データベースシステム　122－11
(2000. 7. 26)

配信型情報源統合環境における配信サービス定義とルール生成

梶野 智行† 北川 博之†† 石川 佳治††

†筑波大学 理工学研究科 ††筑波大学 電子・情報工学系

〒305-8573 茨城県つくば市天王台 1-1-1
筑波大学 電子・情報工学系 データベース研究室
Tel. 0298-53-5385
E-mail: {kaji, kitagawa, ishikawa}@dblab.is.tsukuba.ac.jp

概要

ネットワークの発達に伴い，各種情報源へのアクセスが容易になり，これらの統合利用への要求が増大している．また，情報源の形態も多様化し，ユーザに能動的に情報を配信する配信型情報源が注目されている．したがって，このような情報源に対応した異種情報源統合利用システムが重要になると考え，我々の研究グループではECAルールを用いた配信型情報源統合利用環境を構築してきた．しかしこのシステムではユーザがECAルールを記述することにしているので，複数ルールの記述や整合性の検証などをユーザ自身が行なう必要がある．そこで本稿では，宣言的な記述による配信サービス定義の枠組を定義し，その記述からECAルールを自動生成するための生成規則を導出する．

キーワード: 情報配信，情報統合，ルール処理

Specification of Dissemination Services and Derivation of ECA Rules in Dissemination-Based Information Integration Environments

Tomoyuki Kajino†, Hiroyuki Kitagawa†† and Yoshiharu Ishikawa††

†Master's Program in Sciences and Engineering, University of Tsukuba
††Institute of Information Sciences and Electronics, University of Tsukuba

Database Laboratory, Institute of Information Sciences and Electronics, University of Tsukuba
1-1-1 Tennohdai, Tsukuba, Ibaraki 305-8573, Japan
Tel. +81-298-53-5385
E-mail: {kaji, kitagawa, ishikawa}@dblab.is.tsukuba.ac.jp

Abstract

Integration of heterogeneous information sources has been one of important research issues in recent advanced network environments. In these days, various types of information sources are available. Dissemination-based information sources that actively deliver information from server sites to users are important information sources. In our research group, a dissemination-based information integration system that uses ECA rules to process dissemination-based information sources has been built to incorporate delivered information in a flexible manner. However, the users of the system have to specify and verify ECA rules by themselves. In this paper, we present a framework to specify dissemination services declaratively, and to derive ECA rules automatically.

key words: information dissemination, information integration, rule processing

1 はじめに

近年ネットワークの発達に伴い，各種情報源へのアクセスが容易になってきた．そのため，これらの情報源を統合利用したいという要求が増大している．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．しかし配信型情報源は，配信情報と他の情報源の情報を用いて新たな情報を生成したり，配信情報のフィルタリングによって欲しい情報だけを入手するといったことが困難であるという問題点があげられる．これらの背景を元に，我々の研究グループでは，配信型情報源を含む各種情報源を統合して扱うための機構の研究を行なってきた [3]．一般に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 [3] では、ユーザが ECA ルール [4] を用いて要求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ことにしている．しかし、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、記述しなければならない ECA ルールの数が多数になる可能性もあるため、それらを全てユーザが記述するのは現実的ではない．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．

本研究では、このような枠組みを実現するために、配信型情報源をモデル化し、そのモデルの上での配信要求の宣言的な記述方法を定義する．さらに、宣言的な記述から ECA ルールを自動生成するための生成規則を導出する．以下では、2節で本研究で想定するアーキテクチャについて述べる．3節で ECA ルールについて述べたあと、4節で配信型情報源のモデル化と配信要求の宣言的な記述方法，及び具体的な配信要求の記述例について述べる．続いて5節で宣言的記述から ECA ルールを生成するための生成規則について述べる．6節で ECA ルール生成の具体例を示し、7節で複数の配信要求が与えられた場合の処理について述べる．8節で生成されたルールの最適化について述べ，9節でまとめと今後の課題について述べる．

2 アーキテクチャ

本研究では，メディエータ / ラッパー方式に基づいた配信型情報源統合利用環境 [3] を前提とする．この統合利用環境では，統合対象として配信型情報源，及びリレーショナルデータベースなどを扱うことができる．また，これらの情報源の統合結果を新たな情報配信サービスとして提供することが可能である．このシステムでは，配信型情報源を扱うためのモジュールとして，配信型情報源ラッパー，ルール処理モジュール，配信モジュールを持つ (図 1)．

配信型情報源ラッパーは，情報配信サーバから送られてくる情報 (**配信単位**) を受け取り，情報が到着したことを知らせるイベントを発生する．ルール処理モジュールでは，

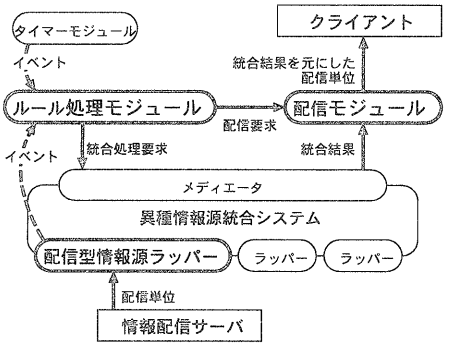


図 1: アーキテクチャ

配信単位の到着や時間経過などのイベントに起因する処理を記述した ECA ルールを保持しており，イベントが発生すると，これらのルールを評価・処理する．配信モジュールは，配信を要求された統合結果をクライアントへ配信する．その際，統合結果をその配信方法において適切な形式に変換する．また，時刻や時間に関するイベントを発生するモジュールとしてタイマーモジュールを持つ．

各モジュールを用いた情報の統合・配信処理は次のように行なわれる．配信型情報源ラッパーが情報配信サーバから配信単位を受け取ると，その配信単位を統合システムで扱っている内部形式に変換して蓄積し，配信単位の到着を知らせるイベントをルール処理モジュールに通知する．また，タイマーモジュールから発生したイベントも同様にルール処理モジュールに伝えられる．イベントを受け取ったルール処理モジュールは，通知されたイベントに対応するルールを処理し，統合システムに統合処理要求を発行する．そして統合結果を配信モジュールが受け取って適切な形式に変換し，クライアントに配信する．

3 ECA ルール

本節では，配信処理を行なうために用いる ECA ルールについて説明し，本研究におけるルールの記述方法について述べる．

ECA ルールとは，アクティブデータベースで用いられており，何らかのイベントに起因して実行される処理を記述するためのものである．ECA 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) の3つの部分によって構成される．イベント節には，時間経過など，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．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．アクション節には，処理対象となるデータのデータモデルに基づいた演算式を記述する．

2節で述べたように，本研究では，イベントを発生するモジュールとして配信型情報源ラッパーとタイマーモジュールの2つがあり，配信型情報源ラッパーでは，情報配信サーバからの情報の到着によるイベントを，またタイマーモジュールでは，時刻や時間に関するイベントをそれぞれ発生する．したがって本研究では，情報の到着を表す

arrival イベント，指定した時刻になったことを表す alarm イベント，指定した時間が経過したことを表す interval イベントを，イベント節に記述することのできるイベントとして扱う．イベント節には，これら 3 つのイベントのうち 1 つ以上を組み合わせて記述する．

また，アクションとして記述する処理には，システム内部の一時領域に対する情報の追加，削除，一時領域の上書きの操作が考えられる．これらの操作は，それぞれ以下のように記述する．

$Temp$ += $NewData$
　情報 $NewData$ を一時領域 $Temp$ に追加する．

$Temp$ -= $NewData$
　情報 $NewData$ を一時領域 $Temp$ から削除する．

$Temp$:= $NewData$
　一時領域 $Temp$ を情報 $NewData$ で上書きする．

ここで $Temp$ は一時領域の識別子，$NewData$ は扱うデータモデルに基づいた演算式の処理結果を表す．

例えば，各企業の株価の情報を配信する株式情報配信サービス $DS_{StockPrice}$ があるとする．このサービスは，企業名，業種，株価を 1 つの配信単位としてユーザに配信しているとする．ここで，業種が IT 関連である企業の情報が配信されてきたら，システムの一時領域 StockPrice に蓄積して欲しいという要求があった場合の ECA ルールは以下のように記述する．

Rule StorageITStockPrice
　on: arrival($DS_{StockPrice}$)
　if: $DS_{StockPrice}.Category$ = 'IT'
　do: StockPrice += $DS_{StockPrice}$;

4　配信要求の宣言的記述

本節では，配信型情報源に対する宣言的な記述を実現するために，まず配信型情報源のモデル化について述べる．また，配信要求の宣言的な記述の方法について述べ，要求の記述に必要な演算の定義を行なう．

4.1　配信型情報源のモデル化

最初に述べたように，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，配信サービスが開始してから終了するまでの間，情報の配信は随時行なわれる．この配信型情報源を論理的に見ると，サービスの開始から終了までに配信されてくる情報を全て含む，データの 1 つの集まりであるとみなすことができる．また，情報の配信は記事，メッセージなどの配信単位ごとに行なわれる．そこで本研究では，1 つの配信サーバから配信されてくる全ての配信単位の集まりを 1 つのリレーションとみなし，配信単位をリレーションのタプルに対応させることによって，配信型情報源をモデル化する．また，情報の配信は時間の経過に沿って行なわれるので，配信の順序を表すために，このリレーションの各タプルには，タイムスタンプが付加されているとする (図 2)．

2 節で述べたように，配信型情報源の利用形態として，(1) 統合対象の情報源としての利用，(2) 統合結果の配信

TS	A_1	$\cdots$	A_n
T_1			
T_2			
$\vdots$			
T_m			

TS: タイムスタンプ
$A_1, \cdots, A_n$:
　情報源固有の情報

図 2: 配信型情報源のモデル化

手段としての利用，という 2 形態が考えられる．これら 2 通りの利用形態のいずれにおいても，先に述べたような配信型情報源をリレーションとしてモデル化するということが可能であるが，統合利用環境の中では，それぞれのタイムスタンプが表す意味は異なるものであると考える．つまり，(1) の利用形態においては，配信単位が情報配信サーバからいつ配信されてきたのかが重要であり，(2) の利用形態においては，統合処理によって生成された配信単位をいつ配信するのかが重要であると考えられる．そこで，(1) の形態で利用する配信型情報源の各配信単位に付加されるタイムスタンプを **ATS**(Arrival Time Stamp) と呼び，配信単位の到着時刻を表すこととする．また，(2) の形態で利用する配信型情報源の各配信単位に付加されるタイムスタンプを **DTS**(Delivery Time Stamp) と呼び，生成された配信単位の配信予定時刻を表すこととする．

4.2　配信要求記述のための枠組み

1 つまたは複数の情報源から，新たな配信サービスを定義するためには，各情報源の情報のうちどの情報を抜き出してどのような結果を生成するのかという配信情報の定義と，生成した各配信単位をいつ配信するのかという配信予定時刻の定義という 2 つの定義を与える必要がある．先ほど述べたように，本研究では，配信型情報源をタイムスタンプを付加したリレーションとしてモデル化した．したがって，各情報源の情報に対する処理はリレーショナル代数を用いて記述することが可能であり，配信情報の定義はリレーショナル代数を用いて与えればよい．

一方，配信単位が到着した時刻の次の午前 8 時に配信して欲しい，あるいは配信単位に含まれる情報の一部を時刻とみなし，その時刻に生成した配信単位を配信して欲しいなど，生成した配信単位の配信予定時刻は，配信単位の到着時刻や含まれる情報を利用して与えることができる．しかし，リレーショナル代数自身には時刻を扱うための枠組が存在しないため，次の午前 8 時というようなことを記述できないので，従来のリレーショナル代数では配信予定時刻の定義を与えることができない．また，あるリレーションに対して新たな属性を追加するといった処理もリレーショナル代数では定義されていないため，仮に配信予定時刻を生成できたとしても配信予定時刻を格納するための属性の追加ができない．そこで本研究では，従来のリレーショナル代数に時刻を扱うための関数と新たな属性を追加するための演算を追加した拡張リレーショナル代数を定義し，それによって配信サービスの定義を記述することを考える．

タイムスタンプに対する処理としては，今度の日曜日の

午前8時を表すタイムスタンプを求めたい，ある2つのタイムスタンプが同じ日付であるかどうかを調べたいといった処理が考えられる．本研究では，タイムスタンプを扱うための関数として，以下のような関数を定義する．

scale ： 与えられたタイムスタンプの粒度を変更するための関数である．例えば，2つのタイムスタンプ TS_1 と TS_2 が同じ日付であるかどうかを調べたいという例の場合には，

$$scale(TS_1,'day') = scale(TS_2,'day')$$

と記述することにより，2つのタイムスタンプを日にち単位で比較可能である．

next ： 与えられた条件を満たす次の時刻を表すタイムスタンプを求めるための関数である．例えば，タイムスタンプ TS が表す時刻の次の日曜日の午前8時を表すタイムスタンプを求めたいといった場合には，

$$next(TS,'Sunday, 8:00a.m.')$$

という記述によって，要求するタイムスタンプが得られる．

max ： 複数のタイムスタンプから，時間的に最新のタイムスタンプを求めるための関数である．

また，リレーションに新たな属性を追加し，ある処理の結果を追加した属性の値として格納するための演算として，Apply 演算をリレーショナル代数に新たに導入する．Apply 演算は以下のような形式で記述する．

$$A_{D_n=f(D_1,\cdots,D_m)}(R)$$

この演算の意味は，リレーション R に新たな属性 D_n を追加し，リレーション R の属性 $D_1, \cdots, D_m$ を引数とする関数 f を評価した結果を属性 D_n の値とするということである．例えば，ある配信サービス DS の配信単位の到着時刻が ATS であり，配信予定時刻 DTS を ATS の次の日曜日の午前8時に設定するための処理は以下のように記述される．

$$A_{DTS=next(ATS,'Sunday,8:00a.m.')}(DS)$$

4.3 配信要求の記述例

ここで，具体的な統合利用例を1つ示し，その利用例に対する配信要求の記述例を示す．

まず配信型情報源として，各企業に関するニュースを配信しているサービスと，各企業の株価情報を配信しているサービスの2つがあるとする．企業ニュース配信サービスでは，企業名とニュースのタイトル及び記事が配信される (図3)．株価情報配信サービスでは，企業名と業種，その企業の株価の終り値が1日1回配信される (図4)．またリレーショナルデータベースには，ユーザ名とそのユーザが現在所有している各企業の株の購入時の価格，株数，閾値として注目している株価の情報が格納されている (図5)．

ATS	Company	Title	Article

図 3: 企業ニュース配信サービス

ATS	Company	Category	Price

図 4: 株式情報配信サービス

Name	BuyPrice	Amount	Threshold

図 5: 所有株式情報リレーション

この時，ユーザ U_1 が株を所有するIT関連企業の株価が指定した閾値以上になったら，その企業の過去3日分のニュース記事と株価と企業名をまとめて配信して欲しいという要求があったとする．

まず，この例に対する配信要求として，過去3日分という条件の元で株式情報と企業ニュースを結合しなければならない．このことを記述するためには株価情報の到着時刻と企業ニュースの到着時刻に対して，日にち単位で比較を行なう必要がある．この比較を行なうために，ここでは時刻を扱う関数 scale を用いて結合条件を記述する．またユーザは，条件を満たす株価情報が到着したらすぐに配信することを要求しているので，配信予定時刻 DTS は株価情報の到着時刻に等しくなる．したがって，この例に対する配信要求の記述は以下のようになる．

$$
\begin{aligned}
&Example1 = \\
&\quad \Pi_{DTS,Company,Price,Article}(\\
&\qquad A_{DTS=ATS}(\\
&\qquad\quad \sigma_{Category='IT'}(StockPrice) \\
&\qquad\quad \bowtie_{Company=ComC} \\
&\qquad\qquad \wedge scale(ATS,'day')-3 \leq scale(ATSc,'day') \\
&\qquad\qquad \wedge scale(ATSc,'day') \leq scale(ATS,'day') \\
&\qquad\quad \delta_{ATS \leftarrow ATSc}(\\
&\qquad\qquad \delta_{Company \leftarrow ComC}(ComNews)) \\
&\qquad\quad \bowtie_{ComC=ComR \wedge Price \geq Threshold} \\
&\qquad\quad \delta_{Company \leftarrow ComR}(\\
&\qquad\qquad \sigma_{Name='U_1'}(UserStockInfo))))
\end{aligned}
$$

ここで，配信要求の記述中の演算子 δ は，リレーションの属性名を変更するための改名演算子である．例えば $\delta_{Company \leftarrow ComC}(R)$ という記述は，リレーション R 中の属性名 $Company$ を $ComC$ に変更することを表す．

5 ECA ルールの生成

本節では，配信型情報源の統合・配信処理の際に用いられる ECA ルールの役割について述べ，配信要求からそれらのルールを生成するための生成規則について述べる．

本研究では，情報源統合に必要な一連の処理を，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，ユーザが記述した配信要求から生成する．

1. 配信単位の蓄積

   情報配信サーバから送られてきた配信単位から，配信要求を満たすのに必要な情報を選択し，システム

内に蓄積するフェーズである．このフェーズの処理を行なうルールを **Storage Rule** と呼ぶ．

2. 新たな配信情報の生成

   情報をユーザに配信すべきタイミングになった時に，蓄積した情報などを使って新たな情報を生成するフェーズである．このフェーズの処理を行なうルールを **Generation Rule** と呼ぶ．

3. 不要な情報の廃棄

   システム内に蓄積された情報のうち，2度と使用されることのない情報を保持しておくことはスペースの無駄使いである．このフェーズでは，すでに不要となった情報の廃棄を行なう．このフェーズの処理を行なうルールを **Expiration Rule** と呼ぶ．

以下では，それぞれの ECA ルールの生成規則について述べる．ECA ルール生成の具体例は，6 節で示す．

5.1 Storage Rule

Storage Rule は，(1) 情報配信サーバから配信される配信単位が到着した時点で発火し，(2) 配信単位をフィルタリングして，(3) システム内の一時リレーションに蓄積する，という 3 つのステップから構成される．これら 3 つのステップのうち，(1) はイベント節で，(2) はコンディション節で，(3) はアクション節で記述する．また Storage Rule は，各情報配信サーバからの配信単位の到着毎に処理を行なわなければならないので，1 つの配信型情報源に対してルールを 1 つ生成する．ただし，配信要求の中で使用されていない配信型情報源については Storage Rule を生成しない．以下で，それぞれの節の生成規則を述べる．

(1) イベント節

Storage Rule は，情報の到着によってルールが発火するので，arrival 記述子によって，

$$\mathrm{arrival}(DS)$$

のように記述する．

(2) コンディション節

コンディション節では，配信されてきた情報のフィルタリングを行なう．配信要求においては，情報のフィルタリングは対応する配信型情報源に対する選択演算で行なわれるので，コンディション節に記述するフィルタリング条件は，その選択演算の選択条件を抜き出したものを記述すればよい．より具体的には，配信要求の記述中の演算のうち，単一の配信型情報源に対する選択演算の選択条件を抜き出して OR 結合したものをコンディション節とする．

(3) アクション節

アクション節では，フィルタリングされた配信単位をシステム内の一時リレーションに格納する処理を行なうので，アクション節の記述は以下のようになる．

$$Temp_{DS} \mathrel{+}= DS$$

ここで，$Temp_{DS}$ は一時リレーションの識別子，DS は配信型情報源の名前である．

また，アクション節では，必要に応じてタイマーのセットを行なう．タイマーのセットを行なう必要があるのは，その Storage Rule に対応する配信型情報源の配信単位の到着時刻やそれに含まれる情報に依存して結果の配信予定時刻が決まる場合である．この場合，到着した配信単位中の情報から配信予定時刻を計算し，その時刻にタイマーイベントが発生するようにタイマーをセットする．例えば，時刻 DTS にタイマーをセットする場合，以下のように記述する．

$$\mathrm{setTimer}(DTS)$$

以上をまとめると，Storage Rule の生成規則は以下のようになる．ここで，アクション節中の setTimer(DTS) の両側の中括弧は，この処理の記述がオプショナルであることを示す．

Rule Storage
on: arrival(DS)
if: 選択演算から抜き出した選択条件
do: $Temp_{DS}$ += DS;
{setTimer(DTS)};

5.2 Generation Rule

Generation Rule は，(1) Storage Rule でセットされた時刻になったら発火し，(2) 統合処理及び配信処理を行なう，という 2 ステップから構成される．このうち，(1) はイベント節で，(2) はアクション節でそれぞれ記述する．これら 2 つの節の生成規則は以下の通りである．

(1) イベント節

Generation Rule は，Storage Rule でセットされた時刻 DTS になった時点で発火するので，alarm 記述子によって，

$$\mathrm{alarm}(DTS)$$

のように記述する．

(2) アクション節

配信要求で記述された演算のうち，単一の配信型情報源に対する選択演算は，Storage Rule においてすでに処理されている．また，配信予定時刻の生成を行なうための処理は，配信予定時刻に Generation Rule を発火するという処理によって終了している．したがって，Generation Rule のアクション節には，ユーザが記述した配信要求から，Storage Rule で処理された選択演算と配信予定時刻を生成するための Apply 演算を除いた部分を記述する．

これらをまとめると，Generation Rule の生成規則は以下のようになる．

Rule Generation
on: alarm(DTS)
if: true
do: ユーザの記述から不要な
選択演算と Apply 演算を除いた式

5.3 Expiration Rule

Expiration Rule は，(1) あるタイミングで発火し，(2) 不要となった蓄積情報を廃棄する，という2ステップから構成される．このうち (1) はイベント節で，(2) はアクション節で記述する．Expiration Rule の処理は，各配信型情報源に対応する一時リレーション毎に行なう必要があるので，それぞれの配信型情報源に対して1つのルールを生成する．ただし，Storage Rule が生成されなかった配信型情報源については，廃棄処理の必要がないので，Expiration Rule を生成しない．これら2つの節の生成規則は以下の通りである．

(1) イベント節

Expiration Rule の発火タイミングとしては，主に2通りが考えられる．1つは Generation Rule の処理が終った時点で発火する場合，もう1つはある一定時間毎に発火する場合である．このうち前者の考え方は，新たな配信単位を生成した後に，不要な蓄積情報が発生するであろうという考えに基づくものである．

しかし，システム内に蓄積している情報は，あくまでも新たな配信単位を生成するのに必要である可能性があるだけで，必ず使用されるという保証はない．例えば，2つの配信サービス A，B があり，A にある条件を満たす情報が到着したら，その時点での B の過去1週間分の情報を配信して欲しいという要求が与えられたとする．この時仮に A から条件を満たす情報が1ヶ月間配信されなかった場合を想定すると，Generation Rule の処理が終った時点で Expiration Rule が発火するアプローチをとった場合，少なくとも約3週間分の情報を無駄に蓄積してしまうことになる．したがって本研究では，Expiration Rule はある一定の時間間隔 τ で発火するというアプローチをとる．Expiration Rule におけるイベント節の記述は以下のようになる．

$$\text{interval}(\tau)$$

(2) アクション節

システム内の一時リレーションに蓄積された情報は，蓄積された時点では全て必要である可能性があった情報である．しかし，時間の経過とともに不要な情報が発生してくる．必要である可能性のあったものが不要になる理由は，その情報を使用して生成される新たな配信単位の配信予定時刻を過ぎるためである．したがって Expiration Rule のアクション節では，蓄積情報の到着時刻と生成される配信単位の配信予定時刻，それと現在時刻の関係を調べ，配信予定時刻が過去の時刻である場合には，該当する蓄積情報を廃棄するという処理を記述すればよい．具体的には，配信要求の記述中から，時刻が条件に含まれる結合演算，選択演算，Apply 演算を抜き出し，各配信型情報源の一時リレーションにおいて保持しておくべき情報の条件を生成する．

これらをまとめると，Expiration Rule の生成規則は以下のようになる．

Rule Expiration
on: interval(τ)
if: true
do: ユーザの記述から生成した廃棄処理

Expiration Rule のアクション節の生成については，6節で具体的な例を用いて説明する．

6 ECA ルールの生成例

本節では，先に示した統合利用例に対して5節で示したルール生成規則を適用することによって得られる ECA ルールについて示す．

4.3 節で示した配信要求の記述に対して5節で述べた ECA ルール生成規則を適用すると，以下のようになる．

Rule StorageExample1_ComNews
on: arrival($DS_{ComNews}$)
if: true
do: ComNews += $DS_{ComNews}$;

Rule StorageExample1_StockPrice
on: arrival($DS_{StockPrice}$)
if: $DS_{StockPrice}.Category$ = 'IT'
do: StockPrice += $DS_{StockPrice}$;
setTimer(T_{ATS});

Rule GenerationExample1
on: alarm(T_{ATS})
if: true
do: $Example1$ =
$\Pi_{DTS,Company,Price,Article}($
$StockPrice$
$\bowtie_{Company=ComC}$
$\wedge scale(ATS,'day')-3 \leq scale(ATSc,'day')$
$\wedge scale(ATSc,'day') \leq scale(ATS,'day')$
$\delta_{ATS \leftarrow ATSc}($
$\delta_{Company \leftarrow ComC}(ComNews))$
$\bowtie_{ComC=ComR \wedge Price \geq Threshold}$
$\delta_{Company \leftarrow ComR}($
$\sigma_{Name='U_1'}(UserStockInfo)))$;

Rule ExpirationExample1_ComNews
on: interval(τ)
if: true
do: $ComNews$:=
$\sigma_{scale(now,'day')-3 \leq scale(ATS,'day')}$
$\wedge scale(ATS,'day') \leq scale(now,'day')}(ComNews)$;

Rule ExpirationExample1_StockPrice
on: interval(τ)
if: true
do: $StockPrice$:= $\sigma_{now \leq ATS}(StockPrice)$;

企業ニュース配信サービスの Storage Rule (StorageExample1_ComNews) では，ユーザの記述中にこの情報源に対する選択演算が含まれていないので，コンディション節が true となっている．また，配信予定時刻が株式情報配信サービスの到着時刻となっており，このサービスからの情報の到着が Generation Rule の発火の要因となるので，株式情報配信サービスの Storage Rule (StorageExample1_StockPrice) のアクション節の最後でタイマーをセットしている．

2つの配信型情報源に対する Expiration Rule のイベント節中の τ は，Expiration Rule が発火する周期を表す．また，now は現在時刻を表す．各ルールのアクション節の導出手順は以下の通りである．まず，蓄積情報を保持するための条件として，生成された配信単位の配信予定時刻が現在時刻以降であることがあげられるので，配信予定時刻を DTS とすると，

(i) $now \leq DTS$

という条件を満たす情報を保持しておけばよい．

ユーザの記述中の時刻が含まれる条件は，

(ii) $DTS = ATSs$

(iii) $scale(ATSs,'day') - 3 \leq scale(ATSc,'day')$
$\wedge scale(ATSc,'day') \leq scale(ATSs,'day')$

の2つである．いずれの配信型情報源においても，到着時刻を格納した属性の属性名は ATS であるが，説明の便宜上株式情報配信サービスからの情報の到着時刻を $ATSs$，企業ニュース配信サービスからの情報の到着時刻を $ATSc$ と記述する．

株式情報配信サービスに対する蓄積情報の保持条件は，上記の条件 (i)，(ii) から

$$now \leq ATSs$$

となる．また，企業ニュース配信サービスに対する保持条件は，上記の条件 (iii) と株式情報配信サービスに対する保持条件から，

$$scale(now,'day') - 3 \leq scale(ATSc,'day')$$
$$\wedge scale(ATSc,'day') \leq scale(now,'day')$$

となる．この保持条件を選択条件とする選択演算がアクション節に記述される．

7 複数の要求に対する処理

前節までは，システムに対して複数の要求が与えられた場合の処理を想定していなかった．しかし，複数の要求が同時に与えられることの方が一般に多くあると考えられる．本節では，同時に複数の要求が与えられた場合に生じる問題点とその解決法について述べる．

複数の要求が同時に与えられた場合，6節までに説明した規則を適用すると，2つの問題点が出てくる．1つは配信情報の蓄積に関する問題，もう1つは不要な蓄積情報の廃棄に関する問題である．説明のために，もう1つの統合利用例を示す．

情報源として，6節で示したものと同じ3つの情報源が存在するとする．この時ユーザ U_2 は，日曜日の午前8時に，U_2 が株を所有する企業の1週間分のニュース記事と企業名をまとめて配信して欲しいとする．この例に対する配信要求は以下のようになる．

$$
\begin{aligned}
&Example2 = \\
&\quad \Pi_{DTS,Company,Price,Article}(\\
&\qquad A_{DTS=next(ATS,'Sunday,8:00a.m.')}(\\
&\qquad\quad ComNews \\
&\qquad\quad \bowtie_{Company=ComR} \\
&\qquad\quad \delta_{Company \leftarrow ComR}(\\
&\qquad\qquad \sigma_{Name='U_2'}(UserStockInfo))))
\end{aligned}
$$

この要求に対する Storage Rule と Expiration Rule は以下のようになる．Generation Rule はここでは省略する．また，この要求では株式情報配信サービスからの情報は使用されていないので，生成されるルールは企業ニュース配信サービスに対するもののみである．

Rule StorageExample2_ComNews
on: arrival($DS_{ComNews}$)
if: true
do: ComNews += $DS_{ComNews}$;
setTimer($next(ATS,'Sunday,8:00a.m.')$);

Rule ExpirationExample2_ComNews
on: interval(τ)
if: true
do: $ComNews$:=
$\sigma_{now \leq next(ATS,'Sunday,8:00a.m.')}(ComNews)$;

蓄積と廃棄それぞれの問題点の詳細と，その解決法を以下に述べる．

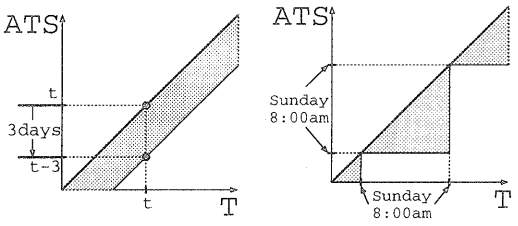
7.1 配信情報の蓄積処理

Storage Rule では，それぞれの配信要求において必要な配信単位をシステム内の一時リレーションに蓄積する．しかし，蓄積処理はそれぞれの配信要求毎に行なわれるため，配信されてきた配信単位を必要とする要求が複数あった場合，同一の情報が複数蓄積されることになり，生成される結果が正しくないものになる可能性がある．そこで，複数の配信要求が与えられても，Storage Rule は1つしか生成しないというアプローチをとる．具体的には，ある配信サービスに対する Storage Rule が存在しない場合には，5節で述べた生成規則によってルールを生成する．一方，既に Storage Rule が存在する場合には，新たなルールを生成するのではなく，既存のルールのコンディション節に新たに追加された配信要求に対するフィルタリング条件を OR 結合によって追加する．

7.2 蓄積情報の廃棄処理

Expiration Rule では，それぞれの配信要求において不要になった蓄積情報を一時リレーションから削除する．しかし，Storage Rule の場合と同様に処理は配信要求毎に行なわれるため，ある配信要求では必要な情報を，別の配信要求から生成された ECA ルールの処理によって廃棄されてしまうということが発生しうる．図6は，それぞれの配信要求において保持すべき情報を図示したものである．図の横軸は廃棄処理を行なう時刻，縦軸は配信単位の到着時刻であり，図全体として，ある時刻において保持すべき到着時刻の範囲を表している．

図6の網かけの部分がそれぞれの配信要求において保持すべき情報を表しており，2つの図の和にあたる領域が全体として保持すべき情報である．しかし先ほども述べたように，配信要求毎に別々に処理してしまうと，2つの図の重ならない領域は互いに不要であると判断して廃棄してしまうことになる．そこで，Storage Rule の場合と同様に，Expiration Rule は全体として1つしか生成せず，各配信型情報源の情報の保持条件を OR 結合したものをアクション節の選択演算の条件として記述することにする．この規則により生成される Expiration Rule は以下のようになる．



(a) Example 1 (b) Example 2

図 6: 廃棄処理の際の保持条件

Rule Expiration_ComNews
on: interval(τ)
if: true
do: $ComNews$:=
$\sigma_{scale(now,'day')-3 \leq scale(ATS,'day')}$
$\wedge scale(ATS,'day') \leq scale(now,'day')$
$\vee now \leq next(ATS,'Sunday,8:00a.m.')(ComNews)$;

Rule Expiration_StockPrice
on: interval(τ)
if: true
do: $StockPrice := \sigma_{now \leq ATS}(StockPrice)$;

8 ルールの最適化

本節では，先に導出したルール生成規則を適用して得られるECA ルールの最適化について述べる．

ここでは統合利用例として，6 節であげた例を少し変更したものを用いる．すなわち，A 社の株価が，ユーザ U_1 が指定した閾値以上になったら，その企業の過去 3 日分のニュース記事と株価と企業名をまとめて配信して欲しいという要求がシステムに対して与えられているとする．この例に対する配信要求は以下のようになる．

$Example3 =$
$\Pi_{DTS,Company,Price,Article}($
$A_{DTS=ATS}($
$\sigma_{Company='A'}(StockPrice)$
$\bowtie_{Company=ComC}$
$\wedge scale(ATS,'day')-3 \leq scale(ATSc,'day')$
$\wedge scale(ATSc,'day') \leq scale(ATS,'day')$
$\delta_{ATS \leftarrow ATSc}($
$\delta_{Company \leftarrow ComC}(ComNews))$
$\bowtie_{ComC=ComR \wedge Price \geq Threshold}$
$\delta_{Company \leftarrow ComR}($
$\sigma_{Name='U_1'}(UserStockInfo))))$

この配信要求に対して生成規則を適用すると，以下のような Storage Rule が生成される．

Rule StorageExample3_ComNews
on: arrival($DS_{ComNews}$)
if: true
do: ComNews += $DS_{ComNews}$;

Rule StorageExample3_StockPrice
on: arrival($DS_{StockPrice}$)
if: $DS_{StockPrice}.Company$ = 'A'
do: StockPrice += $DS_{StockPrice}$;
setTimer(T_{ATS});

ここで配信要求をよく見てみると，株式情報配信サービスから A 社の情報のみが選択され，そのあとで企業ニュース配信サービスの情報のうち，企業名が等しいものとの結合が行なわれている．したがって，結合演算の結果として現れるのは A 社の情報のみであり，企業ニュース配信サービスにおいても A 社の情報かどうかでフィルタリング可能であることが分かる．このように，ある配信型情報源に対する選択条件を他の配信型情報源に伝搬させると，蓄積すべき情報を減らすことができる場合がある．

9 まとめと今後の課題

本稿では，配信型情報源統合利用環境において，新たな配信サービスを定義することを目的とした代数系を定義し，この代数を用いた配信要求の宣言的記述方法について述べた．また，宣言的記述から，配信処理に必要な 3 種類のルールを自動生成するための生成規則を導出した．また，複数の配信要求が与えられた場合も考慮に入れた ECA ルール生成規則の改善について述べた．また，生成された ECA ルールの最適化の可能性についても述べた．

今後の課題としては，ここで述べた ECA ルール生成規則によって生成されるルールの最適化などを含む，生成規則のより一層の形式化を行なうことがあげられる．また，Expiration Rule はある周期 τ で発火すると述べたが，τ が長過ぎると，不要な情報を長期間無駄に蓄積してしまうことになる．また，τ が短過ぎると，ルールが発火しても廃棄すべき情報がない可能性が高くなり，ルールの処理自体が無駄になってしまう．したがって，適切な τ をどのように決定するかについても今後検討していく必要がある．

謝辞

本研究の一部は，文部省科学研究費基盤研究 (B) (12480067) 及び奨励研究 (A) (12780183) の助成による．

参考文献

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

[2] 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.

[3] 水口 弘紀，北川 博之，石川 佳治，森嶋 厚行．情報配信サービスを利用した情報源統合環境の構築．DEWS2000, 2000 年 3 月．

[4] Norman W. Paton, Oscar Diaz. Active Database Systems. ACM CS, March, 1999.