Algebraic Service Specification and Rule Generation for Integrating Multiple Dissemination-Based Information Sources

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

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

February 2001

ISE–TR–01–177

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 sites to users. We have been developing a mediator/wrapper-based information integration system, in which we employ ECA rules to enable users to define new information delivery services integrating multiple existing dissemination-based information sources. However, it is not easy for users to directly specify ECA rules and to verify them. In this paper, we propose a scheme to specify new dissemination-based information delivery services using the framework of the relational algebra. We discuss some important properties of the specification, and show how we can derive ECA rules to implement the services.

1 Introduction

The recent development of network technology has enabled us to access various information sources easily. Due to the need for the integration facility of heterogeneous information sources, information integration (mediation) has been one of important data engineering research issues [5, 6, 9, 14]. Although the technological advance has made it possible to integrate existing heterogeneous information sources, we still have to cope with a new kind of information source - *dissemination-based information sources* [1, 8, 11, 17]. They actively and autonomously deliver information from server sites to clients. Therefore, users can receive up-to-date information automatically, and do not have to worry about where the information is located or when they are updated.

We have been developing a mediator/wrapper-based information integration system named *InfoWeaver* which integrates various information sources including dissemination-based information sources [9, 12, 14]. InfoWeaver provides two types of features regarding the information dissemination: (1) Integration of dissemination-based information sources and (2) Dissemination-based delivery of the integration result. The former enables user-specified selection of the delivered information and integration with other information sources such as relational databases and Web pages. The latter enables effective delivery of the integrated data from the integration system based on the timing requirement specified by users. For example, it is possible to extract interesting portions from the disseminated information, integrate them with data in relational databases, and then periodically deliver the integration results to users using the dissemination facility.

These features require "active" facilities such as event handling and support for timing constraints. Namely, data storage, integration, and delivery need to be performed actively, triggered by events such as arrival of new data and time progress. To realize this, we have employed ECA rules [15]. Users can define new information delivery services, if they add appropriate ECA rules to the system.

However, it is not an easy task for users to specify ECA rules. Moreover, ECA rules are related with each other, and it is difficult to check their consistency. For example, when more than one information delivery services are defined on the same dissemination-based information sources, verification of ECA rules becomes very complicated.

An approach to this problem is to provide a framework in which users can specify their integration and delivery requirements in a more declarative manner. In this paper, we propose a specification scheme based on the relational algebra, employing the relational model as a common model at the mediator level. In the scheme, information sources including dissemination-based sources are modeled

as relations, and new information delivery services are specified using relational algebra-based expressions. Then, consistency of the specifications is checked, and ECA rules to realize the user requirements are derived from algebra-based expressions, and the integration system provides the *information delivery service* by executing the ECA rules. The relational algebra-based specification scheme proposed here provides a sound basis on top of which more declarative and user-friendly specification schemes can be developed. (They may be based on SQL and GUI.)

The remaining part of this paper is organized as follows. In Section 2, we show a simple example of a new information delivery service integrating multiple existing dissemination-based information sources. Mediator/wrapper-based integration system architecture assumed in this paper is shown in Section 3. Section 4 describes ECA rules employed in our approach. In Section 5, we show how dissemination-based information sources are modeled as relations, and present a framework to specify new information delivery services based on the relational algebra. In Section 6, we discuss some properties to verify the given specifications. In Section 7, we show how we can derive ECA rules to implement the services from them. Section 8 mentions related works. Section 9 presents the conclusion and future works.

2 Integration Example

In this section, we show a simple integration scenario. We define a new information delivery service integrating dissemination-based information sources and a relational database (Figure 1). We assume that there exist two dissemination-based information sources:

1. *Industrial news service (INews)*: This source sends industrial news articles to the clients using e-mails. Each e-mail message contains data items such as the company name, news header, and contents.

2. *Stock price information feed service (SPrice)*: This source provides the closing stock price information over the Internet. It is delivered once a day on a per-company base. Namely, it sends a record that consists of the company name, its category of business, and the closing stock price of the day.

In addition to these dissemination-based information sources, we assume the following relational database:

3. *Stockholder's information database (StockholderInfo)*: This database contains a relation that manages the information of the stock items that a stockholder owns. The relation has attributes such as the company name, the purchase price, the number of stocks, and a threshold price value specified by the owner. (How it is used is described below.)

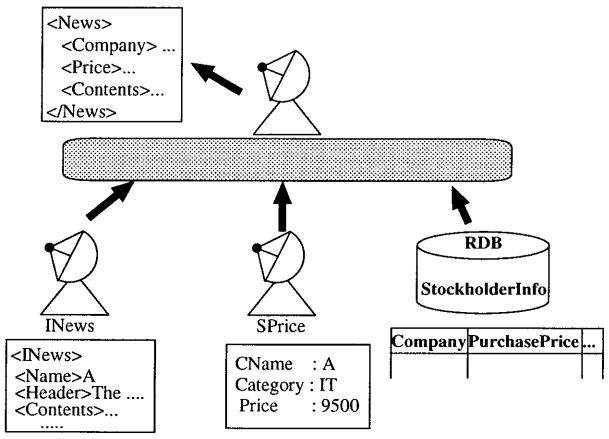

Figure 1: Integration Example

Here, we suppose a user owns stocks of companies in the IT category, and their information is stored in the Stockholder's information database. To show an information integration example, assume that the user has the following demands:

- When a new closing stock price for an IT category company has arrived, check it and send a notification message to the user if it exceeds its *threshold value*. The threshold value is specified by the user for each IT stock item and stored as an attribute value in the Stockholder's information database.
- The notification message should include the closing stock price, company name, and news contents related with the company extracted from the industrial news articles delivered in the last two days.
- The notification should be sent to the user at midnight on the day.

The above demands can be met by defining a new information delivery service in which notification messages generated from data in the underlying three information sources are periodically sent to the user. We show how this requirement is specified in our scheme in Section 5. In Section 7, we present how to derive ECA rules to implement the service from the specification.

3 Integration Architecture

In our research, we assume a mediator/wrapper-based information integration system architecture [9, 12, 14]. The relational model is used here as a common data model at the mediator level. To cope with data incoming from dissemination-based information sources, we allocate wrappers, named *DIS wrappers*, to dissemination-based information sources in addition to wrappers for relational databases and Web pages. This system integrates data from the underlying information sources, triggered by events such as arrival of new data and time progress. The integration results can be actively delivered from the system to users through new dissemination-based information services. (The integration results can also be obtained by conventional queries. This portion is out of the scope in this paper.)

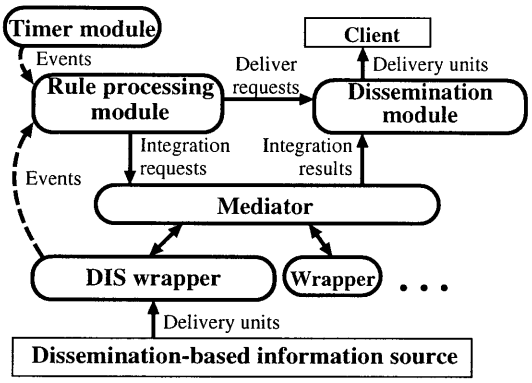

Figure 2: Integration System Architecture

The system employs ECA rules to specify such event-driven data integration and delivery actions. The system architecture is shown in Figure 2. A DIS wrapper receives information (*delivery units*) sent from a dissemination-based information source, and raises events to notify arrival of delivery units. The *rule processing module* holds ECA rules, which specify actions to be triggered by events raised by DIS wrappers and the *timer module*. When events are raised, the rule processing module executes relevant ECA rules, which invoke data storage, integration, and delivery actions. The *mediator* is responsible for data integration. It also manages temporary relations to store delivery units. The *dissemination module* is responsible for delivery of the integration results obtained from the mediator. More details of the integration system architecture are given in [12].

Using the integration example in Section 2, we illustrate the function of each module.

1. When the DIS wrapper receives a delivery unit from the industrial news service, the wrapper translates it into a tuple in the relational model, and notifies an arrival event to the rule processing system.

2. The rule processing system invokes an ECA rule corresponding to the notified event, and sends a request to the mediator to store the new data in a temporary relation.

3. When a delivery unit from the stock price information feed service has arrived, the DIS wrapper also raises an arrival event.

4. The rule processing module invokes another ECA rule, and requests the mediator to check whether the delivered data is related to an IT category company. If it is, the mediator stores the new data in another temporary relation, and sets the timer alarm for midnight of the day.

5. At midnight, the timer module raises an event. Then, the rule processing module requests the mediator to integrate the stock price information and the industrial news articles related with the company which were delivered in the last two days, if the stock price exceeds the specified threshold value.

6. Finally, the rule processing system orders the dissemination module to deliver the integration result to the user.

4 ECA Rules

In this section, we explain ECA rules and classify them into three categories from the viewpoint of their roles in our context. Basically, ECA rules in this paper are same as those in active databases [15]. An ECA rule consists of three parts: the *event* (**on**), *condition* (**if**), and *action* (**do**) clauses.

The event clause specifies an event that triggers activation of the rule. There are two kinds of events: *primitive event* and *composite event*. As indicated in Section 3, the following two primitive events are raised from DIS wrappers and the timer module:

1. **arrival**(*R*): This event is raised from a DIS wrapper and notifies arrival of a new delivery unit from the dissemination-based information source *R*.

2. **alarm**(*Alarm Name*): This event is raised from the timer module when the set time has come. *Alarm Name* designates each alarm event, and is given when the timer module alarm is set by the *setTimer* operator.

A composite event is constructed from primitive events and/or composite events. Details are omitted here.

The condition clause specifies a precondition which must be met for the action clause to be processed. In this context, we allow the condition clauses to contain selection conditions in relational algebra expressions.

The action clause specifies data storage, integration, and delivery actions. Basically, those actions are expressed in relational algebra. As mentioned in Section 3, the mediator manages temporary relations. Update of temporary relations are denoted as shown in Figure 3. In addition, the Deliver operator is used to send data delivery requests to the delivery module. More details of the Delivery operator are given in [12].

(Temp) **:=** *(Expression)*:
Tuples *(Expression)* are assigned to a temporary relation *(Temp)*.
(Temp) **+=** *(Expression)*:
Tuples *(Expression)* are appended to a temporary relation *(Temp)*.
(Temp) **–=** *(Expression)*:
Tuples *(Expression)* are deleted from a temporary relation *(Temp)*.

Figure 3: Manipulation of Temporary Relations

As explained in Section 3, various operations, such as data storage, data integration, and data delivery, are involved in the process of providing a new data delivery service on top of dissemination-based information sources. We divide this process into the following three phases, and each phase is implemented as a set of ECA rules.

1. *Storage of delivery units*: In this phase, triggered by arrival events, the mediator checks whether data sent from the dissemination-based information sources are necessary to meet the user requirements. If they are, they are stored in temporary relations. In the example scenario in Section 3, steps 2 and 4 correspond to this phase. ECA rules employed to implement this phase are called *storage rules*.

2. *Generation of new delivery units*: In this phase, triggered by alarm events, the mediator extracts relevant data from temporary relations and generates the requested delivery units, integrating them with data in other information sources. Then, the delivery module is invoked for data delivery. In the example in Section 3, steps 5 and 6 correspond to this phase. ECA rules used for this phase are called *generation rules*.

3. *Disposal of unnecessary delivery units*: In this phase, unnecessary data in temporary relations are thrown away. In the example in Section 3, stock price data stored in the temporary relation becomes obsolete after midnight. Also, industrial news articles become of no use after midnight the next day. ECA rules used for this purpose are called *garbage disposal rules*, and their activations are triggered by alarm events.

Storage rules corresponding to steps 2 and 4, respectively, in the example in Section 3 can be written as follows.

Rule Storage$_{INews}$
on: arrival(I_{INews})
do: $Temp_{I_{INews}}$ **+=** I_{INews};

Rule Storage$_{SPrice}$
on: arrival(I_{SPrice})
if: $I_{SPrice}.Category = 'IT'$
do: $Temp_{I_{SPrice}}$ **+=** I_{SPrice};
setTimer($next_{*:*:*:0:0:0}(I_{SPrice}.ITS)$, *new*);

Here, I_{INews} and I_{SPrice} denote the most recently delivered data unit (tuple) from the INews and SPrice information sources, respectively. The *setTimer* operator in the action clause of the second rule sets the timer alarm for the time specified by $next_{*:*:*:0:0:0}(I_{SPrice}.ITS)$. The alarm event invokes a generation rule corresponding to steps 5 and 6. $I_{SPrice}.ITS$ denotes the arrival time of the SPrice data, and the function $next_{*:*:*:0:0:0}(I_{SPrice}.ITS)$ derives the time corresponding to midnight of the day. Details of such functions to derive designated time are explained in Section 5.

5 Algebraic Service Specification

As mentioned in Section 3, we use the relational model as a common model at the mediator level. In this section, we show how to specify new information delivery services integrating dissemination-based information sources based on the relational algebra. In Section 7, we show how ECA rules can be derived from the specifications.

First, we model data incoming from dissemination-based information sources and data outgoing through new information delivery services as relations. Relations modeling data from dissemination-based information sources are called *I-sequence relations*. Relations modeling data outgoing through newly defined

information delivery services are called *O-sequence relations*. In both sequence relations, tuples correspond to delivery units. I-sequence relations have the timestamp attribute ITS, which designates the arrival time of the corresponding delivery unit. O-sequence relations have the timestamp attribute OTS, which designates the (scheduled) delivery time of the corresponding delivery unit. In addition to ITS and OTS, they have their own attributes depending on the information sources and the delivery requests.

Logically, an I-sequence relation represents all the data delivered from the underlying dissemination-based information source from its service start time to the service end time. Of course, when the information source is still in service, the whole instance of the I-sequence relation cannot be materialized. Similarly, an O-sequence relation represents all the data delivered through the new information delivery service. Modeling incoming and outgoing data as sequence relations, we can define a new information delivery service by specifying how to obtain an O-sequence relation from I-sequence relations and, if necessary, conventional relations representing other information sources. The basic idea here is to employ the relational algebra for this purpose. The relational algebra provides a sound basis.

In our scheme, a new information delivery service is defined by the following formula:

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

where O_{new} is an O-sequence relation and E is a relational algebra expression to derive a new relation from I-sequence relations $I_1, \cdots, I_n$ and other conventional relations. The operator $\Omega_{f(I_k.ITS)}$ is introduced here. It adds the OTS attribute, sets $f(I_k.ITS)$ as its value, and remove all the ITS attributes included in the relation obtained by E. The function f is a *timestamp function*, and derives a new timestamp from the ITS attribute value in I-sequence relation I_k. (More details of timestamp functions are explained below.) I_k is called a *master I-sequence relation* (or *information source*), and must be referenced in E.

The new information delivery service explained in Section 2 can be specified as follows:

$$\begin{aligned}
O_{new} = \Omega_{next_{*:*:*:0:0:0}(I_{SPrice}.ITS)}(& \\
& \sigma_{Category='IT'}(I_{SPrice}) \\
& \bowtie_{CName=Name} \\
& \quad \wedge previous_{*:*:*:0:0:0}(before_{0:0:1:0:0:0}(I_{SPrice}.ITS)) \leq previous_{*:*:*:0:0:0}(I_{INews}.ITS) \\
& \quad \wedge previous_{*:*:*:0:0:0}(I_{INews}.ITS) \leq previous_{*:*:*:0:0:0}(I_{SPrice}.ITS) \\
& I_{INews} \\
& \bowtie_{Name=Company \wedge Price \geq Threshold} \\
& StockholderInfo).
\end{aligned}$$

Here, I_{INews} and I_{SPrice} are I-sequence relations, and $I_{INews}.ITS$ and $I_{SPrice}.ITS$ are their ITS attributes, respectively. The functions *next*, *previous*, *before* are timestamp functions. The selection, projection, and join are represented by σ, π, and $\bowtie$, respectively.

Timestamp functions considered in the paper are as follows:

1. **immediate**(t): Returns the given timestamp t itself.
2. **next**$_p(t)$: Returns the timestamp, matching the temporal pattern p, chronologically next to the timestamp t.
3. **previous**$_p(t)$: Returns the timestamp, matching temporal pattern p, chronologically previous to the timestamp t.
4. **after**$_{\delta t}(t)$: Returns the time (timestamp) after the time interval δt from the time t.
5. **before**$_{\delta t}(t)$: Returns the time (timestamp) before the time interval δt from the time t.

The temporal pattern p is given in one of the following formats:

(1) *year:month:day:hour:minute:second*

(2) *year:month:dayofweek:hour:minute:second*

Non-negative integers and the wild card symbol '*' can be specified for fields *year*, *month*, *day*, *hour*, *minute* and *second*[1]. The field *dayofweek* can contain one of strings {Sun, Mon, Tue, Wed, Thu, Fri, Sat} or '*'. The time interval δt is specified using the format (1), but '*' is not allowed.

For example,

$$next_{*:*:*:0:0:0}(I_{SPrice}.ITS)$$

gives the timestamp corresponding to the next midnight from the arrival time $I_{SPrice}.ITS$.

$$previous_{*:*:*:0:0:0}(I_{INews}.ITS) = previous_{*:*:*:0:0:0}(I_{SPrice}.ITS)$$

checks whether the timestamps $I_{INews}.ITS$ and $I_{SPrice}.ITS$ imply the same day.

To simplify the discussion, we make the following assumptions in the remaining part of this paper.

[1]To simplify the discussion, we assume that '*' is always specified for *year*.

1. No I-sequence relation is referenced in E more than once.

2. In relations referred in E, the timestamp values appear only in ITS attributes, and only timestamp functions and primitive comparison operators ($=, \neq, <, >, \leq, \geq$) can be used in selection and join conditions concerning timestamps in E.

6 Basic Properties of Service Specifications

Unfortunately, users could specify infeasible information delivery services as follows:

$$
\begin{aligned}
O_{new} = \Omega_{immediate(I_{SPrice}.ITS)}(& \\
& I_{SPrice} \\
& \bowtie_{CName=Name} \\
& \quad \wedge previous_{*:*:*:0:0:0}(after_{0:0:1:0:0:0}(I_{SPrice}.ITS)) = previous_{*:*:*:0:0:0}(I_{INews}.ITS) \\
& I_{INews}).
\end{aligned}
$$

The expression specifies the following requirement:

> *As soon as the system receives a closing stock price data of some company, say A, from the stock price information feed source, it should deliver it with the next day's news articles related with A.*

Obviously, it is infeasible, since the expected news articles have not yet arrived when new delivery units are to be delivered. To exclude such infeasible specifications, we define the notion of consistency.

Definition 1 *The specification* $O_{new} = \Omega_{f(I_k.ITS)}\ (E(I_1, \cdots, I_n))$ *is said to be consistent, if, for all t,*

$$
\begin{aligned}
& \sigma_{OTS=t}(\Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n))) \\
& = \sigma_{OTS=t}(\Omega_{f(I_k.ITS)}(E(\sigma_{ITS \leq t}(I_1), \cdots, \sigma_{ITS \leq t}(I_n)))).
\end{aligned}
$$

□

This definition means that tuples to be delivered at time t must be generated only from tuples obtained up to the time t.

In our scheme, we check the consistency of a given specification based on a theorem. Before explaining the theorem, we introduce the *inverse* $f^{-1}(t)$ for the timestamp function $f(t)$ as follows:

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

For each timestamp function introduced in Section 5, the inverse is defined as follows.

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

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

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

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

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

When the expression $f(t) = f_1(f_2(\cdots(f_n(t))\cdots))$ is given, $f^{-1}(t)$ is defined as follows:

$$\begin{aligned} f^{-1}(t) &= \{u | (\exists u_{n-1}) \cdots (\exists u_1) \\ &\quad (u \in f_n^{-1}(u_{n-1}) \wedge \cdots \wedge u_2 \in f_2^{-1}(u_1) \wedge u_1 \in f_1^{-1}(t))\}. \end{aligned}$$

Theorem 1 *If the specification* $O_{new} = \Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n))$ *satisfies the following conditions, it is consistent:*

(i) *The expression* $\sigma_{I_k.ITS \in f^{-1}(t)}(E(I_1, \cdots, I_n))$ *can be rewritten into the following form:*

$$E'(\sigma_{I_1.ITS \in \psi_1(t)}(I_1), \cdots, \sigma_{I_n.ITS \in \psi_n(t)}(I_n)),$$

where $\psi_1(t), \cdots, \psi_n(t)$ *give selection conditions regarding the ITS values of* $I_1, \cdots, I_n$*, respectively.*

(ii) *For each* $i (1 \leq i \leq n)$, $\psi_i(t)$ *has the upper bound* $max(\psi_i(t))$*, and it is always less than or equal to* t*:* $max(\psi_i(t)) \leq t$. □

Proof *In Definition 1, OTS corresponds to the formula* $f(I_k.ITS)$ *in the specification. Therefore, the lefthand side of the equation in Definition 1 can be transformed as follows:*

$$\begin{aligned} &\sigma_{OTS=t}(\Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n))) \\ &= \Omega_{f(I_k.ITS)}(\sigma_{f(I_k.ITS)=t}(E(I_1, \cdots, I_n))) \\ &= \Omega_{f(I_k.ITS)}(\sigma_{I_k.ITS \in f^{-1}(t)}(E(I_1, \cdots, I_n))). \end{aligned} \quad (1)$$

By Theorem 1(i) and (ii), the expression (1) can be transformed into

$$\Omega_{f(I_k.ITS)}(\sigma_{I_k.ITS \in f^{-1}(t)}(E(\sigma_{I_1.ITS \leq t}(I_1), \cdots, \sigma_{I_n.ITS \leq t}(I_n)))).$$

Replacing the selection condition $I_k.ITS \in f^{-1}(t)$ *with* $f(I_k.ITS) = t$, *we obtain*

$$\sigma_{OTS=t}(\Omega_{f(I_k.ITS)}(E(\sigma_{I_1.ITS\leq t}(I_1),\cdots,\sigma_{I_n.ITS\leq t}(I_n)))). \quad \square$$

The rewriting in Theorem 1(i) is done by pushing down the selection conditions on ITS values as in the conventional query optimization process [18]. Theorem 1(ii) assures that tuples to be delivered at time t can be generated only from tuples obtained up to the time t.

As we mentioned in Section 4, we employ storage rules, generation rules, and garbage disposal rules. However, in some cases, we do not need garbage disposal rules. Let us consider the following requirement:

> *The system should deliver all the important news articles related with company A obtained in the past on its anniversary every year.*

The specification expressing this requirement is obviously consistent. If only this delivery service is defined and important news articles on company A are stored in a temporary relation, we cannot throw any data away. In this case, we need no garbage disposal rule for the temporary relation. Definition 2 defines the property related with this. In Definition 2 and the remaining part of this paper, we use the following notations:

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

Definition 2 *Assume a consistent specification* $O_{new} = \Omega_{f(I_k.ITS)}(E(I_1,\cdots,I_n))$ *is given so that it can be rewritten into the form of Theorem 1(i). Then, if*

$$\psi_i(t) - TSet(\psi_i,t) \neq \phi$$

for some t, it is said that I-sequence relation I_i *may contain disposable data under* ψ_i. $\square$

As shown in the next section, garbage disposal rules are generated only for I-sequence relations which may contain disposable data.

7 Derivation of ECA Rules

In this section, we discuss derivation of ECA rules from the relational algebra-based specifications explained in Section 5. First, we show how to derive storage rules, generation rules, and garbage disposal rules from the specification of a new information delivery service in Subsection 7.1. Subsection 7.2 discusses cases where multiple information delivery services are specified.

7.1 Basic Derivation Scheme

7.1.1 Overview

For a specification $O_{new} = \Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n))$, derivation of ECA rules is outlined as follows:

1. Transform the given specification into $\sigma_{I_k.ITS \in f^{-1}(t)}(E(I_1, \cdots, I_n))$.
2. Push down selection conditions as much as possible as in the conventional query optimization scheme, and rewrite the above expression into the following form:

   $$E'(\sigma_{C_1}(\sigma_{I_1.ITS \in \psi_1(t)}(I_1)), \cdots, \sigma_{C_n}(\sigma_{I_n.ITS \in \psi_n(t)}(I_n))),$$

   where $C_i\,(1 \leq i \leq n)$ is the selection condition on I_i's attributes except ITS.
3. Check whether the specification is consistent. If it is not, exit from the procedure.
4. Derive a storage rule for each I-sequence relation $I_i\,(1 \leq i \leq n)$.
5. Derive a generation rule.
6. For each I-sequence relation I_i, check whether it may contain disposable data under ψ_i. If it is, derive a garbage disposal rule for I_i.

When we apply the steps 1 and 2 to the example specification given in Section 5, we get the following expression.

$$
\begin{array}{l}
\sigma_{Category='IT'}(\\
\quad \sigma_{ISPrice.ITS \in \{u | previous_{*:*:*:0:0:0}(t) \leq u \wedge u < t\}}(I_{SPrice})) \\
\quad \bowtie_{CName=Name} \\
\qquad \wedge previous_{*:*:*:0:0:0}(before_{0:0:1:0:0:0}(I_{SPrice}.ITS)) \leq previous_{*:*:*:0:0:0}(I_{INews}.ITS) \\
\qquad \wedge previous_{*:*:*:0:0:0}(I_{INews}.ITS) \leq previous_{*:*:*:0:0:0}(I_{SPrice}.ITS) \\
\quad \sigma_{I_{INews}.ITS \in \{u | previous_{*:*:*:0:0:0}(before_{0:0:1:0:0:0}(previous_{*:*:*:0:0:0}(t))) \leq u} \\
\qquad \wedge u < previous_{*:*:*:0:0:0}(t)\}(I_{INews}) \\
\quad \bowtie_{Name=Company \wedge Price \geq Threshold} \\
\quad StockholderInfo
\end{array}
$$

In the above expression, $\psi_{SPrice}(t)$, $\psi_{INews}(t)$ are identified as follows:

$$
\begin{array}{rcl}
\psi_{SPrice}(t) & = & \{u | previous_{*:*:*:0:0:0}(t) \leq u \wedge u < t\}, \\
\psi_{INews}(t) & = & \{u | previous_{*:*:*:0:0:0}(before_{0:0:1:0:0:0}(\\
 & & \quad previous_{*:*:*:0:0:0}(t))) < u \wedge u \leq previous_{*:*:*:0:0:0}(t)\}.
\end{array}
$$

They satisfies the condition of Theorem 1(ii) as follows:

$$\begin{aligned} max(\psi_{SPrice}(t)) &= t \leq t, \\ max(\psi_{INews}(t)) &= previous_{*:*:*:0:0:0}(t) \leq t. \end{aligned}$$

Therefore, the specification is consistent.

In the following, we present more details of the steps 4 through 6.

7.1.2 Storage Rules

A storage rule is derived for each I-sequence relation $I_i\,(1 \leq i \leq n)$ referenced in the service specification. The storage rule for I-sequence relation I_i is invoked when a delivery unit has arrived from the underlying dissemination-based information source. Then, in the condition clause, it checks whether the new data (tuple) satisfies the filtering condition shown below. If it does, it requests the mediator to store it into a temporary relation in the action clause. If I-sequence relation I_i is the master in the specification, it sets the timer alarm to raise an alarm event which invokes the relevant generation rule.

The filtering condition in the condition clause should check whether the new tuple will be used in the future. When we get the expression

$$E'(\sigma_{C_1}(\sigma_{I_1.ITS \in \psi_1(t)}(I_1)), \cdots, \sigma_{C_n}(\sigma_{I_n.ITS \in \psi_n(t)}(I_n)))$$

in the above step 2,

$$\sigma_{C_i}(\sigma_{I_i.ITS \in \psi_i(t)}(I_i))$$

can be used to derive the filtering condition. For a tuple to be used in the future, it must satisfy the condition C_i. Also, it must meet the temporal condition

$$I_i.ITS \in TSet^+(\psi_i, now),$$

where "now" stands for the current time.

To summarize, the storage rule for I-sequence relation I_i can be derived as follows.

(i) Case 1: I_i is a master I-sequence relation.

Rule Storage$_i$
on: arrival(I_i)
if: $I_i.ITS \in TSet^+(\psi_i, now) \wedge C_i$
do: $Temp_{I_i}$ **+=** I_i;
$setTimer(f(I_i.ITS), new)$;

The $setTimer(Time, Name)$ operator sets the timer alarm for the time $Time$ to raise an alarm event named $Name$[2]. In the context of storage rules, I_i stands for the new tuple provided by the DIS wrapper rather than the I-sequence relation itself. However, we abuse this notation for simplicity.

(ii) Case 2: Otherwise.

Rule Storage$_i$
on: arrival(I_i)
if: $I_i.ITS \in TSet^+(\psi_i, now) \wedge C_i$
do: $Temp_{I_i}$ **+=** I_i;

Storage rules derived for the service specification given in Section 5 are as follows. They can be reduced to those shown in Section 4.

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

Rule Storage$_{INews}$
on: arrival(I_{INews})
if: $I_{INews}.ITS \in TSet^+(\psi_{INews}, now)$
do: $Temp_{I_{INews}}$ **+=** I_{INews};

7.1.3 Generation Rules

A generation rule is derived from the specification. Its invocation is triggered by the alarm from the timer module set in the storage rule for the master I-sequence relation. For the expression

$$E'(\sigma_{C_1}(\sigma_{I_1.ITS \in \psi_1(t)}(I_1)), \cdots, \sigma_{C_n}(\sigma_{I_n.ITS \in \psi_n(t)}(I_n)))$$

obtained in the step 2, it requests the mediator to execute the expression

$$\Omega_{f(I_k.ITS)}(E'(\sigma_{Temp_{I_1}.ITS \in \psi_1(now)}(Temp_{I_1}), \cdots, \sigma_{Temp_{I_n}.ITS \in \psi_n(now)}(Temp_{I_n}))).$$

Thus, the generation rule can be specified as follows.

[2]If the function f is *immediate*, the alarm event should be raised immediately. In such a case, *setTimer(Immediate, Name)* is used, but more details are omitted.

Rule Generation$_{new}$
on: alarm(new)
if: true
do: $O_{new} = \Omega_{f(I_k.ITS)}($
$E'(\sigma_{Temp_{I_1}.ITS \in \psi_1(now)}(Temp_{I_1}), \cdots, \sigma_{Temp_{I_n}.ITS \in \psi_n(now)}(Temp_{I_n})));$
$Deliver(O_{new});$

The generation rule derived for the service specification given in Section 5 is as follows.

Rule Generation$_{new}$
on: alarm(new)
if: true
do: $O_{new} = Temp_{SPrice}$
$\bowtie_{CName=Name}$
$\wedge previous_{*:*:*:0:0:0}(before_{0:0:1:0:0:0}(I_{SPrice}.ITS)) \leq previous_{*:*:*:0:0:0}(I_{INews}.ITS)$
$\wedge previous_{*:*:*:0:0:0}(I_{INews}.ITS) \leq previous_{*:*:*:0:0:0}(I_{SPrice}.ITS)$
$Temp_{INews}$
$\bowtie_{Name=Company \wedge Price \geq Threshold}$
$StockholderInfo;$
$Deliver(O_{new});$

7.1.4 Garbage Disposal Rules

In the expression

$$E'(\sigma_{C_1}(\sigma_{I_1.ITS \in \psi_1(t)}(I_1)), \cdots, \sigma_{C_n}(\sigma_{I_n.ITS \in \psi_n(t)}(I_n)))$$

obtained in the step 2, if I_i may contain disposable data under ψ_i, then a garbage disposal rule for I_i is derived.

The garbage disposal rule is invoked by alarm events periodically raised from the timer module every time interval INT. We assume that the system administrator determines the time interval INT. In the condition clause, it checks the following condition

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

If it holds, it executes the action clause. In the action clause, it throws away tuples which will be never used in the future as follows:

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

Thus, if I_i may contain disposable data, a garbage disposal rule is derived as follows.

Rule GarbageDisposal$_i$
on: alarm($GarbageDisposal_i$)
if: $TSet^+(\psi_i, now - INT) - TSet^+(\psi_i, now) \neq \phi$
do: $Temp_{I_i}$ -= $\sigma_{Temp_{I_i}.ITS \in TSet^+(\psi_i,\ now-INT) - TSet^+(\psi_i,\ now)}(Temp_{I_i})$;
$setTimer(now + INT, GarbageDisposal_i)$;

In the example scenario, we get $\psi_{SPrice}(t)$ and $\psi_{INews}(t)$ for I-sequence relations I_{SPrice} and I_{INews} as shown in Subsection 7.1.1. Since both

$$\begin{aligned}
&TSet^+(\psi_{SPrice}, t - INT) - TSet^+(\psi_{SPrice}, t) \\
&= \{u | previous_{*:*:*:0:0:0}(t - INT) \leq u \wedge u < previous_{*:*:*:0:0:0}(t)\}, \\
&TSet^+(\psi_{INews}, t - INT) - TSet^+(\psi_{INews}, t) \\
&= \{u | previous_{*:*:*:0:0:0}(before_{0:0:1:0:0:0}(previous_{*:*:*:0:0:0}(t - INT))) \leq u \\
&\quad \wedge u < previous_{*:*:*:0:0:0}(before_{0:0:1:0:0:0}(previous_{*:*:*:0:0:0}(t)))\}
\end{aligned}$$

are not always empty, they may contain disposable data. Therefore, garbage disposal rules for I_{SPrice} and I_{INews} are derived as follows.

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

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

7.2 Rule Generation for Multiple Service Specifications

In the previous subsection, we assumed that we have only one service specification. In this section, we discuss the case we have multiple consistent service specifications $S_1, \cdots, S_m$ defining O-sequence relations $O_1, \cdots, O_m$, respectively, on top of I-sequence relations $I_1, \cdots, I_n$. An I-sequence relation I_i may be referenced in more than one specifications. In the following, we assume that the expression

$$E'^j(\sigma_{C_1^j}(\sigma_{I_1.ITS \in \psi_1^j(t)}(I_1)), \cdots, \sigma_{C_n^j}(\sigma_{I_n.ITS \in \psi_n^j(t)}(I_n)))$$

is obtained by the step 2 in Subsection 7.1.1 from the service specification S_j.

7.2.1 Storage Rules

A storage rule is derived for each I-sequence relation I_i that is referenced in at least one specification. The derivation is similar to that explained in Subsection 7.1.2. A difference is that the filtering condition should take care of all selection conditions

$$\sigma_{C_i^j}(\sigma_{I_i.ITS \in \psi_i^j(t)}(I_i))$$

for I_i. Another difference is that the timer alarm should be set to trigger invocation of the generation rule for each S_j that references I_i as the master I-sequence relation.

The storage rule for I_i is given as follows.

Rule Storage$_i$
on: arrival(I_i)
if: $tag(I_i, \bigvee_j (I_i.ITS \in TSet^+(\psi_i^j, now) \wedge C_i^j))$
do: $Temp_{I_i}$ += I_i;
For each S_j that references I_i as the master,
if j is included in $I_i.Tag$
$setTimer(f_j(I_i.ITS), j)$;

The range of the index j in the disjunction $\bigvee_j(...)$ is the set of indexes of S_j's which reference I_i. The operator $tag(I_i, \bigvee_j(...))$ adds the Tag attribute to the tuple in I_i. The assigned attribute value is the set of index value j such that the tuple satisfies the condition

$$I_i.ITS \in TSet^+(\psi_i^j, now) \wedge C_i^j.$$

It returns the truth value of $\bigvee_j(...)$.

7.2.2 Generation Rules

A generation rule is derived for each service specification S_j. In this context, each temporary relation may include tuples which are used to derive data for different services. Therefore, in the generation rule for specification S_j, we have to extract data for S_j. This can be done by checking the Tag attribute values.

Rule Generation$_j$
on: alarm(j)
if: true
do: $O_j = E'^j(\pi^*(\sigma_{j \in Tag}(Temp_{I_1})), \cdots, \pi^*(\sigma_{j \in Tag}(Temp_{I_n})))$;
$Deliver(O_j)$;

Here, π^* stands for the projection operation to eliminate the Tag attribute.

7.2.3 Garbage Disposal Rules

A garbage disposal rule is derived for I_i, if it may contain disposable data in the context of at least one service specification. Data becomes garbage when nobody is going to use it. The garbage disposal rule for I-sequence relation I_i is derived as follows.

Rule GarbageDisposal$_i$
on: alarm($GarbageDisposal_i$)
if: $(\bigcap_j (TSet^+(\psi_i^j, now - INT) - TSet^+(\psi_i^j, now))) \neq \phi$
do: $Temp_{I_i} \mathrel{-}= \sigma_{Temp_{I_i}.ITS \in \bigcap_j (TSet^+(\psi_i^j, now-INT) - TSet^+(\psi_i^j, now))}(Temp_{I_i})$;
$setTimer(now + INT, GarbageDisposal_i)$;

8 Related Works

In this paper, we have proposed a framework for specifying new information delivery services integrating existing dissemination-based information sources and automatically deriving ECA rules from the specification.

Infogate/EntryPoint/PointCast [8, 17] and *Castanet* [11] are examples of commercial dissemination-based information sources. Although some of those sources provide the information filtering service, they do not have facility for information integration.

DBIS [1, 2] and *Muffin* [16] aim to extract information from multiple dissemination-based information sources. DBIS uses *Information Broker* to access multiple dissemination-based sources transparently and to select information based on user profiles. Muffin can create a new *virtual channel* based on the user profile. To select information from multiple information services, it uses frequency of delivery, freshness, and popularity as well as similarity. These two researches only consider information selection from multiple dissemination-based information sources, and do not consider more sophisticated integration involving dissemination-based sources or other traditional information sources.

In *SADB* [20], ECA rules are used to manipulate data coming from dissemination-based information sources. However, they do not show declarative requirement specification or rule generation schemes.

OpenCQ [10] is an information integration system for distributed heterogeneous information sources. This system is based on the event-driven approach and supports continual queries. A *continual query* consists of three components, a query, a trigger condition, and a termination condition. When a trigger condition becomes true, this system repeatedly executes the query until the stop condition holds. Every time, the difference between the current query execution and

the previous one is reported as a result. Their work is related to our approach, since it follows an event-based approach in the context of information integration. However, OpenCQ considers only a particular case of the dissemination-based delivery of the integration results. Moreover, its mediator cannot store integration results. Therefore, OpenCQ cannot support integration of dissemination-based information sources or complex delivery requirements shown in this paper. Moreover, the users have to write continual queries directly.

Tapestry [21] also supports continual execution of queries for append-only relational databases. A continual query in Tapestry is an SQL query including time conditions in the WHERE clause. Like OpenCQ, this system reports the difference from the previous result to the user. Although an append-only database can be seen as a dissemination-based information sources, the queries allowed in Tapestry are limited to special cases. Moreover, Tapestry does not provide event-driven data processing or information integration facilities.

Production rules and *incremental algorithms* are proposed to be used for maintenance of materialized views [3, 4, 7, 13]. Those works are related with ours, since O-sequence relations introduced in this paper could be regarded as views on top of I-sequence relations. Automatic rule derivation from view definitions is discussed in [4]. However, in the context of materialized view maintenance, they do not consider temporal properties such as arrival and delivery times, temporal dependencies, or timestamp functions. Processes such as garbage disposal are not considered, either. In addition, in [4], tuples for view relations are generated as soon as base relations are updated. In contrast to this, tuples for O-sequence relations are generated at the scheduled delivery time. [13] proposes a scheme to delete tuples in base relations which do not contribute to materialization of view relations. The work is related with the derivation of garbage disposal rules in our scheme. However, they do not consider temporal properties, either.

SEQ [19] is a data model for sequence data. Sequence relations in this paper also model sequences of delivery units. SEQ is based on the relational model, but a number of new operators are introduced. In this paper, we have slightly extended the original relational algebra with operators to cope with timestamps.

9 Conclusion

In this paper, we have discussed a scheme to specify new information delivery services integrating multiple dissemination-based information sources. We have assumed the mediator/wrapper-based information integration system architecture, in which ECA rules are employed to implement event-driven data storage, integration, and delivery operations. In the proposed approach, the relational model is used as a common data model. Existing dissemination-based information sources

are modeled as I-sequence relations, and data to be delivered through new information delivery services are modeled as O-sequence relations. Then, relational algebra-based specifications are used to define O-sequence relations on top of I-sequence relations. In addition, we have discussed some important properties of the specification, and shown how to derive ECA rules from the algebraic service specifications.

The proposed scheme is being implemented on the prototype InfoWeaver system. The algebraic specification provides a sound basis for more declarative and user-friendly specification schemes. They include schemes based on SQL and GUI. Development of such facilities is one of the future research issues. They also include derivation of rules for more sophisticated data management. In this paper, we have employed storage rules, generation rules, and garbage disposal rules. Rules may be used for deriving intermediate data. We also have to cope with cases where the scheduled delivery time is determined in a more complicated manner.

Acknowledgments

This research is supported in part by the Grant-in-Aid for Scientific Research from the Ministry of Education, Science, Sports and Culture, Japan.

References

[1] D. Aksoy, M. Altinel, R. Bose, U. Cetintemel, M. Franklin, J. Wang, and S. Zdonik. Research in Data Broadcast and Dissemination. *Proc. AMCP '98*, pp. 194–207, 1998.

[2] M. Altinel, D. Aksoy, T. Baby, M. Franklin, W. Shapiro, and S. Zdonik. DBIS Toolkit – Adaptable Middleware for Large-Scale Data Delivery. *Proc. ACM SIGMOD '99*, 1999.

[3] J. A. Blakeley, P. A. Larson, and F. W. Tompa. Efficiently Updating Materialized Views. *Proc. ACM SIGMOD '86*, pp. 61–71, May 1986.

[4] S. Ceri and J. Widom. Deriving production rules for incremental view maintenance. *Proc. 17th VLDB*, 1991.

[5] R. Domenig and K. R. Dittrich. An Overview and Classification of Mediated Query Systems. *ACM SIGMOD Record*, 28(3), pp. 63–72, 1999.

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

[7] E. N. Hanson. A performance analysis of view materialization strategies. *Proc. ACM SIGMOD '87*, pp. 440–453, 1987.

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

[9] 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.

[10] L. Liu, C. Pu, and W. Tang. Continual Queries for Internet Scale Event-Driven Information Delivery. *IEEE TKDE*, 11(4), pp. 610–628, 1999.

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

[12] 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.

[13] H. G. Molina, W. J. Labio, and J. Yang. Expiring Data in a Warehouse. *Proc. 24th VLDB*, 1998.

[14] 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.

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

[16] 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.

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

[18] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access Path Selection in a Relational Database Management System. *Proc. ACM SIGMOD '79*, pp. 23–34, 1979.

[19] P. Seshadri, M. Livny, and R. Ramakrishnan. SEQ: A Model for Sequence Databases. *Proc. ICDE*, pp. 232–239, 1995.

[20] T. Terada, M. Tsukamoto, and S. Nishio. Design and Implementation of an Active Database System for Receiving Broadcast Data. *Journal of IEICE*, J83-D-I(12), pp. 1272–1283, 2000.

[21] D. B. Terry, D. Goldberg, D. Nichols, and B. M. Oki. Continuous Queries over Append-Only Databases. *Proc. SIGMOD '92*, pp. 321–330, 1992.