットワーク上に問合せを発行し，必要な情報を収集する戦略をとるというものである．これは，近年着目されている‘pay-as-you-go’型の情報統合の考え方 [6] に基づいている．しかし，特定のレコードの更新状況や，特定のレコードの他ピアへの複製状況を監視したいというような要求を本システムのトレース手法により実現すると，レコードを保持する全ピアへの定期的な問合せが必要となり，ネットワークに多大な負荷がかかるという問題があった．これらの問合せを効率化するため，本稿では PI-REX におけるイベント駆動型問合せを考える．本システムにおけるイベント駆動型問合せの導入により，ネットワーク上への頻繁な問合せの発行を減らし，イベントが発生したときのみ通信をおこなうことで，ネットワークの負荷を減らしリアルタイムにネットワーク上のレコード更新を知ることができる．また，柔軟な問合せを実現する記述法と枠組みを導入し，システムを拡張する．本稿では，本システムにおけるイベント駆動型問合せの概要と実現方法について議論する．

本稿の構成は以下になる．2 節では 関連研究について述べる．3 節では PI-REX の特徴および機能について，その概要を述べる．4 節では，イベント駆動型問合せについて説明し，本システムにおける問合せについて述べる．5 節では，PI-REX におけるイベント駆動型問合せの実行について述べ，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]．本システム PI-REX は，P2P ネットワーク上の情報統合に関する研究 [5] と関連が深い．しかし本システムでは，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 [2], [3], [13]．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．

また，本稿では，PI-REX の枠組みを用いたイベント駆動型の問合せを提案する．P2P ネットワークを用いたイベント駆動型システムの研究については，publish/subscribe(pub/sub) [12], [14] や，連続的問合せ [4], [11]，ECA ルール [8] といった研究が盛んであり，本研究との関連性も高い．

pub/sub は，publisher が送信した情報を subscriber に通知するためのシステムである．pub/sub が不特定多数の subscriber に効率的に情報を送信することを目指しているのに対し，本システムにおけるイベント駆動型問合せは，特定のレコードに対する問合せや，特定ピアに対する問合せを個人が自由に設定することができるという点で異なる．

連続的問合せは，イベントをトリガとして問合せを実行し更新を通知するシステムである．既存の P2P における連続的問合せシステムである PeerCQ [4] は，各ピアが協調して動き，全体で一つの連続的問合せを実現するものである．本システムの問合せは，P2P 上の各ピア同士がお互いの行動に関する連続的問合せを実現するシステムであるという点でその目的が異なる．

ECA ルールは，イベント (Event)，コンディション (Condition)，アクション (Action) の 3 つの部分によって構成される動作記述言語であり，アクティブデータベースで用いられている．本システムにおいても ECA ルールを用いた記述法を提供している．

3. PI-REX の概要

PI-REX システムは P2P ネットワーク上の情報交換において，トレーサビリティを実現し，情報の信頼性を確保する [10]．ここでいう情報の信頼性とは，データの出所や，入手経路などの伝達経路を特定することができ，それにより情報の重複を検出したり，情報が最新の情報であることを保証できることを意味する．詳細については [9], [10], [15], [16] を参照されたい．

3.1 システムの概要

PI-REX は，ユーザが保持しているレコード集合をユーザ間で交換し，交換されたレコードの交換履歴を保持することでトレースを可能とするトレーサブルな P2P レコード交換システムである．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．

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 Asimov	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．

3.2 トレース情報の管理

PI-REX では，リレーショナルデータベース管理システム (RDBMS) を用いて各レコード集合を管理する．また，トレース処理を可能とするためのアプローチとして，各ピアで自身の修正履歴・交換履歴を蓄積，管理し，必要に応じて各ピアへ問合せをおこなう手法をとる．具体的には，各ピアではレコード集合を Data リレーションにより管理し，履歴情報として Change リレーション，From リレーション，To リレーションを管理する．Data リレーション レコード集合を管理．各レコードにはレコード ID(rid) が付与される．

Change リレーション ユーザによるレコードの作成・修正・削除履歴を管理する．

From リレーション レコードの直接の交換元情報を管理する．
To リレーション レコードの直接の交換先情報を管理する．

図 2 にピア A における各リレーションの一例を示す．

(a) Data[Novel] リレーション				(b) Change[Novel] リレーション			
rid	title	author		rid	from_id	time	
#A011	t1	a1		#A012	—	08/1/25	
#A012	t3	a3		#A013	—	08/2/5	
#A013	t4	a4		#A014	#A012	08/2/8	
#A014	t5	a2		—	#A013	08/2/10	

(c) From[Novel] リレーション				(d) To[Novel] リレーション			
rid	from_peer	from_id	time	rid	to_peer	to_id	time
#A011	B	#B032	08/2/1	#A011	C	#C041	08/2/1

図 2 ピア A が保持するリレーションの例

図 2 (a) より，#A011 のレコードは title が t1, author が a1 のレコードであることがわかる．図 2 (c) より，ピア A はこのレコードをピア B から得たこと，そしてそれはピア B のレコード #B032 のコピーであるとわかる．図 2 (d) より，ピア A はこのレコードをピア C に渡しており，ピア C の #C041 というレコードにコピーされたことがわかる．また，図 2 (b) より，#A012 と #A013 はピア A 自身が作成し，#A012 は修正され #A014 に変更され，#A013 は削除されていることがわかる．

3.3 トレース処理の仕組み

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である From, To リレーションより，自分が直接交換をおこなった相手を知ることができるため，その情報を元に問合せを実行・転送し，転送先ピアにおいて再帰的に問合せを実行することにより，トレース処理を実現することができる．

4. イベント駆動型問合せ

イベント駆動型問合せは，データベースに対して問合せを発行することで結果を得るものとは異なり，問合せがシステムに永続的に配置され，設定されたイベントが起きた際に実行される問合せである．イベント駆動型問合せを PI-REX に適用し，新しいフレームワークを導入することで，より柔軟で効率的な処理を可能とする．本節では，PI-REX システムにおけるイベント駆動型問合せにより実現できる機能とその問合せ記述方式について述べる．

4.1 目的

PI-REX では，トレース処理によって様々な機能が実現でき，レコードの更新チェック機能もそのひとつである．しかし，更新を随時確認したい場合，PI-REX のトレース処理による実現方法ではネットワークに多大な負荷がかかることになる．図 3 に，ピア A が作成したレコード rec1 が各ピアへ複製されていき，あるとき，ピア H にてレコード rec1 が rec1' に修正された例を示す．

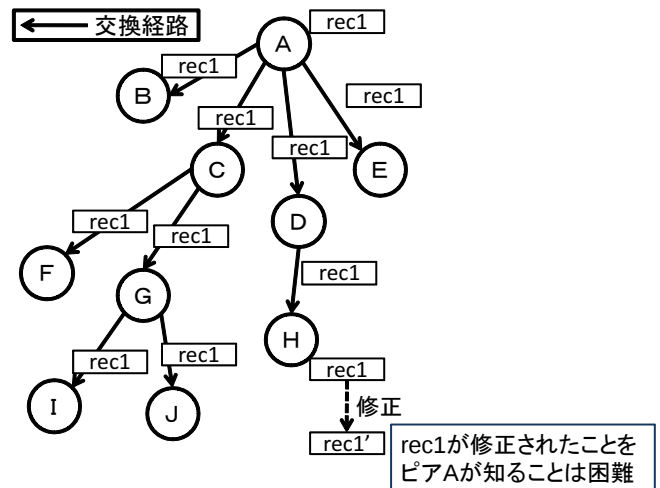


図 3 レコードの複製例

rec1 は，それぞれの経路を通り，ピア A～J に保持されている．その後，ピア H が rec1 を修正しレコード rec1' を作成した．トレース処理を用いてピア A からネットワーク上の他のピア（ピア H）がレコード rec1 を修正したことを知るためには，保持されているレコード複製時の経路を通じて rec1 を保持する全ピア（ピア B～J）に定期的に問合せをおこなう必要がある．しかし，同一レコードを持つ全ピアへの頻繁なトレース処理の実行は，ネットワークに多大な負荷がかかる．定期的な問合せ

の間隔を長くすれば、負荷を軽減することは可能だが、その場合リアルタイム性が問題となる。

この問題は、PI-REX の既存の枠組みでは解決することが困難であると考えられる。よって、このような問合せに対し、ネットワークの負荷をおさえ、リアルタイムに情報を得るための PI-REX における新しい枠組みとして、イベント駆動型問合せを考える。

4.2 PI-REX におけるイベント駆動型問合せ

PI-REX では、もともとレコードが頻繁に修正されるようなレコード集合を扱うことを想定している。そのため、システムに要求される重要な機能として、特定のレコードの更新状況や、特定のレコードの他ピアへの複製状況、特定のピアの行動などの監視機能が考えられる。

一方で、PI-REX におけるトレース処理方式は、'pay-as-you-go' の概念を基礎とし、トレースに必要な他ピアの情報は必要最低限のみ保持し、トレース処理が必要になったときに各ピアに問合せを発行し、必要な情報を収集しながら実行する手法をとる。これはトレース処理が頻繁に発生するものではないということ想定しており、トレース処理を実現するためにレコード交換時におこなわれる履歴情報の共有同期処理などのランニングコストを極力少なくするためである。しかし、更新状況の監視といった、いつ変更されるかわからない情報、リアルタイムで追跡したい情報を現在のトレース処理によって得るためには、レコードを保持する全ピアへの定期的な問合せが必要となり、ネットワークの負荷が大きくなる。特にリアルタイム性が必要となる場合は、頻繁に問合せを発行し確認する必要がある、ネットワークに多大な負荷がかかってしまう。

これらの問合せに対し、問合せ先のピアにあらかじめ配置され、イベント発生時に自動で実行・通知されるようなイベント駆動型の問合せシステムを導入することにより、パフォーマンスの向上、負荷の軽減を実現する(図4)。また、イベント駆動型問合せをそのピア自身に配置することで、重複レコードのチェックなどの処理を自動化することができる。

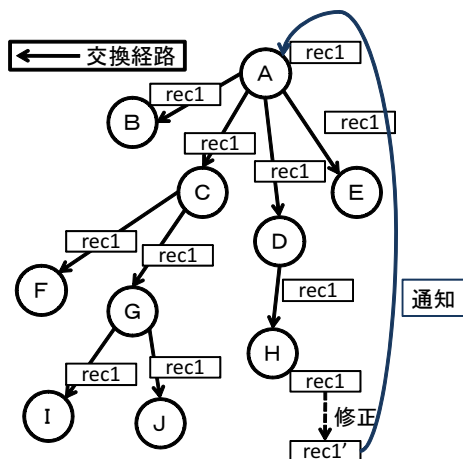


図4 イベント駆動型問合せのイメージ

ユーザは、PI-REX システム上に問合せを配置し、特定のイベントを監視することで、自分にとって有用な情報を継続的に

得ることができる。また、登録時の重複レコードの検出などの自動的に実行可能な処理を自身のピアにイベント駆動型問合せとして設定することにより、利便性を向上することができる。

PI-REX におけるイベント駆動型の問合せは、その配置方法により、レコード配置型とピア配置型の2種類に大きく分けられる。

4.3 レコード配置型問合せ

レコード配置型の問合せは、特定のレコードに配置され、イベントの発生により条件がチェックされ、実行される問合せである。レコード配置型の問合せでは、問合せはレコード交換の際にレコード情報に付与される形で、レコード情報と共にコピーされる(図5)。つまり、レコード配置型の問合せは、ネットワーク上の同一レコードすべてに配置される。



図5 レコード配置型問合せ

レコードがコピーされた時、問合せも同時にコピーされる

問合せが配置されたレコードは、そのレコードを保持するピアにより監視され、条件を満たしたら配置したピアに通知される。

例として、ピアAが以下のようなレコード配置型問合せを配置する場合を考える。これらの問合せはネットワーク上で複製されたレコード全てに配置される。

[例1] レコード配置型問合せ

(1) レコード#A012の属性authorが更新されたら、更新内容をピアAに通知する。

(2) レコード#A012が交換されたら、コピー元のピアとレコードid、コピー先のピアとレコードidをピアAに通知する。

(3) レコード#A012が交換されたら、コピー元のピアとレコードid、コピー先のピアとレコードidをピアAに記録する。

問合せ(1)は、最も典型的で単純なレコード配置型問合せの例である。#A012と同一のレコードを持つ任意のピアがそのレコードのauthor属性を修正した場合、そのピアは、更新通知をピアAに送信する。

問合せ(2)は、自分の作成したレコードの交換をリアルタイムに捕捉するための問合せである。どのピアがどのピアからレコードをコピーしたか、という情報を交換が発生した際にピアAに通知する。

問合せ(3)は、問合せ(2)で得られた情報に対して、その情報をデータベースに蓄積する処理が加わったものである。これらの情報は問合せ(2)で送信されるピアAへの通知に対する問合せをピアA自身にピア配置型問合せとして配置する。これらの履歴を蓄積することは、本システムにおけるトレース処理の考え方である'pay-as-you-go'の考え方に反するものである。そのため、このような問合せを多数のレコードに配置することは推奨できない。しかし、トレース処理を頻繁におこなうよう

なレコードの履歴情報を記録することでトレース処理時のコストの削減をおこなうことができる。

4.4 ピア配置型問合せ

ピア配置型の問合せは、特定のピアに配置され、そのピアの行動をイベントとし、ピアの有するレコード集合に対する問合せをおこなう。自分以外の興味あるピアに配置したり、自分自身に配置して、システムを拡張したりすることなどが考えられる。

例として、ピア A が以下のようなピア配置型問合せを配置する場合を考える。

[例 2] ピア配置型問合せ

(1) ピア B のレコード集合において、author が‘Agatha Christie’であるレコードが登録されたらピア A に通知する。

(2) ピア A がレコードを登録する際に、そのレコードが既に登録されていないか自動的にレコードの重複をチェックする。

(3) ピア B のレコード集合において、author が‘Agatha Christie’で自分が持っていないレコードが登録されたらピア A に通知する。

(4) 過去に #A012 を修正したことのあるピアのレコード集合において、レコードの更新が発生したらピア A に通知する。

問合せ (1) は、ピア B に配置され、ピア B のレコード集合に author が‘Agatha Christie’であるレコードが登録されたら、ピア A に通知される。

問合せ (2) は、ピア A が自身に配置する問合せで、レコードを登録するときに、自動的に重複チェックをしてくれるような問合せである。重複チェックは、トレース処理を用いることで実現できる。

問合せ (3) は、ピア B に配置され、ピア B のレコード集合を監視する。実際の実装では、ピア A の持っていないレコードのみ通知するという条件はピア B 側では考慮されず、ピア B 側では、(1) と同様の問合せが実行され、ピア A 側でフィルタリングにより重複が除去されるが、ユーザはそのことを意識せずに問合せを記述することが可能である。

問合せ (4) は、問合せの配置をトレース処理を用いておこなう例である。トレース処理により、#A012 を保持しているピアをトレースし、その各ピアに対し問合せを配置する。しかし、このように配置にトレース処理を用いた場合、問合せが配置されるピアは、配置する時点で #A012 を保持しているピアに限られ、問合せ配置後に #A012 が他のピアへコピーされた場合、それを検出し、問合せを配置することはできない。配置後に #A012 をコピーしたピアにも自動的にこの問合せを配置したい場合は、ピア配置型問合せではなく、レコード配置型問合せを用いればよい。

4.5 ECA ルールによる問合せ記述

イベント駆動型問合せの記述には、アクティブデータベースにおいて用いられている ECA ルールを本システムに適するように拡張したものを用いる。ECA ルールは、イベント (Event)、コンディション (Condition)、アクション (Action) の 3 つの部分によって構成される動作記述言語である。イベントは環境内で発生する事象、コンディションはルールを実行するための制

約条件、アクションは実行する操作 (問合せ) を表す。

本システムでは、複数のピアが問合せを実行し、複数のピアへ結果を返すことが想定される。また、問合せを配置するピアとその結果を受け取るピアが一致しないような問合せを記述したいことがある。問合せ結果の通知先ピアを記述するため、ECA ルールを結果の送信先が記述できるよう MESSAGE 節を追加する。従って、ECA ルールは以下のような形式になる。

ECA ルール記述形式

```
RULE rulename
  ON event
  IF condition
  DO action
  MESSAGE message TO peers
```

また、本システムにおいて各ピアで発生するイベントとして、主に表 1 のようなものが考えられる。

イベント	内容
REGISTER(rid)	レコードの登録
UPDATE(rid, pre_rid, change_attrs)	レコードの更新
EXCHANGE(from_rid, to_peer, to_rid)	レコードの交換
DELETE(rid)	レコードの削除
NOTIFICATION(rule_name, peer, messages)	通知

表 1 イベント

これらのイベントが発生した際、各イベントに関する情報として以下のようなものが利用可能である。

REGISTER

- rid: 登録されたレコードの rid

UPDATE

- rid: 更新されたレコードの rid
- pre_rid: 更新される前のレコードの rid
- change_attrs: 変更された属性

EXCHANGE

- from_rid: 交換されたレコードの rid
- to_peer: 交換した先のピア
- to_rid: 交換した先のレコードの rid

DELETE

- rid: 削除したレコードの rid

NOTIFICATION

- rule_name: 通知を送ったルールの名前
- peer: 通知を送ったピア
- messages: 送られてきたメッセージ

4.5.1 問合せの記述例

以上のような記述形式に従って、例 1 の問合せ (1) の記述例を ECA ルール記述例 1 に示す。レコード配置型は、配置されたレコードに関するイベントに対して発生する。そのため、問合せ記述の IF 節にレコード ID などの条件を追加して明示的にレコードを特定する必要はない。

ECA ルール記述例 1

レコード #A012 の属性 author が更新されたら更新内容をピア A に通知する。

```
DEPLOY(RECORD, update_notification)
RULE update_notification
  ON UPDATE
  IF updateAttr('author')
  MESSAGE event_info TO A
```

まず 1 行目はレコード配置型の問合せとして update_notification というルールをレコードに配置する処理である。これを記述することで、ひとつの問合せとして複数のルールを記述し、柔軟に配置することができる。2 行目以降でルール update_notification を記述している。更新イベント発生時 (ON 節)、その更新属性が 'author' であれば (IF 節)、イベント内容をピア A に通知する (MESSAGE 節)。

問合せ (2) の記述例を ECA ルール記述例 2 に示す。EXCHANGE イベントを捕捉し、問合せ (1) と同様、イベントにより得られた情報をピア A に通知する。

ECA ルール記述例 2

レコード #A012 が交換されたら、コピー元のピアとレコード id、コピー先のピアとレコード id をピア A に通知する。

```
DEPLOY(RECORD, exchange_notification)
RULE exchange_notification
  ON EXCHANGE
  MESSAGE event_info TO A
```

問合せ (3) の記述例を ECA ルール記述例 3 に示す。問合せ (2) の記述に加え、自身のピアで記録をおこなうという問合せ exchange_storing の記述が必要となる。exchange_storing はピア配置型の問合せで、ピア A 自身に配置される。

ECA ルール記述例 3

レコード #A012 が交換されたら、コピー元のピアとレコード id、コピー先のピアとレコード id をピア A に記録する。

```
DEPLOY(RECORD, exchange_notification)
DEPLOY(A, exchange_storing)
RULE exchange_notification
  ...

RULE exchange_storing
  ON NOTIFICATION
  IF NAME='exchange_notification'
  DO register_exchange(from_peer, from_id, to_peer, to_id)
```

exchange_storing では、他ピアからの通知を NOTIFICATION イベントとして捕捉し、得られた交換情報を記録する。DO 節に記述されている register_exchange は、システムにあらかじ

め登録されているストアドプロシージャである。コンディション部 (IF 節) やアクション部 (DO 節) では、これらのストアドプロシージャを利用して処理を記述することができる。

5. イベント駆動型問合せの実装

前節では、イベント駆動型問合せを本システムに導入した際に可能となる機能や利点、問合せ記述方式について述べた。本節では、システムにイベント駆動型問合せを実装する際の、問合せの配置方法及び実行方法について述べる。また、実行の最適化について述べる。

5.1 問合せ実行までの流れ

最初に、ユーザがイベント駆動型問合せを記述し、最終的に結果を取得するまでの概略を示す。

(1) 問合せの配置

- ユーザが ECA ルール (4.5 節) で問合せを記述する。
- それぞれの問合せの配置方法 (5.3 節) に従い、ネットワーク上のピアに問合せが転送される。
- 問合せを受け取ったピアで問合せが解析、登録され、活性化される (5.4 節)。

(2) 問合せの実行判定・実行 (5.5 節)

- ピア内でイベントが発生し、該当するイベントに対する問合せをチェックする。

- 該当する問合せの実行条件をチェックする。
- 条件を満たしていれば、問合せが実行される。
- 問合せに記述された送信先ピアへ結果を送信する。

(3) 通知メッセージの受け取り

- メッセージの表示など、適切な処理をおこなう。
- 以降では、これらの処理について詳しく述べる。

5.2 システムの構成

図 6 に各ピアで実行されるシステムの構成を示す。

問合せ実行用のコンポーネント群では、Event-driven Query Manager が他のコンポーネントを制御する。Event Detector、Condition Evaluation Manager において、それぞれイベントと条件が判定され実行されるべき問合せが抽出される。Query Evaluator で問合せが実行され、Notification Sender によって各ピアに通知される。

配置用コンポーネント群では、Deployment Manager が問合せの配置方法を管理する。Deployment Manager は問合せの配置手順を解析し、各ピアへ転送する。また、各ピアからの問合せを受け取り登録するために Query Registration Manager へ渡す。Query Registration Manager では、問合せの登録・管理をおこなう。

5.3 問合せの配置

ECA ルールにより記述された問合せは Deployment Manager で解析され、各ピアに転送・配置される。ここでは、レコード配置型とピア配置型の各ピアへの問合せの配置方法について述べる。

イベント駆動型の問合せは主に、他ピアから情報を得るためのものであり、ECA ルールによりイベント駆動型の問合せを記述した後、問合せ先のピアに問合せを転送し、配置・実行し

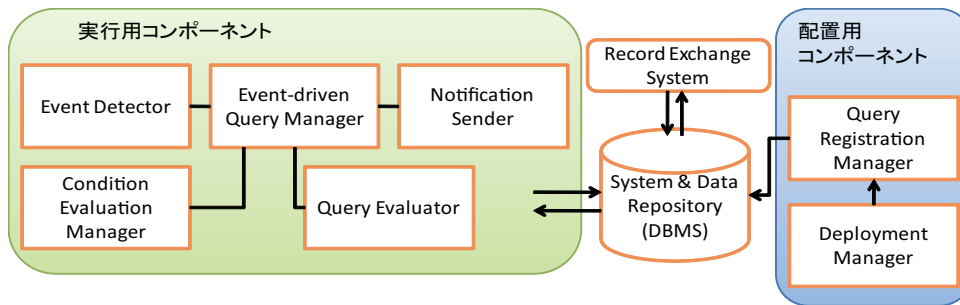


図 6 システム構成

てもらう必要がある。各ピアは自身の問合せを他ピアに実行してもらったり他ピアからの問合せを実行するというように、互いに協調して動く。

5.3.1 レコード配置型問合せの配置

レコード配置型問合せでは、ネットワーク上にあるすべての同一レコードに対して、問合せを配置する必要がある。問合せを配置する際、問合せを配置するレコードが、自分が作成したレコードである場合と、他人から複製したレコードである場合の2通り考えられる。自分が作成したレコードである場合は、基本的に作成時に問合せを配置し、交換時に共に複製されていくことで、自動的にネットワーク上のすべての同一レコードに問合せが配置される。他人から複製したレコードに問合せを配置する場合は、トレース処理を用いて、作成者までさかのぼり、更にそのレコードを持つ全ピアへ問合せを転送し、配置する必要がある。

例として、ピア A が以下の二つの問合せを配置する場合を考える。

(1) 私が提供したレコード #A012 に変更があれば通知してほしい。

(2) 私が持っているレコード #A011 に変更があれば通知してほしい。

問合せ (1) と (2) は、本質的に同じ問合せであるが、その配置方法が異なる。

問合せ (1) はピア A が #A012 のレコードを作成した際に、レコードに付与される。問合せが付与されたレコードが他ピアにコピーされる際に、レコードと共に問合せも共にコピーされ、ネットワーク上の同一レコード全てに問合せが自動的に配置される。その様子を図 7 に示す。

一方、問合せ (2) は、ピア A がレコードを他ピアから取得した時点で、同様のレコードがネットワーク上に多数存在することが考えられるため、それらのレコード全てに問合せを付与する必要がある。ピア A は、まず、トレース処理をおこなうことで、#A011 の作成者であるピア B を見つけ出し、ピア B の持つ #A011 の複製元のレコードに問合せを配置する。その後、ピア B は履歴を基に、そのレコードを複製していったピアすべてに、同様の問合せを配置する。さらに、各ピアでそれを繰り返すことにより、ネットワーク上の #A011 と同一のレコード全てに問合せが配置される。その様子を図 8 に示す。図の矢印で示されたような交換経路で交換されたレコードに対し、ピア A

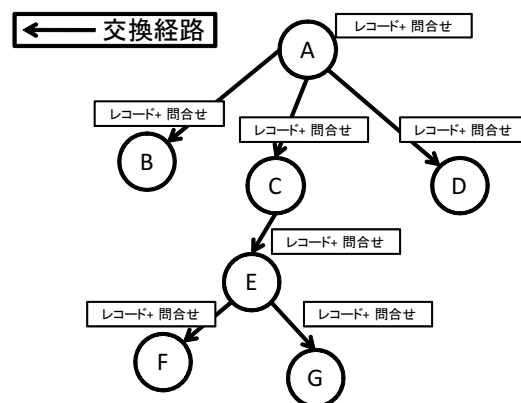


図 7 レコード作成者の問合せ配置方法

から点線矢印で示された配置経路を通じて問合せが付与される。

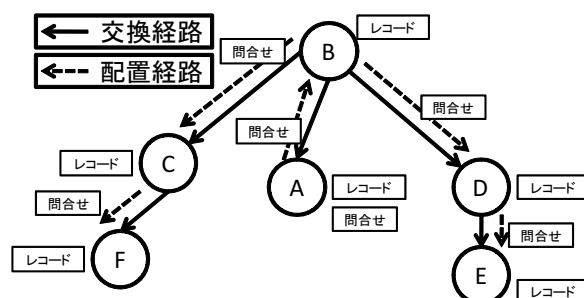


図 8 作成者以外の問合せ配置方法

5.3.2 ピア配置型問合せの配置

ピア配置型問合せでは、レコードの検索時に配置する方法や、直接ピアを指定して配置する方法の他、レコード配置型問合せと同様にトレース処理を用いて配置する方法などが考えられる。

5.4 問合せの登録

問合せが配置されたレコードを登録した際や、5.3 節で述べたような方法で他ピアから問合せの配置メッセージが来た際、Query Registration Manager により、問合せが登録される。ECA ルールで記述された各ルールは、システムによって解析され、RDBMS に以下のような Rule リレーションとして登録される。

Rule(rule_id, event, condition, action, message)

ECA ルールをリレーショナル形式で格納することにより、発生したイベントに該当するルールや、条件判定などを容易に実

現することができる。その他登録された時間や、登録したピア、レコード配置型の場合は問合せと問合せが付与されているレコードとのマッピング情報などが記録される。

5.5 問合せの実行

システム内でイベントが発生すると、システムによって、イベント・条件の判別がおこなわれ、実行されるべき問合せが抽出・実行される。

イベントの判別 RDBMS へ問合せすることで該当するルールを抽出する。

条件の判定 RDBMS より抽出したルールの条件を発生したイベントの情報と順番に照らし合わせることで条件を満たしたルールを取り出す。

アクションの実行 自身のデータベースに対しアクションを実行し、結果を格納する。

メッセージの送信 イベントにより得られた情報やアクションの実行により得られた情報を設定されたピアへ送信する。

6. まとめと今後の課題

本稿では、トレーサビリティを有する P2P レコード交換システム PI-REX における、イベント駆動型問合せについて述べた。これにより、更新の通知などの処理を効率的に実現できることを示した。また、イベント駆動型問合せを導入する際の、問合せの記述方法や、システム上での実現方法について述べた。

今回示した問合せ方法では、レコード配置型問合せでは同一レコードを持つすべてのピアに問合せを配置する必要があった。しかし、複数のピアが同様の問合せを配置することを考えた場合、すべてのピアに配置せずとも、同じ問合せを配置しているピアを利用することで、そこから転送・通知を受けるような問合せの配置が可能である。今後はより効率的な問合せの配置方法や問合せ実行時の最適化について考えたい。また、実際に PI-REX 上にシステムを実装し、シミュレーションによる評価実験をおこないたいと考えている。

謝 辞

本研究の一部は、日本学術振興会科学研究費（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] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In *Proc. VLDB*, pp. 953–964, 2006.
- [3] P. Buneman, J. Cheney, W.-C. Tan, and S. Vansummeren. Curated databases. In *Proc. ACM PODS*, 2008.
- [4] B. Gedik and L. Liu. PeerCQ: a decentralized and self-configuring peer-to-peer information monitoring system. *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pp. 490–499, May 2003.
- [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] A. Halevy, M. Franklin, and D. Maier. Principles of Dataspace Systems. In *Proc. ACM PODS*, pp. 1–9, 2006.
- [7] Z. Ives, N. Khandelwal, A. Kapur, and M. Cakir. ORCHES-

TRA: Rapid, collaborative sharing of dynamic data. In *Proc. CIDR*, pp. 107–118, 2005.

- [8] V. Kantere and A. Tsois. Using eca rules to implement mobile query agents for fast-evolving pure p2p database systems. In *MDM '05: Proceedings of the 6th international conference on Mobile data management*, pp. 164–172, New York, NY, USA, 2005. ACM.
- [9] 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.
- [10] F. Li and Y. Ishikawa. Traceable P2P record exchange based on database technologies. In *Proc. APWeb*, pp. 660–671, 2008.
- [11] L. LIU. Continual queries for internet scale event-driven information delivery. *IEEE Trans. Knowledge and Data Engineering*, 11(4):610–628, 1999.
- [12] A. Rowstron, A. marie Kermarrec, M. Castro, and P. Druschel. Scribe: The design of a large-scale event notification infrastructure. In *In Networked Group Communication*, pp. 30–43, 2001.
- [13] W.-C. Tan. Research problems in data provenance. *IEEE Data Eng. Bull.*, 27(4):45–52, 2004.
- [14] Y. Zhu and Y. Hu. Ferry: A p2p-based architecture for content-based publish/subscribe services. *IEEE Transactions on Parallel and Distributed Systems*, 18(5):672–685, 2007.
- [15] 飯田卓也, 李峰栄, 石川佳治. トレーサブルな P2P レコード交換システム PI-REX の設計. 情報処理学会研究報告, 2008-DBS-146:289–294, 2008 年 9 月.
- [16] 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の効率化について. 情報処理学会研究報告, 2008-DBS-146:97–102, 2008 年 9 月.