Dynamic Creation of Multimedia Web Views on Heterogeneous Information Sources

Atsuyuki Morishima†, Hiroyuki Kitagawa†, Hironori Mizuguchi††, and Seiichi Koizumi††
†Inst. of Info. Sci. and Elec., Univ. of Tsukuba ††Doctoral Program in Eng., Univ. of Tsukuba
1-1-1 Tennohdai, Tsukuba, Ibaraki 305-8573, Japan
{mori, kitagawa, hironori, izumi}@dblab.is.tsukuba.ac.jp

Abstract

Integration of the World Wide Web and other information sources is strongly required in the recent advanced application environments. Although information sources contain various types of data objects, the volume of multimedia objects has been increasing drastically. This paper proposes a multimedia integration scheme using SMIL in a mediator-based integration system. The scheme achieves dynamic creation of various Web views combining multimedia objects stored in heterogeneous information sources such as the Web, structured documents, and databases. A key point is to utilize SMIL as a framework for dynamic multimedia integration. Since there already exist some players for SMIL documents, dynamic integration results including multimedia objects can be rendered in such players, without special-purpose ones, and are ready to be broadly distributed. The main contributions of the paper are the followings. (1) We propose a dynamic, SMIL-based multimedia integration scheme. This scheme utilizes an extended relational data model named WebNR/SD. (2) We report on implementation of the proposed scheme. We have already implemented the first prototype system which realizes integration of heterogeneous information sources including multimedia objects. Actually, it can construct SMIL-based multimedia views dynamically. This shows the practical significance of our scheme.

1. Introduction

Rapid advances of computer network technologies make more and more information sources accessible from our computers. Therefore, integration of heterogeneous information sources has been one of the most important issues in recent advanced application environments. In particular, with the broad acceptance of the World Wide Web (or Web), integration of the Web and other information sources has been strongly required. Moreover, the powerful CPUs and the large storage space of computers today has made manipulation of multimedia objects such as video and audio much easier. Thus, the volume of multimedia data has been increasing drastically.

In this paper, we propose a dynamic multimedia integration scheme using SMIL (Synchronized Multimedia Integration Language) [27]. SMIL is a XML-based language for multimedia integration on the Web, and is currently a W3C recommendation. In the proposed scheme, the user can create various Web views combining multimedia objects stored in heterogeneous information sources such as the Web, structured documents (i.e., XML and SGML documents), and databases. The view creation is done by dynamic generation of SMIL-based hypertext structures according to specifications to amalgamate heterogeneous information sources.

A key point is *to utilize SMIL as a framework for dynamic multimedia integration*. There already exist some players for SMIL documents, such as GRiNS [7] and RealPlayer G2 [19]. Therefore, dynamic integration results including multimedia objects can be rendered in such players, without special-purpose ones. The results are ready to be broadly distributed.

Figure 1 shows our integration environment. We call it *InfoWeaver* [14][16]. The mediator [25] and wrappers [20] are used for integration. This is one of the promising approaches to integrate heterogeneous information sources. Many integration systems follow this approach [8][18][20]. In this approach, the mediator serves as a coordinator. It first dispatches wrappers to information sources. Integration of information sources is attained as follows. First, wrappers provide the mediator with schema information on local information sources. All the schema information constitutes the integrated schema, and is shown to the client. If users submit a data manipulation request to the mediator, the mediator analyzes the request, decomposes it into local processing requests, and sends them to wrappers. Each wrapper translates requests into local commands, and issues them to the local information source. Local commands may be simple requests to get particular data items such as Web page fetches, or complicated queries which utilize the querying capability of local information sources, such as SQL queries. The wrapper receives the intermediate result, translates it into the common data model (in our environment, WebNR/SD data model), and sends it back to the mediator. Finally, the mediator collects data from the wrappers and produces the final result, which is returned to the client.

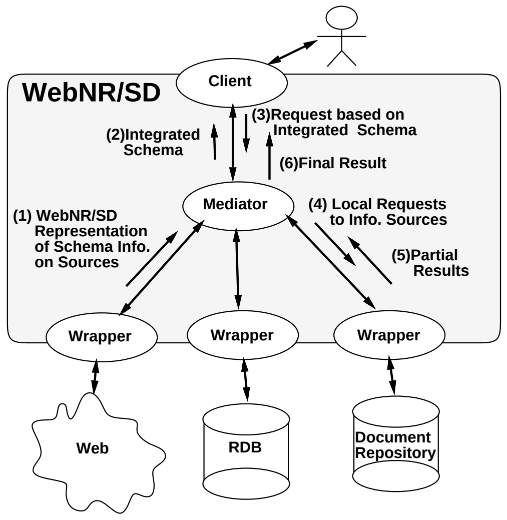
WebNR/SD
Client
(2)Integrated Schema
(3)Request based on Integrated Schema
(6)Final Result
Mediator
(1) WebNR/SD Representation of Schema Info. on Sources
(4) Local Requests to Info. Sources
(5)Partial Results
Wrapper
Wrapper
Wrapper
Web
RDB
Document Repository

Figure 1. Integration environment InfoWeaver

To construct SMIL views as the result of dynamic integration, dynamic manipulation of structured documents is indispensable. In our environment, we achieve this by using an extended relational data model named WebNR/SD. WebNR/SD is based on the nested relational data model and an abstract data type for structured documents named the *SD type*.

The features of WebNR/SD are as follows. (1) It realizes amalgamation of heterogeneous information sources. WebNR/SD can deal with relations in RDBs and structured documents such as SGML documents and HTML/XML Web pages. Moreover, WebNR/SD can integrate multimedia objects, by manipulating their references (or anchors) in structured documents. Special operators named *converters* are used for amalgamation. They translate relational structures into structured documents and vice versa. (2) It provides a number of functions to manipulate structured documents. A number of text retrieval functions are associated with the SD type. Operators for manipulation of tag attributes in structured documents are also provided.

The main contributions of the paper are the following. (1) We propose a *dynamic, SMIL-based* multimedia integration scheme. This scheme utilizes WebNR/SD. (2) We report on implementation of the proposed scheme. We have already implemented the first prototype system which realizes integration of heterogeneous information sources including multimedia objects. Actually, it can construct SMIL-based multimedia views dynamically. This shows the practical significance of our scheme.

The rest of this paper is organized as follows. Section 2 explains an example of dynamic creation of SMIL-based multimedia Web views, and shows the issues to be considered. Section 3 describes WebNR/SD. Section 4 shows how the SMIL view creation is realized by WebNR/SD. Section 5 explains the prototype system development. Section 6 briefly surveys related works. Section 7 is the conclusion.

2. Dynamic Creation of SMIL-based Multimedia Web Views

2.1. Example

In this subsection, we show an example of SMIL-based dynamic integration in InfoWeaver. We consider two relational databases and the Web as information sources. (1) *A baseball game video database*: This is a relational database which contains video (RealMedia) objects and their metadata. The video objects are stored in the relation V-DATA(<u>VID</u>, Content, MIME-Type, Game, Inning). The domain of the attribute Content is an ADT, VIDEO type. Each VIDEO value in Content records an inning of a game. The metadata is stored in GAME(<u>GID</u>, Date, Team1, Team2, Score) and V-INDEX(<u>Video, Begin, End</u>, Batter, Pitcher). GAME contains the information about games. V-INDEX contains the information about what batter was at bat and who was pitching in the designated duration (represented by Begin and End) in the Video (Video is a foreign key on V-DATA). (2) *A baseball statistics database*: This is a relational database which maintains the latest statistics about baseball players. This database contains the relation BATTING-STATS(<u>P-Name</u>, Hit, RBI, AVG). (3) *A baseball players' profile Web site*: This site contains the profile information of baseball players. The Web-site structure is shown in Figure 2. The index page contains links to baseball team pages. Each team page contains the team logo (as a reference to a GIF format file) and links to player pages. Each player page contains the profile data. The profile data may have hypertext links to other Web pages such as the home page of the school from which the player graduated.

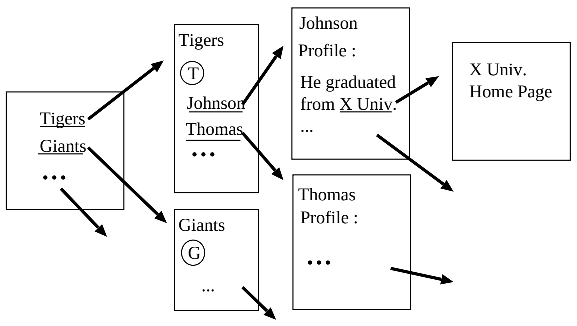

Figure 2. Baseball players' profile Web-site

The requirement here is to create a multimedia Web view on top of the above information sources (Figure 3). A Web page is constructed for each player whose hitting average is more than .300 for the current season. It is a *multimedia page* (Figure 3(a)). It consists of three different kinds of data objects: (1) *A digested video* collecting only the scenes in which he was at bat in the previous day's game. (2) *The* *logo of the team the player belongs to.* (3) *Text description of his profile,* which may have hypertext links to other Web pages.

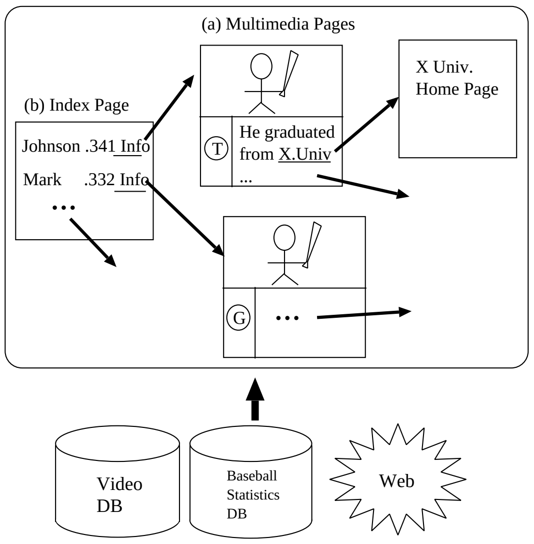

Figure 3. Multimedia Web view

An index page is also created (Figure 3(b)). It contains the players' names, hitting averages, and links to the players' multimedia pages.

2.2. SMIL

As mentioned before, our integration scheme uses SMIL as a basis for multimedia integration. SMIL (Synchronized Multimedia Integration Language) can represent integrated multimedia contents as XML-based tagged text. Figure 4 is an example SMIL document. This represents a multimedia page explained in Subsection 2.1. A SMIL document must start with the tag `<smil>` and end with `</smil>`.

The document has two main parts. First, the head part specifies the layout of the multimedia document. In this example, the layout part says the document contains three rectangular regions. Attributes of tags are specified in the following way. The `id` attribute is the identifier of the region. The `left` and `top` specify the top-most and left-most positions, respectively. The `height` and `width` specify the size.

Then, the body part specifies how to present multimedia objects in the layout. This example contains two videos, one image, and one textstream. The `<video/>`, `<img/>`, and `<textstream/>` tags[1] represent references to the video, image, and textstream objects [2], respectively. Figure 5 elaborates on attributes of tags in the body part of this example. Note that some URLs in the sample SMIL document start with "rtsp." RTSP (Real Time Streaming Protocol) is an IETF proposed standard for control of streaming media in the Internet [21]. In this example, RealMedia(.rm) and RealText(.rt) are streaming media. RTSP is currently supported by RealServer [19]. Also, the World Wide Web consortium is developing Jigsaw, a Web server which is expected to support RTSP in the future [26].

Tags `<seq>` and `<par>` give synchronization information. The multimedia objects directly surrounded by the `<seq>` tag are presented in a sequential way. On the other hand, the `<par>` tag specifies that objects are presented in parallel. Therefore, the sample SMIL document specifies that the multimedia presentation shows in parallel a sequence of two video clips, an image object, and a textstream object.

```
<smil>
<head>
<layout>
<root-layout height="373" width="506"/>
<region id="R1" left="0" top="0"
                height="215" width="346"/>
<region id="R2" left="0" top="216"
                height="157" width="167"/>
<region id="R3" left="168" top="216"
                height="155" width="338"/>
</layout>
</head>
<body>
<par>
  <seq>
    <video src="rtsp://...Inning_1.rm" region="R1"
           clip-begin="300.00s" clip-end="310.00s"/>
    <video src="rtsp://...Inning_3.rm" region="R1"
           clip-begin="205.00s" clip-end="223.00s"/>
  </seq>
  <img src="http://...Logo1.gif" region="R2"/>
  <textstream src="rtsp://...Profile1.rt"
              region="R3"/>
</par>
</body>
</smil>
```

Figure 4. Sample SMIL file

`src`	the URL of the referenced data object
`region`	the region where the data object is rendered
`clip-begin`	the beginning time the video object is clipped
`clip-end`	the ending time the video object is clipped

Figure 5. Tag attributes in the body part of the example

2.3. Issues to be Considered

The above scenario is a prospective example of SMIL-based multimedia integration and Web view creation. To realize this, the integration environment is required to provide the following functions: (1) to give an integrated schema for relational databases and the Web, (2) to amalgamate heterogeneous information sources, and (3) to manipulate document structures and tag attributes for construction of SMIL documents.

[1]The `<g/>` is called *an empty-element tag*. It is semantically equivalent to `<g></g>`.

[2]In SMIL, text and textstream objects are stored in separate data files (in this example, Profile1.rt, where rt means that this file is in RealText format [19], a textstream object), like video and image objects.

3. WebNR/SD

In our scheme, the functions in Subsection 2.3 are realized by the data model WebNR/SD. The model presented here is obtained by extending our previous model in [14]. The major extension is to make it possible to explicitly handle tag attributes. For this purpose, we introduce tag attribute operators. As shown before, tag attribute handling is crucial in construction of SMIL documents. We focus on issues necessary for explaining the multimedia integration scenario, and omit some details which are not directly related with this context. A more detailed description of WebNR/SD except for the new tag attribute operators is given in [14].

WebNR/SD is a data model which introduces the abstract data type concept into nested relations. It has an abstract data type named the *SD type* (structured document type) to deal with raw structured documents (SGML, XML, and HTML documents). Figure 6 shows a sample relation in WebNR/SD. The domains of attributes A and D are String and Integer, respectively. The domain of B and E is the SD type [3]. We call values of the SD type *SD values*. In particular, we call an SD value consisting of only one referencing element, such as an anchor in HTML and a `<video/>` tag in SMIL, an *H-link value*.

A	B	C	
		D	E
abc	“`<table><dep> Department...`”	1	“`<a href="http://..">...`”
		2	...
def	...	3	...
		4	...

Figure 6. Sample relation in WebNR/SD

WebNR/SD provides the nested relational algebra operators (Figure 7), and a number of functions associated with the SD type to retrieve text elements contained in documents. Also, WebNR/SD has operators, called *converters*, to dynamically convert structured documents into nested relational structures and vice versa.

Moreover, WebNR/SD has a number of operators for Web integration. The *Import* operator is used to fetch the contents of Web pages as SD values from the outside Web world on demand. In contrast, The *Export* operator exports SD values stored in relations as Web pages to the outside Web world. The *Navigate* operator is used to realize navigational queries (explained in Subsection 3.3).

Integration of the Web, structured documents and relational databases is achieved by combining the operators.

3.1. Tag Attribute Operators

To deal with tag attributes of structured documents, tag attribute operators **RA** and **WA** are provided. They make the model presented in this paper an extension of the original one in [14]. Figure 8 gives examples of **RA** and **WA**, where

[3]Actually, the SD type can also deal with structured documents with DTD [14]. In this paper, this aspect is ignored for simplicity.

Selection	$\sigma_p(r)$
Projection	$\pi_{A_{i1},\ldots,A_{im}}(r)$
Cartesian product	$r_1 \times r_2$
Nest	$\nu_{B=(A_{i1},\ldots,A_{im})}(r)$
Unnest	$\mu_B(r)$
Union	$r_1 \cup r_2$
Difference	$r_1 - r_2$

Figure 7. Nested relational algebra operators

$$r_2 := \mathbf{RA}_{B\to(src:C)}(r_1),$$

and

$$r_2 := \mathbf{WA}_{(src:C)\to B}(r_3).$$

$\mathbf{RA}_{B\to(src:C)}(r_1)$ extracts values of the tag attribute `src` from SD values contained in B. Only attribute values of the outermost tags are extracted. The extracted values are stored in C. The domain of C is String. On the other hand, $\mathbf{WA}_{(src:C)\to B}(r_3)$ adds attributes to the outermost tags of SD values in Attribute B. Values in C are used as values of tag attribute `src`. In case that SD values in B of r_3 already have tag attribute `src`, the previous values are overwritten by values in C. If a value in C is ’#’, the corresponding tag attribute itself is deleted [4].

Let ’V’$_{(B)}$ denote a unary relation with attribute B containing an attribute value `'V'`. We represent $\mathbf{WA}_{(a:A)\to B}(r \times$ ’V’$_{(B)})$ as $\mathbf{WA}_{(a:A)\to B:'V'}(r)$, and $\mathbf{WA}_{(a:V)\to B}(r \times$ ’V’$_{(V)})$ as $\mathbf{WA}_{(a:'V')\to B}(r)$, for notational convenience. Moreover, we represent $\mathbf{RA}_{B\to(a_1:A_1)}(\ldots\mathbf{RA}_{B\to(a_n:A_n)}(r)\ldots)$ as $\mathbf{RA}_{B\to(a_1:A_1,\ldots,a_n:A_n)}(r)$. This rule applies to **WA**, too.

r_1:

A	B
1	“`<img src="logo1.gif" region="R2"/>`”

r_2:

A	B	C
1	“`<img src="logo1.gif" region="R2"/>`”	Logo1.gif

r_3:

A	B	C
1	“`<img region="R2"/>`”	Logo1.gif

Figure 8. Examples of WA and RA

3.2. Converters

Converters are operators to translate nested relational structures into structured documents and vice versa. Explained here are some *composite converters*. They are defined as combinations of primitive converters and other primitive operators such as nested relational algebra operators [14]. Converters are classified into *pack operators* and *unpack operators*.

[4]Of course, you can use the character ’#’, with the escape character.

Pack Operators

Pack operators construct new SD values from sub-relation structures containing SD values. Figure 9 gives an example of a composite pack operator, where

$$r_5 := \mathbf{P}_{T(Player) \rightarrow Table,'g'}(r_4).$$

This constructs new SD values in the attribute $Table$ from sub-relations in attribute T of r_4. The parameter g is used as the outermost tags of new SD values in the attribute $Table$.

Figure 9 also gives an example of another composite pack operator, where

$$r_7 := \mathbf{P}_{(Digest,IMG,PF) \rightarrow Info,'par'}(r_6).$$

This pack is different from the previous one in parameter specification. In general, it takes the form of $\mathbf{P}_{(A_1,\ldots,A_n) \rightarrow A,g}(r)$. It constructs new SD values from sequence structures of relational attributes.

Unpack Operators

Figure 10 shows a composite unpack operator which constructs new sequence structures of relational attributes from SD values, where

$$r_9 := \mathbf{U}_{B \rightarrow (C[p\text{-}name]\ as\ x, D[p]\ as\ y)}(r_8).$$

This example constructs new sequence structures of relational attributes (consisting of attributes C and D) in the result relation r_9. Attributes C and D includes SD values representing text elements which are extracted from SD values of attribute B in r_8. The SD values in attribute C and attribute D represent text elements surrounded by `<p-name>` and `<p>` tags, respectively [5].

SD values in attribute B of relation r_9 contain *SD references*, denoted by "&x.m;" and "&y.n;." SD references refer to SD values stored in the sub-relation structures, and are used by some pack operators. Further explanation is found in [14].

r_4:

A	T
	Player
1	"<player>P1</player>"
	"<player>P2</player>"
	"<player>P3</player>"

r_5:

A	Table
1	"<g><player>P1</player> <player>P2</player> <player>P3</player></g>"

r_6:

A	Digest	IMG	PF
1	"<seq>S1</seq>"	"<img src="..."/>"	"<text .../>"

r_7:

A	Info
1	"<par><seq>S1</seq> <img src="..."/> <text .../></par>"

Figure 9. Examples of Pack operators

[5]We use expressions of *the region algebra* [6] to specify which text elements are to be extracted by Unpack.

r_8:

A	B
1	"<profile-page>...<p-name>N1</p-name>... ...<p>Profile 1</p>...<profile-page>"

r_9:

A	B	C	D
1	"<profile-page> ...&x.1;... ...&y.1;... </profile-page>"	"<p-name>N1 </p-name>"	"<p>Profile 1 </p>"

Figure 10. Example of Unpack operator

3.3. Web-Related Operators

Export and Import Operators

Export (**e**) exports SD values stored in a relation to the Web world as new Web pages. Import (**i**) imports Web pages residing in the Web world into a relation as SD values[6].

Figure 11 gives an example of Export and Import, where

$$r_{11} := \mathbf{e}_{A,U}(r_{10}),$$

and

$$r_{10} := \mathbf{i}_{A,U}(r_{11}).$$

$\mathbf{e}_{A,U}(r_{10})$ creates Web pages whose URLs are given in attribute U. The Web pages' contents correspond to SD values stored in attribute A of r_{10}. $\mathbf{i}_{A,U}(r_{11})$ works in the opposite direction. Attribute A in the result relation r_{10} gets SD values corresponding to Web pages which are referred to by URLs stored in attribute U [7].

r_{10}:

ID	A	U
1	"<smil><head>...</head> <body>...</body></smil>"	http://T.ac.jp/p.smil

r_{11}:

ID	U
1	http://T.ac.jp/p.smil

Figure 11. Example of Export and Import

Navigate Operator

Navigate (**N**) realizes navigational queries. That is, this operator navigates through the Web according to a *path regular expression* specified as a parameter[8].

In path regular expressions, hypertext links are represented by $\rightarrow$ (a local link whose destination and source pages reside in the same Web server) and $\Rightarrow$ (a global link whose destination and source pages reside in different Web servers), while Web pages are represented by character strings (*labels*) and a period. For example, $B \rightarrow . \Rightarrow$

[6]The **e** and **i** are primitive versions of **E** and **I** in [14]. **E** and **I** can be defined as combinations of **e**, **i**, converters, and tag attribute operators.

[7]It is necessary that the contents of the Web pages be structured documents.

[8]The path regular expressions here are essentially the same as those of WebSQL queries [13].

$C \to D$ is a path regular expression that represents the set of paths that start with a Web page B in a Web server X, followed by a page in the same server X and a page C in a different server $Y(\neq X)$, and end with a page D in the server Y.

Path regular expressions can contain alternation and repetition structures. For example, $A(\to .)* \to B \mid A \Rightarrow B$ represents the set of paths which start with a page A, followed by one or more local links leading to B, or directly go from a page A to B via a global link. '$*/n_1..n_2/$' can be used as syntactic sugar. For example, $A(\to .)*/2..3/ \to B$ is equivalent to $A \to . \to . \to B \mid A \to . \to . \to . \to B$ [9].

Suppose that there is a hypertext link structure shown in Figure 12 in the Web. In this figure, each Ui stands for a URL, α, β, γ are Web server names, and lower-case letters show hypertext links. Figure 13 gives the result of the following Navigate operation:

$$r_{13} := \mathbf{N}_{A(\to .)*\to .\Rightarrow B\to C,D}(r_{12}).$$

Labels in path regular expressions are used to associate Web pages with relational attributes. The second parameter (in this case, D) specifies that attribute D of the resulting relation contains the set of paths specified by the path regular expression. As shown in Figure 13, note that (1) each page on the paths is represented by an H-link value referring to the Web page instead of the contents of the page itself, and that (2) links to those pages corresponding to the period do not appear in sub-relations of attribute D.

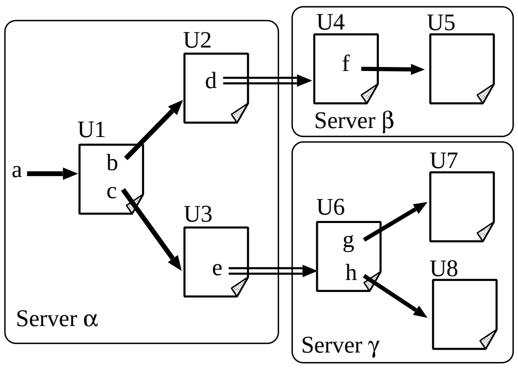

Figure 12. Example of hypertext link structure

URL generator

URL generator (**URL**) generates original URLs. It can be used before the Export operation in creating new Web pages. An example of **URL** is shown in Figure 14, where

$$r_{15} := \mathbf{URL}_{U,'http','smil'}(r_{14}).$$

This expression extends relation r_{14} by adding a new attribute U, which contains generated URLs.

[9]Path regular expressions can also specify word containment conditions on Web pages. Details are omitted here.

r_{12}:

ID	A
1	“<a href="U1">a</a>”

r_{13}:

ID	A	D	
		B	C
1	“<a href="U1"> a</a>”	“<a href="U4"> d</a>”	“<a href="U5"> f</a>”
		“<a href="U6"> e</a>”	“<a href="U7"> g</a>”
		“<a href="U6"> e</a>”	“<a href="U8"> h</a>”

Figure 13. Example of Navigate operator

r_{14}:

A	B
1	b1
2	b2

r_{15}:

A	B	U
1	b1	http://.../new/1.smil
2	b2	http://.../new/2.smil

Figure 14. Example of URL generator

4. Applying WebNR/SD to Multimedia Integration

This section explains how WebNR/SD realizes construction of the SMIL-based Web view in Subsection 2.1. The functions in Subsection 2.3 are realized in the following way.

(1) In the integrated schema, relations in relational databases are represented as they are. The Web is modeled as a collection of unary relations containing H-link values. For example, bookmarks in Web browsers can be used to construct such unary relations. The H-link values serve as anchor points to fetch necessary Web pages with Navigate and Import operators. In the case of the example in Subsection 2.1, we use the index page of the baseball players' profile Web site. The Web wrapper provides a unary relation “WEB” which contains only an H-link value referring to the index page (Figure 15). Note that in our context, we use the term of integrated schema to represent merely the collection of schema information from all information sources. Our framework can be used with some other frameworks for schema integration and conflict resolution such as that depicted in [23].

All of multimedia objects are translated into H-link values referring to the objects by wrappers. For example, values of the attribute Content in the relation V-DATA are translated into H-link values. (Therefore, the domain of the attribute becomes the SD type.) Our approach is flexible in that any types of multimedia objects can be accommodated as long as they make sense in SMIL.

(2) In the example scenario, Web pages, relations, image (GIF) files, and video (RealMedia) files should be amalgamated. As mentioned above in (1), they are basically represented as relations and SD values. Therefore, we can amalgamate and restructure them by unpack operators and nested relational algebra operators.

(3) WebNR/SD provides various operators for construction of structured documents and Web structures. We can use pack operators, tag attribute operators, and Web-related operators for construction of SMIL-based multimedia Web

views.

We show specification of the data integration scenario in Section 2 in WebNR/SD expressions. We assume that Web pages are written in XML[10] and have the following structures: (1) In each player's page, the player's name is enclosed by `<p-name>`, and the text description of the profile is tagged by `<p>`. (2) The reference to the team's logo image object in each team page is represented as `<img src="`URL`"/>` surrounded by `<t-logo>` tags. We also assume that default layouts for SMIL are provided in the form of unary relations. For example, Relation LAYOUT (with the attribute Head) in the following expression contains an SD value corresponding to the head part of the sample SMIL document in Figure 4. We derive the result index Web page with the tag structure as shown in Figure 16.

Page
“<a href="http://.../index.xml"></a>”

Figure 15. Relation WEB

```
<table>
<player>
<p-name>Johnson</p-name><avg>.305</avg>
<a href="http://....smil">info</a>
</player>
<player> ... </player>
...
</table>
```

Figure 16. Structure of the result index page

In the following expressions, we omit tags and SD references in parameters, when there is no possibility of confusion. For example, $\mathbf{U}_{E\to(P\text{-}Name[p\text{-}name]\ as\ P\text{-}Name,P[p]\ as\ P)}(r)$ is simply represented as $\mathbf{U}_{E\to(P\text{-}Name,P)}(r)$. Also, $\mathbf{P}_{(P\text{-}Name,AVG,A)\to Player,'Player'}(r)$ is represented as $\mathbf{P}_{(P\text{-}Name,AVG,A)\to Player}(r)$.

(1) Team logos, players' names, and their profiles are extracted from the baseball players' profile Web-site.

$$\begin{aligned} r_{16} := \ & \mathbf{U}_{P_2\to(P\text{-}Name,P)}(\mathbf{U}_{P_1\to(T\text{-}Logo)}(\\ & \mathbf{i}_{P_2,U_2}(\mathbf{RA}_{E\to(href:U_2)}(\mathbf{i}_{P_1,U_1}(\mathbf{RA}_{D\to(href:U_1)}(\\ & \mu_C(\mathbf{N}_{Page\to D\to E,C}(WEB)))))) \end{aligned}$$

(2) Batters hitting more than .300 this season are selected and joined with the other information.

$$\begin{aligned} r_{17} := \ & \sigma_{AVG>0.3}(BATTING\text{-}STATS) \bowtie r_{16} \\ & \bowtie_{P\text{-}Name=Batter} V\text{-}INDEX \\ & \bowtie_{Video=VID} V\text{-}DATA \\ & \bowtie_{Game=GID} \sigma_{Date='...'}(GAME) \end{aligned}$$

(3) Digested videos for the players are constructed.

$$\begin{aligned} r_{18} := \ & \mathbf{P}_{D(Content)\to Digest,'seq'}(\nu_{D=(Content)}(\\ & \pi_{P\text{-}Name,AVG,Content,T\text{-}Logo,P}(\\ & \mathbf{WA}_{(clip\text{-}begin:Begin,clip\text{-}end:End,region:'R1')} \\ & \qquad\qquad _{\to Content}(r_{17})))) \end{aligned}$$

(4) RealText files[11] for profiles are constructed from the players' profile information. And they are exported to the Web space.

$$\begin{aligned} r_{19} := \ & \mathbf{e}_{\,Profile,U_3}(\mathbf{URL}_{U_3,'rtsp','rt'}(\\ & \mathbf{P}_{(P)\to Profile,'window'}(r_{18}))) \end{aligned}$$

(5) SMIL files for multimedia pages are constructed and exported.

$$\begin{aligned} r_{20} := \ & \mathbf{e}_{\,MP,U_4}(\mathbf{URL}_{U_4,'http','smil'}(\\ & \mathbf{P}_{(Head,Body)\to MP,'smil'} \\ & (LAYOUT \times \mathbf{P}_{(Info)\to Body}(\\ & \mathbf{P}_{(Digest,IMG,PF)\to Info,'par'}(\\ & \mathbf{WA}_{(src:U_3,region:'R3')\to PF:'<textstream/>'}(\\ & \mathbf{WA}_{(region:'R2')\to IMG}(\\ & \mathbf{U}_{(T\text{-}Logo)\to IMG}(r_{19}))))))) \end{aligned}$$

(6) The index page is constructed and exported.

$$\begin{aligned} r_{21} := \ & \mathbf{e}_{\,Table,U_5}(\mathbf{URL}_{U_5,'http','xml'}(\\ & \mathbf{P}_{T(Player)\to Table}(\nu_{T=(Player)}(\\ & \pi_{Player}(\mathbf{P}_{(P\text{-}Name,AVG,A)\to Player}(\\ & \mathbf{WA}_{(href:U_4)\to A:'<a>info</a>'}(r_{20})))))) \end{aligned}$$

5. Prototype Development

We have developed the first prototype system of InfoWeaver. Figure 17 shows the architecture of the prototype system. OpenText index server, the Web, Oracle8, and mSQL are connected as information sources.

The prototype system has been designed according to the following three principles.

Extensibility It is desirable that the integration system can accept new types of information sources and wrappers easily. Therefore, the prototype system is designed so that it would require little additional hard-coding in incorporating new wrappers into the system. Specifically, the mediator and wrappers in the prototype system are designed to have the same interface. And they *dynamically* exchange information about metadata and query processing power of information sources at the execution time.

[10] At present, most Web pages are written in HTML. However, we use XML pages here for simplicity because they can explicitly show their internal structure by user-defined tags. In order to apply our framework to HTML pages, we have to (1) add additional functions to the SD type for substring extraction, or (2) design Web wrappers to transform HTML into XML. For example, we can use grep-like functions for substring extraction. Also, we can adopt a scheme to transform HTML into XML such as that reported in [22]. But this aspect is beyond the scope of this paper.

[11] RealText files must start with `<window>` and end with `</window>`.

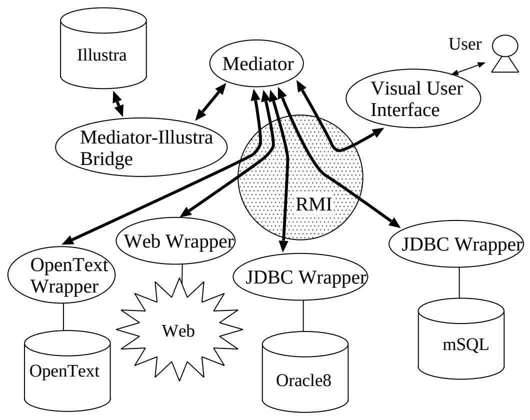

Figure 17. Architecture of the prototype system

Data Transfer On Demand In general, integration of information sources involves a large amount of data objects to be processed. Therefore, software modules such as the visual user interface, the mediator, and wrappers require a lot of working space in integration processes. In this prototype system, data objects are transferred among modules in a lazy fashion in order to reduce the working space and intermediate query processing costs. The transfer control is performed at the tuple level and value level.

Utilization of Existing Tools One of the goals of the first prototype development was to develop the system as quickly as possible. Since WebNR/SD consists of existing and well-known data modeling constructs, such as nested relational structures and the abstract data type concept, there are many programs and systems available for their manipulation. We used a number of existing software products. For example, the prototype system uses ORDBMS Illustra as a back-end engine to implement the mediator.

All codes are written in Java except the mediator-Illustra bridge module which is written in C. This is because Illustra had no API for Java applications. Communication among the modules is attained by RMI. The visual user interface module is implemented as a Java applet so that the system can be operated at any place where the Web browser is available.

Illustra serves as a back-end engine of the mediator. It directly executes relational algebra operators in WebNR/SD expressions and partially executes some other operators such as nest and unnest.

The JDBC wrapper communicates with a relational database through a JDBC driver. It translates WebNR/SD expressions including only relational algebra operators into SQL queries. The OpenText wrapper translates WebNR/SD expressions related to document manipulation into PAT [24] expressions.

The Web wrapper is different from the other wrappers in that it provides no means of query translation because the Web has no query processing capability. It is responsible for processing Web-related operators such as Navigate, Import, and Export.

As mentioned in Section 4, in the integrated schema, all multimedia objects are represented as H-link values to the objects, even if their contents are actually stored in relations as ADT values or BLOBs. Current implementation assumes that the contents of multimedia objects in relations are physically stored in an external file system, like the BFILE type of Oracle8 [17] (Figure 18). The JDBC wrapper constructs H-link values referring to the files, and provides the mediator with the view as if the H-link values were stored in the relations. The H-link values work as handles to the multimedia contents through the RealServer associated with the wrapper. We are now trying to develop a mechanism which can cope with multimedia objects directly stored in relations. This mechanism checks what multimedia objects are required in the data manipulation result, just after the result is constructed. Then, it warehouses their contents in the file system the mediator manages.

Figure 19 is a screen shot of the prototype system showing the SMIL-based Web view required in Subsection 2.1.

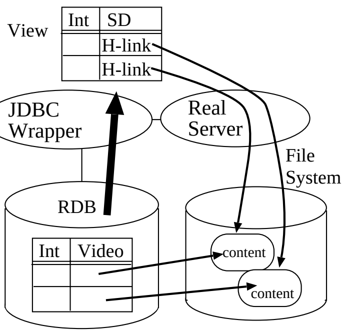

Figure 18. Construction of H-link values referring to multimedia objects

6. Related Work

Our proposed scheme can construct SMIL-based Web views dynamically on top of heterogeneous information sources. Since SMIL documents are XML-based structured documents, and required to have specified document structures and tag attributes, systems which can construct SMIL-based Web views must be able to manipulate detailed document structures.

Surprisingly, there are only a few implemented systems which meet such requirements. Strudel [8] is a Web-site management system. Like InfoWeaver, it can construct Web views on various information sources. An essential difference is that Strudel separates the construction process of Web-sites into three different steps and requires different specification languages, StruQL and the HTML tem-

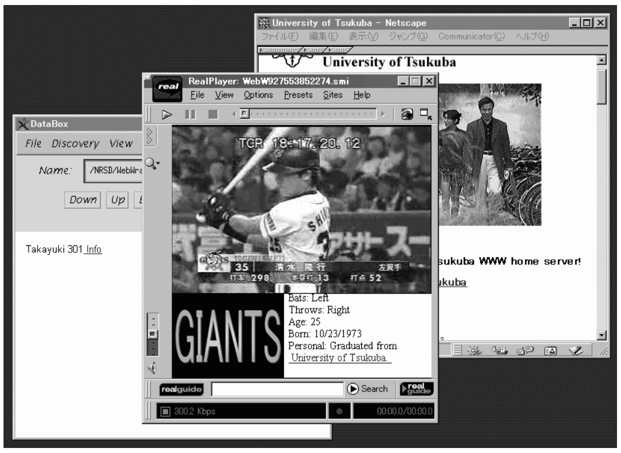

Figure 19. Screen shot of the prototype system

plate language. We think that such separation is not suitable for dynamic and ad-hoc integration scenarios as shown in Section 2. Also, Strudel provides asymmetric integration. It is only the Web view that Strudel can construct on heterogeneous information sources. InfoWeaver provides symmetric integration. For example, it can construct relational views over heterogeneous information sources. Another difference between Strudel and InfoWeaver is in their underlying data models. While InfoWeaver employs an extended relational data model incorporating ADTs, Strudel utilizes a variation of graph-based semistructured data models [1][5]. Data manipulation in Strudel and that in InfoWeaver are specified in a novel language named StruQL, and in an extended relational algebra named WebNR/SD algebra, respectively. We believe that utilization of extended relational models incorporating ADTs is a natural extension of the conventional data manipulation framework and that it is one of promising approaches to integration of heterogeneous data objects including multimedia objects. In fact, this approach is popular in commercial systems such as Oracle8 and Informix Universal Server, and also taken in the SQL3 standard, for integrating various kinds of data objects. T/RDBMS [4] is similar to our approach in that it combines the relational data model and the abstract data type representing structured documents for data integration. However, it cannot manipulate Web objects.

Although WebOQL [2] and Araneus [3] do not focus on integration of heterogeneous information sources, they can extract information from the Web and produce Web views. Like Strudel, WebOQL is based on a variation of graph-based semistructured data models. Araneus is similar to InfoWeaver in that it uses relational structures in the integration process. However, Araneus separates the view construction process into different steps and employs different languages.

Languages and formalisms such as W3QL [11], WebSQL [13], RAW [10], and WebLog [12] can query the Web based on hypertext link structures and/or pages' contents. However, their capabilities of exploiting pages' inner structures are generally very restricted. W3QL and WebSQL are SQL-like query languages, and RAW is an extension of the relational algebra. They are dedicated to querying and provide no way to restructure Web pages. WebLog is a logic-based language which enables us to query and restructure Web pages, exploiting various extensible built-in predicates. However, WebLog deals with Web pages at a rather abstract level, where they have only flat structures. A page in WebLog is defined as a set of *rel-infons*, each of which is a group of related information appearing on the page.

XSLT [29] and XML-QL [28] specified in a W3C working draft and a note, respectively, can restructure XML documents. To the best of our knowledge, there have not been implemented systems which can integrate heterogeneous information sources based on them.

7. Conclusion

In this paper, we have proposed a dynamic, SMIL-based multimedia integration scheme in a mediator-based information integration environment. The scheme utilizes an extended relational data model named WebNR/SD, which incorporates ADT named the SD type into nested relational structures. In WebNR/SD, the SD type provides the capability of accommodating various kinds of data objects. Combinations of converters, tag attribute operators, Web-related operators, and nested relational algebra operators enable data amalgamation, restructuring, and dynamic construction of SMIL-based multimedia Web views on heterogeneous information sources containing Web pages, relations, and multimedia objects such as video and image objects.

We have also reported on the implementation of the first prototype system. It works in distributed computing environments, and is available through the user interface module implemented as a Java applet.

Future works include the extension of the visual data manipulation language, HQBE [15], available in InfoWeaver. Although HQBE allows visual specification of data integration, the current version cannot deal with multimedia objects directly. We are now trying to combine authoring functions with querying functions in the visual environment for integration specification.

Another future work is on layout and multimedia synchronization in presentation. At present, layout information is specified by default layouts, and synchronization is explicitly specified by users. A goal is to develop an automatic or semi-automatic framework to construct such information based on data semantics and constraints.

Acknowledgement

This work has been supported in part by the Ministry of Education, Science, Sports and Culture, Japan, and Information Broadcasting Laboratories, Inc. The authors are grateful to Nippon Television Network Corporation for providing sample video data.

References

[1] Serge Abiteboul, "Querying semi-structured data," *Proc. 6th International Conference on Data Theory (ICDT'97)*, pp. 1-18, 1997.

[2] G. O. Arocena, A. O. Mendelzon, "WebOQL: Restructuring Documents, Databases, and Webs," *Proc. ICDE'98*, pp.24-33, 1998.

[3] P. Atzeni, G. Mecca, and P. Merialdo, "To Weave the Web," *Proc. VLDB'97*, pp. 206-215, 1997.

[4] G. E. Blake, M.P. Consens, P. Kilpeläinen, P. Larson, T. Snider, and F. Tompa, "Text/Relational Database Management Systems: Harmonizing SQL and SGML," *Proc. International Conf. on Applications of Databases*, no. 819 in Lecture Notes in Computer Science, Vadstena, Sweden, pp. 267-280, 1994.

[5] Peter Buneman, "Semistructured data," *Proc. PODS'97*, pp. 117-121, 1997.

[6] M. P. Consens and T. Milo, "Algebras for Querying Text Regions," *Proc. PODS'95*, pp. 11-22, 1995.

[7] Oratrix Development BV. "GRiNS home page," http://www.oratrix.com/GRiNS/.

[8] M. Fernández, D. Florescu, J. Kang, and A. Levy, "Catching the Boat with Strudel: Experiences with a Web-Site Management System," *Proc. SIGMOD'98*, pp. 414-425, 1998.

[9] P. C. Fischer, and S. J. Thomas, "Operators for Non-First-Normal-Form Relations," *Proc. IEEE COMPSAC83*, Chicago, pp. 464-475, 1983.

[10] T. Fiebig, J. Weiss, and G. Moerkotte, "RAW: A Relational Algebra for the Web," *Proc. the Workshop on Management of Semi-structured Data*, pp. 34-41, May 1997.

[11] D. Konopnicki and O. Shmueli, "W3QS: A Query System for the World-Wide Web," *Proc. VLDB'95*, pp. 54-65, 1995.

[12] L. Lakshmannan, F. Sadri, and I. Subramanian, "A Declarative Language for Querying and Restructuring the Web," *Proc. the 6th Int. Workshop on Research Issues in Data Eng. (RIDE'96)*, Feb. 1996.

[13] A. O. Mendelzon, G. A. Mihaila, and T. Milo, "Querying the World Wide Web," *Proc. PDIS'96*, pp. 80-91, Dec. 1996.

[14] A. Morishima and H. Kitagawa, "Integrated Querying and Restructuring of the World Wide Web and Databases," *Proc. Intl. Symp. on Digital Media Information Base (DMIB'97)*, pp. 261-271, Nov. 1997.

[15] A. Morishima and H. Kitagawa, "A Visual User Interface for Integration of Heterogeneous Information Sources," *IEICE Transactions on Information and Systems*, Vol. J82-D-I, No. 1, pp.315-326, 1999. (in Japanese)

[16] 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.

[17] Oracle Corporation, "Oracle8 Concepts, Release 8.0," http://www.oracle.com/support/products/o8s/docs805/a58227.pdf.

[18] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom, "Object Exchange Across Heterogeneous Information Sources," *Proc. ICDE'95*, pp. 251-260, Mar. 1995.

[19] RealNetworks, Inc., "RealNetworks Home Page," http://www.real.com/.

[20] M. T. Roth and P. M. Shwarz, "Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources," *Proc. VLDB'97*, pp. 266-275, 1997.

[21] IETF, "Real Time Streaming Protocol," *IETF Request for Comments: 2326*, http://www.ietf.org/rfc/rfc2326.txt.

[22] A. Sahuguet and F. Azavant, "Building light-weight wrappers for legacy Web data-sources using W4F," *Proc. VLDB'99*, 1999.

[23] F. Saltor, M. Castellanos, and M García-Solaco, "Overcoming Schematic Discrepancies in Interoperatable Databases," *Interoperable Database Systems (IFIP Trans. A-25, Proc. DS-5)*, pp. 191-205, 1993.

[24] A. Salminen and F. Tompa, "PAT expressions: an algebra for text search," *Proc. the 2nd International Conference on Computational Lexicography (COMPLEX '92)*, pp. 309-332, 1992.

[25] G. Wiederhold, "Mediators in the Architecture of Future Information Systems," *IEEE Computer*, pp. 38-49, March 1992.

[26] W3C. "Jigsaw Overview," http://www.w3.org/Jigsaw/.

[27] W3C. "Synchronized Multimedia Integration Language (SMIL) 1.0 Specification," W3C Recommendation, http://www.w3.org/TR/REC-smil.

[28] W3C. "XML-QL: A Query Language for XML," W3C note, http://www.w3.org/TR/.

[29] W3C. "XSL Transformations (XSLT) Specification Version 1.0," W3C working draft, http://www.w3.org/TR/.