データベースシステム　125－59
（２００１．７．１８）

情報源固有の制約を考慮した配信型情報源統合方式

渡辺 陽介 †　　北川 博之 ††　　石川 佳治 ††
† 筑波大学 システム情報工学研究科　　†† 筑波大学 電子・情報工学系

概要

現在ネットワーク上で様々な情報源が提供され、異種情報を統合利用する必要性が高まっている。また情報源の形態が多様化し、情報をサーバ側からユーザへ能動的に配信する配信型情報源が広く利用されるようになってきた。我々の研究グループでは、配信型情報源に対応した異種情報源統合環境を構築している。この中では、配信型情報源を扱うための能動的な処理にECAルールを用いている。必要なECAルールはユーザの記述した配信要求から生成されるが、配信型情報源が持つ情報配信スケジュール等の情報現固有の制約を用いることで、ルールによる処理内容をより適切なものとし、またより正確に要求の整合性をチェックすることが可能である。本稿では、情報源固有の制約を考慮に入れた配信要求の処理手法について述べる。

Integration of Multiple Dissemination-Based Information Sources Using Constraints on Information Sources

Yousuke Watanabe †,　　Hiroyuki Kitagawa ††,　　Yoshiharu Ishikawa ††
† Graduate School of Systems and Information Engineering, University of Tsukuba
†† Institute of Information Sciences and Electronics, University of Tsukuba

Abstract

Integration of heterogeneous information sources has been one of important data engineering research issues. Various types of information sources are available today. They include dissemination-based information sources, which actively and autonomously deliver information from server to users. We have been developing a mediator/wrapper-based information integration system, in which we employ ECA rules to define new information delivery services integrating multiple existing dissemination-based information sources. In this system, ECA rules are derived from the relatinal algebra specified by the user. If we utilize constraints on information sources, such as information delivery schedule, system can derive more appropriate ECA rules and more strictly check consistency of requirements. In this paper, we propose a scheme to process information integration requirements using constraints on dissemination-based information sources.

1　はじめに

ネットワークの発達にともない、各種情報源へのアクセスが容易になってきた。そのため、異種情報源の統合利用への要求が増大しており、情報統合はデータ工学の重要な研究課題の一つとなっている[1, 2, 3]。また、情報源の形態もますます多様化し、その１つとして配信型情報源が注目されている。配信型情報源は情報配信サーバからクライアントに対して情報を能動的に配信することを特徴とする情報源であり、これによって，ユーザは情報がどこにあり、いつ更新されたかなどを気にすることなく新鮮な情報を素早く入手することが可能となる。しかし一般には，複数の配信型情報源同士や配信型情報源と他の情報源の情報を統合して新たな配信サービスを提供したり、異なる配信タイミングで情報を配信するといったことは困難である。配信型情報源では情報の配信は随時行われるため、配信型情報源を統合利用するためには、情報の到着や時間経過などのイベントに起因して能動的に情報の再構成や配信処理を行う必要がある。そこで我々は、ECAルール[4]を用いて処理を記述することによって、イベントに起因する情報の再構成や配信処理を行う配信型情報現統合環境を構築した[5]。配信型情報源の統合利用を実現するために必要なECAルールは通常複数になるが、ユーザがこれら複数のルール間の関係を意識しながら記述するのは容易ではない。また、同じ情報源を用いた異なる配信要求同士が相互に関連する場合の一貫性の検証も問題となる。これらの問題に対する１つのアプローチとして、ユーザが配信要求を宣言的に記述すると、その記述から必要なECAルールを自動的に生成するような枠組みが考えられる。我々は配信型情報源をリレーションとしてモデル化し、リレーショナル代数の枠組みをベースとした配信要求の記述方法を定義した[6]。

ECAルールを導出する際、配信要求だけを調べて代数式の検証するよりも、情報源についての詳し

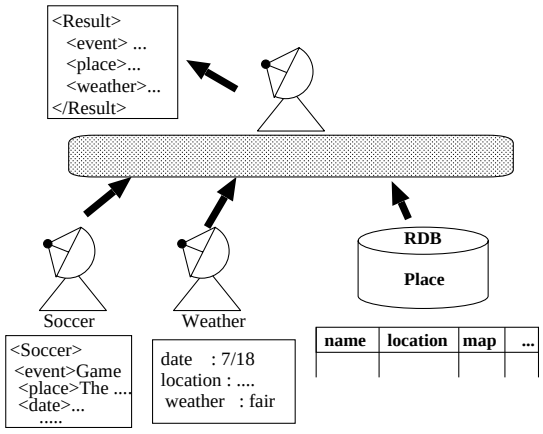

図 1: 統合利用例

い知識を用いた方が有利であると考えられる。配信型情報源には、到着した情報が暗黙の内に満たす制約がいくつか付随していることがある。例えば放送局のような配信型情報源では、番組表に基づいて決まった時刻に情報を送信しており、番組情報の到着時刻には決まったパターンがある。本稿では、そのような配信型情報源に固有な到着時刻の制約に着目し、それらを制約記述言語 [7] を用いてあらかじめ記述して与えておくことで、ユーザから要求が来た時にその制約を考慮したルール生成を行なう方法について述べる。

2 統合例

本節では、配信型情報源を用いた具体的な説明のために以下のような情報源を仮定する (図 1)。

- **サッカー情報配信サービス** (Soccer): この情報源はユーザが応援しているサッカーチームに関係した、試合などのイベント情報を配信している。送られてくる情報には、イベント名 (event)、開催地 (place)、日付 (date)、詳細内容 (moreInfo) などが含まれている。情報は毎週水曜日に届く。
- **天気予報配信サービス** (Weather): 地域毎に、一日の天気や一週間の天気などの天気予報の情報を配信している。日付 (date)、地域 (location)、天気 (weather)、降水確率 (percentage)、予報期間 (term) といった属性を含む。一週間分の天気予報は、毎日 6 時に配信される。
- **施設情報リレーション** (Place): 建物や競技場など施設に関する情報をもった情報源。この情報源は配信型ではなく、通常のリレーショナルデータベースである。属性として施設名 (name)、所在地 (location)、電話番号 (phone)、地図 (map) を持っている。

上記の情報源に対する統合要求として、サッカーの試合に関する情報が到着したら、同じ日に届いた天気予報と試合会場の地図をつけて、翌日の午前 0 時に配信して欲しい、というものが考えられる。これは、サッカー情報配信サービス、天気予報配信サービスという 2 つの配信型情報源と施設情報リレーションという非配信型情報源を統合し、ユーザに対する新たな配信型情報サービスを定義することに相当する。

3 統合システム概要

本節では、我々の研究グループがこれまでに構築を行ってきた配信型情報源統合環境 [5, 6] について説明する。

3.1 アーキテクチャ

本研究の前提となる配信型情報源統合環境は、統合アーキテクチャとしてメディエータ／ラッパー方式を採用している (図 2)。まず情報源ごとに対応するラッパーがあり、情報源に固有な問合せ処理などを請け負っている。ラッパーの上位にはメディエータがあり、情報の統合処理を行う。本システムはメディエータにおける共通データモデルとしてリレーショナルモデルを用いている。

配信型情報源の統合利用には、配信型情報源ラッパーを用いる。配信型情報源ラッパーは、ラッパーとしての機能に加えて、情報配信サーバからの情報 (**配信単位**) が送られてくると情報が到着したことを通知する arrival イベントを発生する機能を持っている。イベントを発生させるモジュールとして、他にタイマーモジュールがあり、指定した時刻が来たことを通知する alarm イベントを発生する。

本統合環境では、各イベントに対応する処理を ECA ルールとして記述する。ECA ルールはルール処理モジュールが保持し、イベントが発生するとそのイベントに対応する ECA ルールを評価・処理して、メディエータに統合処理要求を送ったり、配信モジュールに統合結果の配信要求を出したりする。ルール生成モジュールは、ユーザから渡された配信要求記述から処理に必要な ECA ルールを自動生成する。

2節の統合例を用いて各モジュールの機能を説明する。

1. 配信型情報源ラッパーが天気予報配信サービスから配信単位を受け取ると、ラッパーはそれをリレーションのタプルに変換し、ルール処理モジュールに情報の到着を表すイベント（arrival イベント）を通知する。
2. ルール処理モジュールは通知された arrival イベントに対応する ECA ルールを発火し、到着した配信単位を一時リレーションに蓄積するための要求をメディエータに送る。

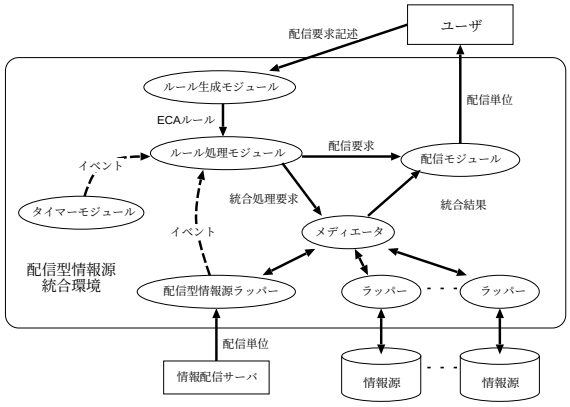

図 2: システムアーキテクチャ

3. サッカー情報配信サービスから配信単位が到着すると、配信型情報源ラッパーは同様に arrival イベントを発生する。

4. ルール処理モジュールは ECA ルールを発火し、到着した情報が試合に関するものであるかどうかをメディエータにチェックさせる。もしそうであれば、メディエータは別の一時リレーションにその情報を蓄積し、午前 0 時にタイマーをセットする。

5. 午前 0 時になると、タイマーモジュールは alarm イベントを発火する。ルール処理モジュールはそのイベントを受け取ると、サッカー情報と同じ日に届いた天気予報、さらに施設情報の統合要求をメディエータに発行する。

6. 最後に、ルール処理モジュールは配信モジュールに統合結果をユーザに配信するように要求する。

3.2 ECA ルール

ECA ルールはアクティブデータベースで使われているものと同じで、イベントに応じた処理を記述するのに用いられる。一つのルールはイベント節 (on 節)、コンディション節 (if 節)、アクション節 (do 節) で構成されている。

イベント節には、ルールが発火するためのきっかけとなるイベントを記述する。情報の到着を表す arrival イベントが配信型情報源ラッパーによって生成され、指定された時刻になったことを知らせる alarm イベントがタイマーモジュールによって生成される。

コンディション節には、アクション節に記述されたアクションが実行されるために満たされるべき条件を記述する。本研究では、コンディション節に記述される条件としてリレーショナル代数演算の選択条件のみを許す。

アクション節には、情報の蓄積、統合、配信のアクションを記述する。これらのアクションは基本的にリレーショナル代数式を用いて記述する。演算子の詳細については [5] を参照されたい。

3.1節で述べたように、配信型情報源を統合して新たな情報配信サービスとして提供するためには、配信単位の蓄積、統合、配信などの処理が必要となる。本研究では、これら一連の処理を以下の 3 つのフェーズに分け、それぞれのフェーズに対応する処理を行なうための ECA ルールを記述することによって配信要求を実現する。

1. **配信単位の蓄積**
   このフェーズは arrival イベントによって発火する。到着した配信単位がユーザの要求を満たすのに必要なら、メディエータ内の一時リレーションに蓄積する。この処理を行うルールを **storage rule** と呼ぶ。

2. **新たな配信情報の生成**
   このフェーズは alarm イベントによって発火し、情報を統合して要求された配信単位を生成する。この処理を行うルールを **generation rule** と呼ぶ。

3. **不要なデータの廃棄**
   一時リレーションから将来使用されないことが明らかな情報を定期的に削除する。この処理を行うルールを **garbage disposal rule** と呼ぶ。このルールは alarm イベントによって発火する。

例えば、 3.1のステップ 2と 4のそれぞれに対応する storage rule は以下のように書ける。

Rule $\mathrm{Storage}_{Weather}$
on: $\mathrm{arrival}(I_{Weather})$
if: $I_{Weather}.term =' week'$
do: $Temp_{I_{Weather}}$ += $I_{Weather}$;

Rule $\mathrm{Storage}_{Soccer}$
on: $\mathrm{arrival}(I_{Soccer})$
if: $I_{Soccer}.event =' game'$
do: $Temp_{I_{Soccer}}$ += I_{Soccer};
setTimer(
$next_{*:*:*:0:0:0}(I_{Soccer}.ITS), new)$;

ここで、$I_{Weather}$ と I_{Soccer} はそれぞれ Weather、Soccer それぞれの情報配信サービスから配信された最新の情報を表す。2 つ目の ECA ルールのアクション節中の $setTimer$ 演算子は、第 1 引数によって記述される時刻にタイマーをセットするための演算である。このタイマーによって発生する alarm イベントは、ステップ 5と 6の処理に対応する generation rule を発火させる。$I_{Soccer}.ITS$ は Soccer の到着時刻を表し、関数 $next_{*:*:*:0:0:0}(I_{Soccer}.ITS)$ はその到着時刻の直後の午前 0 時を表すタイムスタンプを返す。この関数の詳細については 3.3節で述べる。

3.3 配信要求記述

本統合環境は各情報源をリレーションとしてモデル化しており、配信型情報源も同様にリレーションとして表される。情報配信サーバから到着した配信単位は、そのリレーションに属する 1 タプルとしてモデル化される。我々は統合対象となるこのようなリレーションを**I-sequence リレーション**と呼んでいる。I-sequence リレーションは、その情報源が元々持っている属性の他にタプルが到着した時刻を表すための**ITS 属性**を持つ。統合結果も同様にリレーションとして表され、統合結果となる各タプルは配信予定時刻が来るとユーザに配信される。この統合結果のリレーションを**O-sequence リレーション**と呼ぶ。O-sequence リレーションは各タプルの配信予定時刻を表す**OTS 属性**を持つ。ITS・OTS 属性ともに時刻ドメインの値である。

配信型情報源を I-sequence リレーションとしてモデル化したことで、ユーザが要求する統合結果の O-sequence リレーションを I-sequence リレーションや他の情報源のリレーションからどのようにして得るのかをリレーショナル代数で記述することにより、新たな情報配信要求を定義することが可能となる。新たな情報配信要求は以下のような式で定義することができる。

$$O_{new} = \Omega_{f(I_k.ITS)}(E(I_1, \ldots, I_n))$$

O_{new} は O-sequence リレーション、E は I-sequence リレーションや他のリレーションである $I_1, \ldots, I_n$ から新たなリレーションを生成するためのリレーショナル代数式である。また、$\Omega_{f(I_k.ITS)}$ は新たに導入した演算子で、E を評価して得られるリレーションに OTS 属性を付加してその値を $f(I_k.ITS)$ に設定し、ITS 属性を全て除去する。関数 f はタイムスタンプ関数で、I-sequence リレーション I_k の ITS 属性の値から新たなタイムスタンプを生成する。I_k を**マスタ I-sequence リレーション** (または**マスタ情報源**) と呼び、E 中で参照されていなければならない。タイムスタンプ関数として以下の 5 つの関数を考える。

1. $\mathbf{immediate(t)}$: 与えられたタイムスタンプ t そのものを返す。
2. $\mathbf{next_p(t)}$: タイムスタンプ t に最も近い未来に条件 p を満たす時刻を返す。
3. $\mathbf{previous_p(t)}$: タイムスタンプ t に最も近い過去に条件 p を満たす時刻を返す。
4. $\mathbf{after_{\delta t}(t)}$: タイムスタンプ t から時間 δt だけ経過した時刻
5. $\mathbf{before_{\delta t}(t)}$: タイムスタンプ t から時間 δt だけ遡った時刻

時刻の条件 p には以下の形式のうちの一つを与える。

(1) year:month:day:hour:minute:second
(2) year:month:dayofweek:hour:minute:second

year、month、day、hour、minute、second の各フィールドには非負整数かワイルドカード (*) を記述することができる。フィールド dayofweek には、文字列 $\{Sun,\ Mon,\ Tue,\ Wed,\ Thu,\ Fri,\ Sat\ \}$ のうちの一つか、ワイルドカードを指定できる。δt は (1) の形式で記述するが、ワイルドカードは指定できない。

タイムスタンプ関数の利用例として

$$next_{*:*:*:0:0:0}(ITS_{Soccer})$$

は Soccer の配信単位の到着した時刻の次の午前 0 時に対応するタイムスタンプを返す。

この枠組に従うと、2節で説明した新たな情報配信要求の例は以下のような代数式で記述することができる。

$$
\begin{aligned}
&O_{new} = \Omega_{next_{*:*:0:0:0}(Soccer.ITS)}(\\
&\sigma_{Soccer.event='game' \wedge Weather.term='week'}(\\
&\quad Soccer \\
&\quad \bowtie_{Soccer.date=Weather.date} \\
&\qquad \wedge Soccer.place=Place.name \\
&\qquad \wedge Weather.ITS<next_{*:*:*:0:0:0}(Soccer.ITS) \\
&\qquad \wedge previous_{*:*:*:0:0:0}(next_{*:*:*:0:0:0}(Soccer.ITS)) \\
&\qquad \leq Weather.ITS \\
&\quad Weather \\
&\quad \bowtie_{Weather.location=Place.location} \\
&\quad Place \\
&))
\end{aligned}
$$

4 配信型情報源の制約記述

本節では制約記述言語とそれを用いた配信型情報源が持つ制約の記述について述べる。

4.1 制約記述言語

本研究では、[7] で述べられている制約記述言語を用いて情報源の持つ制約を記述する。制約の宣言は以下の形式である。

$$\sigma_{LHS}(I_1 \times \ldots \times I_n) \quad \equiv \quad \sigma_{RHS}(I_1 \times \ldots \times I_n)$$

LHS、RHS は条件式であり、通常 RHS には LHS よりも厳しい条件を記述する。制約の宣言は、情報源 $I_1, \ldots, I_n$ に対して左辺の問合せを行なった結果と、右辺の問合せの結果が常に等しくなるということを表している。すなわち、問合せ中に左辺の記述があるときは、その部分を右辺に置き換えてもよいことが保証される。

制約記述の例として 2節の天気予報配信サービスの location 属性の値は施設情報リレーションの location 属性のどこかに含まれていなければならないという制約は、以下のように書くことができる。

$$\sigma_{True}(Weather) \equiv$$
$$\sigma_{\exists p \in Place(Weather.location = p.location)}(Weather)$$

その情報源が満たす制約を全て調べあげて列挙する必要はなく、既にわかっていて役に立つと思われるものだけを選んで記述するだけでよい。一般に、ユーザは情報源についての詳細な知識は持ち合わせていないことがほとんどであり、自分で制約を発見することは困難である。実際には各情報源の管理者等が情報源を検証して制約を記述し、それを情報源のユーザに提供するという形態が望まれる。

4.2 配信型情報源の制約

配信型情報源がもつ他の情報源とは違った特徴として、到着時刻に関する制約がある。本研究では4.1節の制約記述言語を用いてITS属性の制約を記述し、配信要求記述の処理の際に役立てる。ITS属性の制約記述にもタイムスタンプ関数を用いる。

情報源に固有な制約を記述する前に、まずどの配信型情報源も必ず満さなければならない制約を以下に示す。

$$\sigma_{True}(I) \quad \equiv \quad \sigma_{I.ITS \leq t}(I)$$

これは到着した情報のITS属性は未来の値にならないことを表している。また、ある問合せが配信予定時刻 t より未来の情報を要求しても、この制約によって配信予定時刻までの情報しか要求しない問合せに置き換えることができる。

次に、配信型情報源に固有な制約の記述について述べる。2節のサッカー情報配信サービスのITS属性に関する特徴として情報が到着するのは水曜日であるというものがある。すなわち、ITS属性は水曜日の0時0分0秒から木曜日の0時0分0秒の間の値になるという条件が成り立っているので、以下のように制約を記述することができる。

$$\sigma_{True}(Soccer) \equiv$$
$$\sigma_{Soccer.ITS \leq t}$$
$$\wedge previous_{\star:\star:Wed:0:0:0}(Soccer.ITS)$$
$$\leq Soccer.ITS$$
$$\wedge Soccer.ITS < after_{0:0:1:0:0:0}($$
$$previous_{\star:\star:Wed:0:0:0}(Soccer.ITS))$$
$$(Soccer)$$

2節の天気予報配信サービスのITS属性に関する特徴としては、一週間分の予報は毎日6時に届くというものがある。よって一週間分の天気予報のITS属性は必ず6時になっており、天気予報配信サービスは以下の制約を持つ。

$$\sigma_{Weather.term='week'}(Weather) \equiv$$
$$\sigma_{Weather.term='week'}$$
$$\wedge Weather.ITS \leq t$$
$$\wedge Weather.ITS = after_{0:0:0:6:0:0}($$
$$previous_{\star:\star:\star:0:0:0}(Weather.ITS))$$
$$(Weather)$$

また、この2つの情報源に対して「同じ日に届いた」という条件で結合演算を行なうと、サッカー情報が届かない日の天気予報は常に破棄され、水曜日の天気予報だけにならなければならない。この制約は以下のように記述することができる。

$$\sigma_{previous_{\star:\star:\star:0:0:0}(next_{\star:\star:\star:0:0:0}($$
$$Soccer.ITS)) \leq Weather.ITS$$
$$\wedge Weather.ITS < next_{\star:\star:\star:0:0:0}(Soccer.ITS)$$
$$(Soccer \times Weather)$$
$$\equiv$$
$$\sigma_{previous_{\star:\star:\star:0:0:0}(next_{\star:\star:\star:0:0:0}($$
$$Soccer.ITS)) \leq Weather.ITS$$
$$\wedge Weather.ITS < next_{\star:\star:\star:0:0:0}(Soccer.ITS)$$
$$\wedge previous_{\star:\star:Wed:0:0:0}(Weather.ITS)$$
$$\leq Weather.ITS$$
$$\wedge Weather.ITS < after_{0:0:1:0:0:0}($$
$$previous_{\star:\star:Wed:0:0:0}(Weather.ITS))$$
$$(Soccer \times Weather)$$

4.3 制約の適用処理

情報源 $I_i (i = 1, \ldots, n)$ についての制約の集合 $C = \{C_j\} (j = 0, \ldots, m)$ が与えられたとき、Cを問合せEに対して適用する手続き $Reduce(E, C)$ を以下のように定義する。ただし、Eは選択演算、射影演算、結合演算または直積からなる任意の問合せとする。

関数：*Reduce(E, C)*

1. Eを $\pi_A \sigma_P (I_1 \times \ldots \times I_n)$ の形式に変換する。ただし、A は射影演算によって残す属性、P は問合せ中の選択条件と結合条件を合わせた条件式とする。

2. 以下の処理を条件Pが変化しなくなるまで行なう。

   P に各 C_j の LHS にマッチする条件があるかを調べ、あればその部分を制約記述の RHS に置き換える。このとき、一度使われた制約条件が二度使われることがないようにする。

5 配信要求記述の性質

ユーザが、以下のような配信要求を記述したとする。

$$O_{new} = \Omega_{next_{\star:\star:Tue:0:0:0}(Soccer.ITS)}($$
$$\sigma_{Soccer.ITS \geq previous_{\star:\star:Sun:0:0:0}(t)}$$
$$(Soccer))$$

これは次の火曜日の午前0時に、その週の日曜の午前0時以降に届いたサッカーの情報を配信して欲しいというものである。しかしサッカーの情報は水曜日にならないと届かないのであるから、火曜日の午前0時の時点でこの要求を処理しようとしても全く

意味がない。よってこの配信要求は実現不可能である。このような実現不可能な要求記述を検出するために本研究では、制約を考慮した配信要求の整合性を定義する。これは、[6] で示した整合性の概念をより一般化したものである。

整合性の概念について説明する前に、以下のようなタイムスタンプ関数 $f(t)$ の逆関数 $f^{-1}(t)$ を導入する。

$$f^{-1}(t) = \{u | f(u) = t\}.$$

3.3節で導入した各タイムスタンプ関数に対する逆関数値は以下のような式で計算できる。

(1) $\boldsymbol{immediate}^{-1}$
$immediate^{-1}(t) = \{u | u = t\}$

(2) $\boldsymbol{next}^{-1}$
$next_p^{-1}(t) = \{u | previous_p(t) \leq u < t\}$

(3) $\boldsymbol{previous}^{-1}$
$previous_p^{-1}(t) = \{u | t < u \leq next_p(t)\}$

(4) $\boldsymbol{after}^{-1}$
$after_\delta^{-1}(t) = \{u | u = before_\delta(t)\}$

(5) $\boldsymbol{before}^{-1}$
$before_\delta^{-1}(t) = \{u | u = after_\delta(t)\}$

定義 1 情報源 $I_1, \ldots, I_n$ と、それらに対する制約の集合を C とする。配信要求記述 $O_{new} = \Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n))$ が以下の条件を満たすならば、その配信要求は整合しているという。

$$Reduce(\sigma_{I_k.ITS \in f^{-1}(t)}(E(I_1, \cdots, I_n), C)) = \pi_A(\sigma_{P(t) \wedge Q}(I_1 \times \cdots \times I_n))$$

としたときに、

$$\exists u(\sigma_{P(f(u)) \wedge Q}(I_1 \times \cdots \times I_n) \neq \phi)$$

が成立する。
ただし、 A は射影演算で残す属性、 $P(t)$ は ITS 属性に関する条件式、 Q はそれ以外の属性に関する条件式とする。 □

定義 1は、整合している配信要求にはある配信予定時刻 $t = f(u)$ において配信できる情報が少なくとも一つ存在することを意味している。

前出の実現不可能な例の配信要求記述に $Reduce$ を行なうと以下の式が得られる。

$$\begin{aligned} \mathrm{E} = \quad & \sigma_{previous_{\star:\star:Tue:0:0:0}(t) \leq Soccer.ITS} \\ & \wedge Soccer.ITS < t \\ & \wedge previous_{\star:\star:Sun:0:0:0}(t) \leq Soccer.ITS \\ & \wedge previous_{\star:\star:Wed:0:0:0}(Soccer.ITS) \\ & \quad \leq Soccer.ITS \\ & \wedge Soccer.ITS < after_{0:0:1:0:0:0}(\\ & \quad previous_{\star:\star:Wed:0:0:0}(Soccer.ITS)) \\ & (Soccer) \end{aligned}$$

どんな配信予定時刻 t(ただし $t = next_{*:*:Tue:0:0:0}(u)$) についても、この条件を満たす $Soccer.ITS$ が存在しないので $E = \phi$ になり、よってこの配信要求記述は整合していない。

6 ECA ルールの導出

6.1 処理手順

配信要求記述 $O_{new} = \Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n))$ に対する ECA ルールの導出は以下の手順で行なう。

1. 与えられた配信要求記述から $\sigma_{I_k.ITS \in f^{-1}(t)}(E(I_1, \cdots, I_n))$ を導出する。
2. 4.3節の手続き $Reduce$ により制約を適用する。
$$\pi_A(\sigma_{P(t) \wedge Q}(I_1 \times \cdots \times I_n))$$
ただし、 $P(t)$ は ITS 属性に対する条件、 Q はそれ以外の属性の条件とする。
3. 配信要求が整合しているかチェックする。もし整合していなければ処理を終了する。
4. 式中の選択条件を可能な限りプッシュすることにより、式を以下の形式に書き換える。
$$\pi_A(\sigma_{P_{join}(t) \wedge Q_{join}}(\sigma_{P_1(t) \wedge Q_1}(I_1) \times \cdots \times \sigma_{P_n(t) \wedge Q_n}(I_n)))$$
ここで、 $P_{join}(t), Q_{join}$ は $I_1, \ldots, I_n$ 間の結合条件、 $P_i(t), Q_i$ $(1 \leq i \leq n)$ は各 I_i の属性に対する選択条件である。
5. 各 I-sequence リレーション $I_i (1 \leq i \leq n)$ に対して storage rule を生成する。
6. generation rule を生成する。
7. 各 I-sequence リレーション I_i に対して garbage disposal rule を生成する。

6.2 Storage rule

storage rule は、配信要求記述で参照されている各 I-sequence リレーション I_i $(1 \leq i \leq n)$ に対して生成する。 I-sequence リレーション I_i に対する storage rule は対応する配信型情報源からの情報が到着したときに発火する。そしてコンディション節では、到着した配信単位が以下に示すフィルタリング条件を満たすかどうかをチェックする。もし条件を満たすならば、アクション節で到着した配信単位を一時リレーションに格納するようにメディエータに要求する。さらに I_i がマスタ情報源の場合、配信要求に対応する generation rule を発火させるためにタイマーのセットを行う。

フィルタリング条件は、新たに到着した配信単位が将来使用されるかどうかをチェックするのに用いられる。ステップ 4 において式

$$\pi_A(\sigma_{P_{join}(t) \wedge Q_{join}}(\sigma_{P_1(t) \wedge Q_1}(I_1) \times \cdots \times \sigma_{P_n(t) \wedge Q_n}(I_n)))$$

が得られたとき、 $\sigma_{P_i \wedge Q_i}(I_i)$ からフィルタリング条件を生成することができる。将来使用される配信

単位は条件 Q_i を満たさなければならない。また、ITS 属性に対する条件 $I_i.ITS \in TSet^+(\psi_i, now)$ ("now" は現在時刻を表す) も満たしていなければならない。ただし、

$$\begin{aligned} \psi_i(t) &= \{u | P_i(t)\} \\ TSet^+(\psi_i, t) &= \bigcup_{\tau \geq 0} \psi_i(t+\tau). \end{aligned}$$

である。

3.3節で示した配信要求に 6.1節のステップ 4 までを適用すると、以下の式が得られる。

$$\begin{aligned} &\sigma_{Soccer.date=Weather.date} \\ &\quad \wedge Weather.location=Place.location \\ &\quad \wedge Soccer.place=Place.name \\ &\quad \wedge previous_{*:*:*:0:0:0}(next_{*:*:*:0:0:0}(\\ &\qquad Soccer.ITS)) \leq Weather.ITS \\ &\quad \wedge Weather.ITS \leq next_{*:*:*:0:0:0}(Soccer.ITS)(\\ &\qquad \sigma_{previous_{*:*:*:0:0:0}(t) \leq Soccer.ITS} \\ &\qquad\quad \wedge Soccer.ITS < t \\ &\qquad\quad \wedge previous_{*:*:Wed:0:0:0}(t) \leq Soccer.ITS \\ &\qquad\quad \wedge Soccer.ITS < after_{0:0:1:0:0:0}(\\ &\qquad\qquad previous_{*:*:Wed:0:0:0}(t)) \\ &\qquad\quad \wedge Soccer.event=' game' \\ &\qquad\quad (Soccer) \\ &\qquad\quad \times \\ &\qquad \sigma_{previous_{*:*:*:0:0:0}(t) \leq Weather.ITS} \\ &\qquad\quad \wedge Weather.ITS < t \\ &\qquad\quad \wedge previous_{*:*:Wed:0:0:0}(t) \leq Weather.ITS \\ &\qquad\quad \wedge Weather.ITS < after_{0:0:1:0:0:0}(\\ &\qquad\qquad previous_{*:*:Wed:0:0:0}(t)) \\ &\qquad\quad \wedge Weather.ITS = after_{0:0:0:6:0:0}(\\ &\qquad\qquad previous_{*:*:*:0:0:0}(t)) \\ &\qquad\quad \wedge Weather.term=' week' \\ &\qquad\quad (Weather) \\ &\qquad\quad \times \\ &\qquad (Place))) ; \end{aligned}$$

この式から導出される storage rule は以下のようになる。

Rule Storage$_{Weather}$
on: arrival($I_{Weather}$)
if: $I_{Weather}.ITS \in TSet^+(\psi_{Weather}, now)$
$\wedge I_{Weather}.term =' week'$
do: $Temp_{I_{Weather}}$ += $I_{Weather}$;

ただし

$$\begin{aligned} &\psi_{Weather}(t) \\ &\quad = \{u | previous_{*:*:*:0:0:0}(t) \leq u \wedge u < t \\ &\qquad \wedge previous_{*:*:Wed:0:0:0}(t) \leq u \\ &\qquad \wedge u < after_{0:0:1:0:0:0}(previous_{*:*:Wed:0:0:0}(t)) \\ &\qquad \wedge u = after_{0:0:0:6:0:0}(previous_{*:*:*:0:0:0}(t))\} \end{aligned}$$

Rule Storage$_{Soccer}$
on: arrival(I_{Soccer})
if: $I_{Soccer}.ITS \in TSet^+(\psi_{Soccer}, now)$
$\wedge I_{Soccer}.event =' game'$
do: $Temp_{I_{Soccer}}$ += I_{Soccer};
$setTimer(next_{*:*:*:0:0:0}(I_{Soccer}.ITS), new)$;

ただし

$$\begin{aligned} &\psi_{Soccer}(t) \\ &\quad = \{u | previous_{*:*:*:0:0:0}(t) \leq u \wedge u < t \\ &\qquad \wedge previous_{*:*:Wed:0:0:0}(t) \leq u \\ &\qquad \wedge u < after_{0:0:1:0:0:0}(previous_{*:*:Wed:0:0:0}(t))\} \end{aligned}$$

6.3 Generation rule

generation rule は配信要求記述ごとに生成される。generation rule は、マスタ情報源に対応する storage rule 中でセットされたタイマーによる alarm イベントによって発火する。ステップ 4 で得られる代数式

$$\begin{aligned} &\pi_A(\sigma_{P_{join}(t) \wedge Q_{join}}(\\ &\quad \sigma_{P_1(t) \wedge Q_1}(I_1) \times \cdots \times \sigma_{P_n(t) \wedge Q_n}(I_n))) \end{aligned}$$

に対して、式

$$\begin{aligned} &\pi_A(\sigma_{P_{join}(t) \wedge Q_{join}}(\sigma_{I_1.ITS \in \psi_1(t)}(Temp_{I_1}) \times \\ &\quad \cdots \times \sigma_{I_n.ITS \in \psi_n(t)}(Temp_{I_n}))) \end{aligned}$$

を処理するようにメディエータに要求する。従って、3.3節で示した配信要求から導出される generation rule は以下のようになる。

Rule Generation$_{new}$
on: alarm(new)
if: true
do: $O_{new} = Temp_{Soccer}$

$$\begin{aligned} &\sigma_{Soccer.date=Weather.date} \\ &\quad \wedge Soccer.place=Place.name \\ &\quad \wedge previous_{*:*:*:0:0:0}(next_{*:*:*:0:0:0}(\\ &\qquad Soccer.ITS)) \leq Weather.ITS \\ &\quad \wedge Weather.ITS \leq next_{*:*:*:0:0:0}(Soccer.ITS) \\ &\quad \wedge Weather.location=Place.location(\\ &\sigma_{Soccer.ITS \in \psi_{Soccer}}(Temp_{Soccer}) \\ &\quad \times \\ &\sigma_{Weather.ITS \in \psi_{Weather}}(Temp_{Weather}) \\ &\quad \times \\ &Place\ ); \end{aligned}$$

$Deliver(O_{new})$;

6.4 Garbage disposal rule

ステップ 4 で得られた式

$$\begin{aligned} &\pi_A(\sigma_{P_{join}(t) \wedge Q_{join}}(\\ &\quad \sigma_{P_1(t) \wedge Q_1}(I_1) \times \cdots \times \sigma_{P_n(t) \wedge Q_n}(I_n))) \end{aligned}$$

において、I_i が条件 ψ_i の元で廃棄可能な情報を含みうるならば、I_i に対する garbage disposal rule を生成する。garbage disposal rule はタイマーモ

ジュールから時間INT毎に発生するalarmイベントによって周期的に発火する。ここでは、本システムの管理者が適切なINTの値を決定するものとする。garbage disposal ruleのコンディション節では、条件

$$TSet^{+}(\psi_i, now - INT) - TSet^{+}(\psi_i, now) \neq \phi$$

をチェックする。もし条件が満たされるならば、アクション節で不要な情報の廃棄を行う。廃棄処理は以下の式によって表される。

$$Temp_{I_i} -= \sigma_{t \in TSet^{+}(\psi_i, now-INT) - TSet^{+}(\psi_i, now)}(Temp_{I_i})$$

3.3節で示した配信要求から導出されるgarbage disposal ruleは以下のようになる。

Rule GarbageDisposal$_{Weather}$
on: alarm($GarbageDisposal_{Weather}$)
if: $TSet^{+}(\psi_{Weather}, now - INT) - TSet^{+}(\psi_{Weather}, now) \neq \phi$
do: $Temp_{I_{Weather}} -= \sigma_{Temp_{I_{Weather}}.ITS \in TSet^{+}(\psi_{Weather}, now-INT) - TSet^{+}(\psi_{Weather}, now)}(Temp_{I_{Weather}})$;
$setTimer(now + INT, GarbageDisposal_{Weather})$;

Rule GarbageDisposal$_{Soccer}$
on: alarm($GarbageDisposal_{Soccer}$)
if: $TSet^{+}(\psi_{Soccer}, now - INT) - TSet^{+}(\psi_{Soccer}, now) \neq \phi$
do: $Temp_{I_{Soccer}} -= \sigma_{Temp_{I_{Soccer}}.ITS \in TSet^{+}(\psi_{Soccer}, now-INT) - TSet^{+}(\psi_{Soccer}, now)}(Temp_{I_{Soccer}})$;
$setTimer(now + INT, GarbageDisposal_{Soccer})$;

7 結論

本稿では、リレーショナル代数式による配信要求記述からECAルールを生成して新たな情報配信サービスを定義できるような配信型情報源において、制約記述言語を用いて配信型情報源に固有な制約を表現し、配信要求の処理に適用する方法について述べた。また、制約を考慮した配信要求の整合性を定義した。

本稿で提案した枠組みは、現在3節で示したアーキテクチャをもとにして、我々の研究グループで研究開発を進めてきたプロトタイプシステム上での実装作業を行なっている。現段階の実装では、条件式を分割する際に与えられた配信要求から導出されるすべての条件を出力している。条件式が多いと実行時にルールを評価する効率が低下してしまうので、条件式を最適化するような枠組を今後検討していく必要がある。また、条件式を変形する際に現在用いている変換規則では変換できないケースがあり得る。従って、より柔軟に対応できる変換規則を検討する必要があると考えられる。

参考文献

[1] A. Elmagarmid, M. Rusinkiewicz, and A. Sheth (Eds.). Management of Heterogeneous and Autonomous Database Systems. Morgan Kaufmann, 1999.

[2] A. Morishima and H. Kitagawa. InfoWeaver: Dynamic and Tailor-Made Integration of Structured Documents, Web, and Databases. *Proc. ACM DL '99*, pp. 235–236, 1999.

[3] H. Kitagawa, A. Morishima, and H. Mizuguchi. Integration of Heterogeneous Information Sources in InfoWeaver. *Advances in Database and Multimedia for the New Century – A Swiss/Japanese Perspective*, World Scientific Publishing, pp. 124–137, 2000.

[4] N. W. Paton and O. Diaz. Active Database Systems. *ACM Comp. Serv.*, 31(1), 1999.

[5] H. Mizuguchi, H. Kitagawa, Y. Ishikawa, and A. Morishima. A Rule-oriented Architecture to Incorporate Dissemination-based Information Delivery into Information Integration Environments. *Proc. 2000 ADBIS-DASFAA*, pp. 185–199, 2000.

[6] H. Kitagawa, T. Kajino, and Y. Ishikawa. Algebraic Service Specification and Rule Generation for Integrating Multiple Dissemination-Based Information Sources. Proc. 7th International Conference on Database Systems for Advanced Applications. 2001, pp. 344–351.

[7] Garcia-Molina, Labio, and Yang. Expiring Data in a Warehouse. Proc. 24th VLDB Conference, pp. 500–511, 1998.

[8] M. Qiang, H. Kondo, K. Sumiya, and K. Tanaka. Virtual TV Channel: Filtering Merging and Presenting Internet Broadcasting Channels. *ACM DL Workshop on WOWS*, 1999.

[9] Infogate Inc.. Infogate. http://www.infogate.com/.

[10] Marimba Inc.. Castanet. http://www.marimba.com/products/castanet-intro.htm.

[11] Microsoft Inc.. Active chanel. http://msdn.microsoft.com/workshop/delivery/.

[12] S. Ramakrishnan and V. Dayal. The PointCast Network. *ACM SIGMOD Record*, 27(2), p. 520, 1998.